Migrating between GitHub products with GitHub Enterprise Importer

In this article

Learn how to complete the entire process of migrating from one GitHub product to another, from planning to implementation to completing follow-up tasks.

About migrations between GitHub products

With GitHub Enterprise Importer, you can migrate to GitHub Enterprise Cloud. For more information, see "About GitHub Enterprise Importer".

If you're migrating between GitHub products, such as from GitHub Enterprise Server to GitHub Enterprise Cloud, you can use this guide to plan and implement your migration and complete follow-up tasks. For a full list of supported migration paths, see "Migration support for GitHub Enterprise Importer."

Planning your migration

To plan your migration, ask yourself the following questions.

- Do we want to migrate by organization or by repository?
- How soon do we need to complete the migration?
- Do we understand what will be migrated?
- Who will run the migration?
- Do we want to maintain a similar organization structure after migrating?

Do we want to migrate by organization or by repository?

First, if your migration source and destination are both GitHub.com, decide whether you want to migrate on an organization-by-organization basis or on a repository-by-repository basis.

> **Note:** If you're migrating from GitHub Enterprise Server, you can only migrate repositories.

If you choose repository-by-repository migrations, only repository-level data is migrated. If you pick the organization-by-organization migration strategy, selected organization-level data is also migrated, including teams and their access to repositories.

However, when you migrate an organization, all repositories owned by the source

organization are migrated at the same time. You cannot break the repositories into batches or skip migrating any of the organization's repositories. If you have a large number of repositories, or if you can't tolerate downtime for all your repositories at the same time, you might need to run repository migrations instead.

Additionally, an organization migration creates a new organization in the destination enterprise account. If you want to migrate repositories into an existing organization, you'll need to run repository migrations instead.

Finally, you must be an enterprise owner of the destination enterprise account to migrate organizations. If you want to task someone who is not an enterprise owner perform your migrations, they will need to run repository migrations.

How soon do we need to complete the migration?

Determine your timeline, which will largely dictate your approach. The first step for determining your timeline is to get an inventory of what you need to migrate. To gather this data, use the GitHub Migration Analyzer. This open-source tool generates a report that details how much data there is to migrate for an organization.

- Number of repositories
- Number of pull requests
- Number of issues
- Number of users
- Usage of projects and wikis

Migration timing is largely based on the number of pull requests and issues in a repository. If you want to migrate 1,000 repositories, and each repository has 100 pull requests and issues on average, and only 50 users have contributed to the repositories, your migration will likely be very quick. If you want to migrate only 100 repositories, but the repositories each have 75,000 pull requests and issues on average, and 5,000 users, the migration will take longer and require much more planning and testing.

After you use the analyzer, you can weigh your inventory data against your desired timeline. If your organization can withstand a higher degree of change, then you might be able to migrate all your repositories at once, completing your migration efforts in a few days. However, you may have various teams that are not able to migrate at the same time. In this case, you might want to batch and stagger your migrations to fit the teams' timelines, extending your migration effort.

1. Use the GitHub Migration Analyzer, then review your migration inventory.
2. To understand when teams can be ready to migrate, interview stakeholders.
3. Fully review the rest of this guide, then decide on a migration timeline.

Do we understand what will be migrated?

Ensure that you and your stakeholders understand what data can be migrated by GitHub Enterprise Importer.

1. Review the data that's migrated for your migration source. For more information, see "Migration support for GitHub Enterprise Importer."
2. Make a list of any data that you'll need to manually migrate or recreate.

Who will run the migration?

Decide who will run your migrations, and ensure that this person has the required access. Your options will depend on whether you're migrating by organization or by repository.

Deciding who will run organization migrations

To migrate an organization, you must be an organization owner for the source organization, or an organization owner must grant you the migrator role for that organization.

Additionally, you must be an enterprise owner on the destination enterprise account. You cannot grant the migrator role for enterprise accounts.

1. Confirm that the person who will run your migrations is an enterprise owner of the destination enterprise account.
2. If that person is not an organization owner for the source organization, grant them the migrator role for the organization. For more information, see "Granting the migrator role for GitHub Enterprise Importer."
3. Confirm that the person has correctly configured personal access tokens to meet all the access requirements. For more information, see "Managing access for GitHub Enterprise Importer."

Deciding who will run repository migrations

To migrate a repository, you must be an organization owner for both the source organization and the destination organization, or an organization owner must grant you the migrator role for each organization where you're not an owner.

1. Decide whether you want an organization owner to perform your migrations, or whether you need to grant the migrator role to someone else.
2. If you chose to grant the migrator role, decide which person or team you'll grant the role to.
3. Grant the migrator role to the person or team. For more information, see "Granting the migrator role for GitHub Enterprise Importer."

   > **Note:** Remember to grant the migrator role for both the source organization and the destination organization.

4. Confirm that the person has correctly configured personal access tokens to meet all the access requirements. For more information, see "Managing access for GitHub Enterprise Importer."

Do we want to maintain a similar organization structure after migrating?

Next, consider whether you want to maintain a similar organizational structural after migrating. If you want to break your migration effort into batches, this will help you determine your batches. If you intend to keep a one-to-one correspondence between organizations in your source and destination, then we recommend batching migrations by organization. This is the simplest approach, especially if you're migrating from GitHub.com, because you can migrate an entire organization with one command. If you're migrating from another source, the GitHub CLI can generate a script to migrate all

repositories in a single organization.

If you intend to change your organizational structure, consider other batching factors. You can batch repositories owned by similar teams or a business division, or you can batch by the destination organization. We recommend batching by teams if possible. If you batch by business division or destination organizations, you'll increase the number of stakeholders involved, which can lead to shorter windows of time for your migrations.

Even if you change your organizational structure, you can still prepare a script for your migration. Use the GitHub CLI command, then move the lines for each repository into different scripts as needed.

> **Note:** You can run multiple batches simultaneously. For example, if you're batching by teams, you could run the migrations for multiple teams in the same time window.

1. Decide what your new organization structural will be.
2. Decide if you need to break up your migration effort into smaller batches.
3. If so, decide how you want to break up your migrations.

Running your migrations

After you complete your planning, you can start the actual migrations. To help uncover problems that might be unique to your enterprise during and after the migration, we highly recommend performing trial runs of all migrations. After resolving any issues uncovered by the trial runs, you can run your production migrations.

Trial migrations help you determine several critical pieces of information.

- Whether the migration for a given repository can complete successfully
- Whether you can get the repository back to a state where your users can successfully start working
- How long a migration will take to run, which is useful for planning migration schedules and setting stakeholder expectations

Trial runs do not require much time coordination. GitHub Enterprise Importer never requires downtime for users of a repository being migrated. However, we recommend halting work during production migrations to ensure that new data isn't created during the migration, which would then be missing from the migrated repository. This isn't a concern for trial migrations, so trial runs can take place at any time. To reduce the time it takes to complete your trial migrations, you can schedule the batches for your trial runs back-to-back. Users of those repositories can then validate the results on their own time.

For repository migrations, we recommend creating a test organization to use as a destination for your trial migrations. You can use a single organization for all trial runs, or you can create one test organization for each intended destination organization. Consider including `-sandbox` at the end of the organization names, to clarify that the organizations are intended only for migration validation and not for production. You can delete the test organizations after you're done.

1. If you're running a repository migration, create a test organization for your trial migrations.
2. If your source organization uses IP allow lists, configure the list to allow access by GitHub Enterprise Importer. For more information, see "Managing access for GitHub Enterprise Importer."

3. Run the trial migrations. For more information, see "Preparing to run a migration with GitHub Enterprise Importer."
4. Complete the follow-up tasks described below for the trial migrations.
5. Ask users to validate the results of the migrations.
6. Resolve any issues uncovered by your trial migrations.
7. If your destination uses IP allow lists, configure the list to allow access by GitHub Enterprise Importer. For more information, see "Managing access for GitHub Enterprise Importer."
8. If you're running a repository migration and you want to migrate GitHub Advanced Security settings, enable GitHub Advanced Security for the destination organization. For more information, see "Managing security and analysis settings for your organization."
9. Run your production migrations. For more information, see "Migrating repositories with GitHub Enterprise Importer" or "Migrating organizations with GitHub Enterprise Importer."
10. Optionally, delete the test organization.

Completing follow-up tasks

After each migration has finished, you will need to complete some additional tasks before the repository is ready for work.

- Reviewing the migration log
- Migrating Git LFS objects
- Setting repository visibility
- Configuring GitHub Actions
- Configuring IP allow lists
- Managing GitHub Advanced Security
- Enabling webhooks
- Reinstalling GitHub Apps
- Recreating teams
- Reclaiming mannequins

Reviewing the migration log

First, review the migration log for each migrated repository. For more information, see "Accessing your migration logs for GitHub Enterprise Importer."

If any specific items within the repository could not be migrated, you'll see a warning in the migration log.

> **Note:** If the "Migration Log" issue includes "Migration completed" at the bottom, the repository was migrated. Error messages only indicate that specific items within the repository, such as a comment on a pull request, may not have migrated correctly.

1. Review your migration logs.
2. Review any warnings each log, and ensure that none are blocking the acceptance of the migration. For more information, see "Troubleshooting your migration with

GitHub Enterprise Importer."

Migrating Git LFS objects

GitHub Enterprise Importer does not migrate Git LFS objects. If the source repository uses Git LFS, you can manually push Git LFS objects to the migrated repository locally. For more information, see "Duplicating a repository."

Setting repository visibility

All repositories are migrated as private, and only the user that ran the migration and organization owners will have access to the repository. If you don't want the repository to be private, change the visibility.

- You can change a repository's visibility in the browser. For more information, see "Setting repository visibility."
- Alternatively, you can use GitHub CLI to change repository visibility from the command line. You can even add this command to the script that was generated to run your migrations. For more information, see `gh repo edit` in the GitHub CLI documentation.

Configuring GitHub Actions

If you use GitHub Actions in a repository, your workflows are automatically migrated as part of the Git repository.

During the migration process, GitHub Actions is disabled for all migrated repositories to avoid workflows being accidentally triggered, but GitHub Actions is re-enabled when the migration finishes.

If you were using larger runners, self-hosted runners, or encrypted secrets, you must reconfigure them.

> **Note:** Workflow run history for GitHub Actions is not included in migrations.

1. If you use self-hosted runners, reconfigure your runners.
   - Add runners to the appropriate repository, organization, or enterprise. For more information, see "Adding self-hosted runners."
   - To use runners at the organization or enterprise level, update your workflows. For more information, see "Using self-hosted runners in a workflow."
2. If you use larger runners, reconfigure your runners.
   - Configure runner groups to control access to your runners. For more information, see "Controlling access to larger runners."
   - Set up your larger runners. For more information, see "Managing larger runners."
   - Update your workflows to point to your runners. For more information, see "Running jobs on larger runners."
3. Re-add any encrypted secrets.
   - To use the browser, see "Using secrets in GitHub Actions."
   - To use GitHub CLI, see `gh secret` in the GitHub CLI documentation.

4. Reconfigure environments. For more information, see "Using environments for deployment."

Configuring IP allow lists

If you added the IP ranges for GitHub Enterprise Importer to the IP allow lists for your source or destination organizations, you can remove those entries. If you disabled your identity provider's IP allow list restrictions for your destination enterprise, you can re-enable them now.

For more information, see "Managing access for GitHub Enterprise Importer."

Managing GitHub Advanced Security

If you enabled GitHub Advanced Security for the destination organization before migrating repositories, the settings for individual features were migrated. If not, you'll need to re-enable individual features after the migration. For more information, see "Managing security and analysis settings for your repository."

There are additional post-migration steps for each feature.

Secret scanning

When secret scanning is enabled for the destination repository, a scan of the entire repository will be performed. After the scan is complete, all alerts will be populated, but without remediation states.

You can use the REST API to update the alerts to mirror any remediations in the source repository. For more information, see "Secret scanning" in the REST API documentation.

The user associated with these updated remediations will be the user who owns the personal access token that was used for the API calls, not the user who remediated the alert in the source repository, and the date associated with the remediation will be the date of the API call, not the date the alert was remediated in the source repository.

Code scanning

Code scanning alerts are not migrated by GitHub Enterprise Importer. However, the alerts are available as SARIF data in the source repository. You can use the REST API to upload this data to the destination repository. For more information, see "Code Scanning" in the REST API documentation.

Code scanning alerts that are populated this way will differ from the original alerts in the source repository.

- Alerts will only include the detection and the latest state of the alert, not the entire timeline from the source repository.
- Alerts will only be identified as `open` or `fixed`. Other remediation states, such as `dismissed` and `reopened`, will be lost.
- The dates for all events on the alert will be the date of the API call, not the dates when the events originally occurred on the source repository.
- All actors, such as the alert creator, will change to the owner of the personal access token used for the API call.

Dependabot alerts

When Dependabot alerts and the dependency graph are enabled, Dependabot alerts will be rebuilt from the current state of the default branch. Remediation states of these alerts

are not migrated, and any previous alerts are also not migrated.

You'll need to re-add any encrypted secrets for Dependabot. For more information, see "Configuring access to private registries for Dependabot."

Enabling webhooks

All active webhooks in the source repository are migrated. However, the migrated webhooks will be disabled by default. You can re-enable these webhooks in the repository settings.

1. Navigate to the settings for the migrated repository.
2. In the "Code and automation" section of the sidebar, click **Webhooks**.
3. To the right of the webhook you want to enable, click **Edit**.
4. If you were using a secret token to secure the webhook, under "Secret", re-add the secret.
5. At the bottom of the page, select **Active**.
6. Click **Update webhook**.

Reinstalling GitHub Apps

If you had any GitHub Apps installed on the source repository, you'll need to reinstall them on the migrated repository. For more information, see "Installing your own GitHub App."

Recreating teams

If you migrated on an organization-by-organization basis, you only need to reinstate team membership. If you migrated on a repository-by-repository basis, you will need to recreate teams, give those teams access to repositories, and then reinstate team membership.

Recreating teams for organization migrations

Teams and their repository access are migrated as part of an organization migration, but team membership is not. After your migration, you must add users to the migrated teams.

We highly recommend using team synchronization to manage team membership through your identity provider (IdP). For more information, see "Configuring SCIM provisioning for Enterprise Managed Users" or, for enterprises that do not use Enterprise Managed Users, "Managing team synchronization for organizations in your enterprise."

Otherwise, you can manually add members to your organization, and then add organization members to teams. For more information, see "Adding organization members to a team."

Recreating teams for repository migrations

Teams are not migrated as part of a repository migration. You must manually recreate teams and give each team access to the repository.

1. Re-create teams. For more information, see "Creating a team."

2. Add organization members to teams. For more information, see "Adding organization members to a team."
3. Give each team access to the repository. For more information, see "Managing team access to an organization repository."

Reclaiming mannequins

After you run a migration with GitHub Enterprise Importer, all user activity in the migrated repository (except Git commits) is attributed to placeholder identities called mannequins.

You can reattribute the history for each mannequin to an organization member with the GitHub CLI or in your browser. If you use the GitHub CLI, you can reclaim mannequins in bulk. For more information, see "Reclaiming mannequins for GitHub Enterprise Importer."

> **Note:** Only organization owners can reclaim mannequins. If you've been granted the migrator role, contact an organization owner to perform this step.

1. Decide if you want to reclaim mannequins.
2. Plan when you'll complete reclaims.
3. Reclaim mannequins.
4. If any of the members do not already have appropriate access to the repository via team membership, give the members access to the repository. For more information, see "Managing an individual's access to an organization repository."