Migration support for GitHub Enterprise Importer

In this article

GitHub Enterprise Importer migrates a variety of data to GitHub from our supported sources.

About migration support for GitHub Enterprise Importer

GitHub Enterprise Importer can migrate to GitHub Enterprise Cloud from any of our supported migration sources. The data included in each migration depends on the source.

GitHub Enterprise Importer supports migrations to GitHub Enterprise Cloud from the following sources.

- Azure DevOps (ADO) Cloud
- Bitbucket Server and Bitbucket Data Center 5.14+
- GitHub.com
- GitHub Enterprise Server (GHES) 3.4.1+

During the beta, there are known limitations for the Importer that apply to all sources.

Azure DevOps migration support

If your migration source is Azure DevOps, you can migrate repositories.

You can only use GitHub Enterprise Importer to migrate from Azure DevOps Cloud, not from Azure DevOps Server. If you currently use Azure DevOps Server and want to migrate to GitHub, you can migrate to Azure DevOps Cloud first. For more information, see Migrate to Azure DevOps on the Azure site.

We currently only support migrating the following repository data from Azure DevOps to GitHub Enterprise Cloud.

- Git source (including commit history)
- Pull requests
- User history for pull requests
- Work item links on pull requests

- Attachments on pull requests
- Branch protections for the repository (user-scoped branch protections not included)

If you want to migrate Azure Pipelines to GitHub Actions, contact your GitHub account manager.

Bitbucket Server migration support

Migrations from Bitbucket Server are only supported for Bitbucket Server or Bitbucket Data Center version 5.14+ or higher.

If your migration source is Bitbucket Server, you can migrate repositories. We currently only support migrating the following repository data from Bitbucket Server to GitHub Enterprise Cloud.

- Git source (including commit history)
- Pull requests (including comments, pull request reviews, pull request review comments at the file and line level, required reviewers, and attachments)

Currently, the following data is **not** migrated.

- Personal repositories owned by users
- Branch permissions
- Commit comments
- Repository settings

GitHub Enterprise Importer does not migrate CI pipelines from Bitbucket Server.

GitHub.com migration support

If your migration source is GitHub.com, you can migrate individual repositories or entire organizations.

When you migrate an organization, a new organization is created within the destination enterprise account. Then, the following data is migrated to the new organization.

- Teams
- Repositories
- Team access to repositories
- Member privileges
- Organization-level webhooks
- Default branch name for new repositories created in the organization

All repositories are migrated with private visibility. If you want to set a repository's visibility to public or internal, you can do this after the migration using the UI or API.

Team membership is **not** migrated. After the migration, you'll need to add members to migrated teams. For more information, see "Migrating between GitHub products with GitHub Enterprise Importer."

> **Note:** References to teams, such as `@octo-org/octo-team`, are **not** updated as part of an organization migration. This might cause problems in the destination organization, such as `CODEOWNERS` files not working as expected. For more information about how to prevent and resolve these issues, see "Troubleshooting your migration with GitHub Enterprise Importer."

When you migrate a repository, either directly or as part of an organization migration, only the following data is migrated.

- Git source (including commit history)

- Pull requests
- Issues
- Milestones
- Wikis
- Projects (classic) at the repository level
- GitHub Actions workflows
- Commit comments
- Active webhooks
- Repository topics
- Repository settings
    - Branch protections (see "Branch protections" for more details)
    - GitHub Pages settings
    - Autolink references
    - GitHub Advanced Security settings
    - Pull request settings
        - Automatically delete head branches
        - Allow auto-merge
        - Allow merge commits (commit message setting is reset to the default message)
        - Allow squash merging (commit message setting is reset to the default message)
        - Allow rebase merging
- Releases (up to 10 GB per repository)
- User history for the above data

Currently, the following data is **not** migrated.

- Any Projects (the new projects experience)
- Code scanning results
- Commit status checks
- Dependabot alerts
- Dependabot secrets
- Discussions at the repository level
- Edit history of issue comments and pull request comments
- Fork relationships between repositories (see "About forks")
- GitHub Actions secrets, variables, environments, self-hosted runners, larger runners, or workflow run history
- GitHub Codespaces secrets
- GitHub Apps and GitHub App installations
- Git LFS objects and large binaries (repositories using Git LFS are still supported, see "Limitations of GitHub Enterprise Importer")
- Packages in GitHub Packages
- Projects (classic) at the organization level
- References between pull requests and issues in different repositories (see "Autolinked references and URLs")
- Remediation states of secret scanning results
- Repositories owned by user accounts
- Repository properties (public beta)
- Repository stars
- Repository watchers
- Rulesets

- Tag protection rules
- Users' profiles, SSH keys, signing keys, or personal access tokens
- Webhook secrets
- User access to the repository

When you migrate a repository directly, teams and team access to repositories are not migrated.

Branch protections

Branch protections apply a specified set of rules to a specific branch name or branch name pattern. For more information, see "About protected branches."

Branch protections will always be migrated, but certain rules will not be migrated. The following branch protection rules are not migrated.

- Allow specific actors to bypass required pull requests
- Require approval of the most recent push
- Require deployments to succeed before merging
- Lock branch
- Restrict pushes that create matching branches
- Allow force pushes

The following limitations also apply:

- If a branch protection rule optionally allows you to specify people, teams, or apps that are exempt from the rule, such as "Restrict who can dismiss pull request reviews," the exceptions will not be migrated.
- If the "Allow force pushes" rule is enabled in "Specify who can force push" mode, the rule will not be migrated.

GitHub Enterprise Server migration support

If your migration source is GitHub Enterprise Server, you can migrate repositories.

To migrate from GitHub Enterprise Server (GHES), you must have GHES version 3.4.1 or higher.

Item	GHES 3.4.1+	GHES 3.5.0+
Git source (including commit history)	**X**	**X**
Pull requests	**X**	**X**
Issues	**X**	**X**
Milestones	**X**	**X**
Wikis	**X**	**X**
Projects (classic) at the repository level	**X**	**X**
GitHub Actions workflows	**X**	**X**
Commit comments	**X**	**X**
Active webhooks	**X**	**X**

Branch protections	X	X
GitHub Pages settings	X	X
User history for the above data	X	X
Releases		X

Different size limits per repository apply depending on your GHES version.

Limit	GHES <3.8.0	GHES 3.8.0+
Git source	2GB	10GB
Metadata	2GB	10GB

Currently, the following data is **not** migrated.

- Any Projects (the new projects experience)
- Code scanning results
- Commit status checks
- Dependabot alerts
- Dependabot secrets
- Discussions at the repository level
- Edit history of issue comments and pull request comments
- Fork relationships between repositories (see "About forks")
- GitHub Actions secrets, variables, environments, self-hosted runners, larger runners, or workflow run history
- GitHub Codespaces secrets
- GitHub Apps and GitHub App installations
- Git LFS objects and large binaries (repositories using Git LFS are still supported, see "Limitations of GitHub Enterprise Importer")
- Packages in GitHub Packages
- Projects (classic) at the organization level
- References between pull requests and issues in different repositories (see "Autolinked references and URLs")
- Remediation states of secret scanning results
- Repositories owned by user accounts
- Repository properties (public beta)
- Repository stars
- Repository watchers
- Rulesets
- Tag protection rules
- Users' profiles, SSH keys, signing keys, or personal access tokens
- Webhook secrets
- Teams
- User or team access to the repository
- Repository settings for pull requests

Limitations

There are limits to what GitHub Enterprise Importer can migrate. Some are due to limitations of GitHub.com, while others are limitations of GitHub Enterprise Importer

itself.

Limitations of GitHub.com

- **2 GB size limit for a single Git commit:** No single commit in your Git repository can be larger than 2 GB. If any of your commits are larger than 2 GB, you will need to split the commit into smaller commits that are each 2 GB or smaller.
- **255 byte limit for Git references:** No single Git reference, commonly known as a "ref", can have a name larger than 255 bytes. Usually, this means that your references cannot be more than 255 characters long, but any non-ASCII characters, such as emojis, may consume more than one byte. If any of your Git references are too large, we'll return a clear error message.
- **100 MB file size limit:** No single file in your Git repository can be larger than 100 MB. Consider using Git LFS for storing large files. For more information, see "Managing large files."

Limitations of GitHub Enterprise Importer

- **10 GB size limit for a Git repository:** This limit only applies to the source code. To inspect the size of your repository, use the git-sizer tool and check the total size of blobs.
- **10 GB limit for metadata**: The Importer cannot migrate repositories with more than 10 GB of metadata. Metadata includes issues, pull requests, releases, and attachments. In most cases, large metadata is caused by binary assets attached to releases. You can exclude releases from the migration with the `migrate-repo` command's `--skip-releases` flag, and then move your releases manually after the migration.
- **Git LFS objects not migrated**: The Importer can migrate repositories that use Git LFS, but the LFS objects themselves will not be migrated. They can be pushed to your migration destination as a follow-up task after the migration is complete. For more information, see "Duplicating a repository."
- **Follow-up tasks required:** When migrating between GitHub products, certain settings are not migrated and must be reconfigured in the new repository. For a list of follow-up tasks you'll need to complete after each migration, see "Migrating between GitHub products with GitHub Enterprise Importer."
- **Delayed code search functionality:** Re-indexing the search index can take a few hours after a repository is migrated, and code searches may return unexpected results until re-indexing is complete.
- **Rulesets configured for your organization can cause migrations to fail**: For example, if you configured a rule that requires email addresses for commit authors to end with `@monalisa.cat`, and the repository you're migrating contains commits that don't comply with this rule, your migration will fail. For more information about rulesets, see "About rulesets."