Migrating from Azure DevOps with GitHub Enterprise Importer

In this article

About migrations from Azure DevOps

Planning your migration

Running your migrations

Completing follow-up tasks

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

About migrations from Azure DevOps

With GitHub Enterprise Importer, you can migrate to GitHub Enterprise Cloud on a repository-by-repository basis. For more information, see "About GitHub Enterprise Importer".

If you're migrating from Azure DevOps (ADO), you can use this guide to plan and implement your migration and complete follow-up tasks.

Enterprises who migrate from ADO to GitHub typically follow a multi-phase approach.

1. Migrate repositories from ADO to GitHub.
2. Migrate pipelines from Azure Pipelines to GitHub Actions.
3. Migrate remaining assets, such as boards and artifacts, from ADO to GitHub.

This guide will guide you through completing the first phase, migrating repositories to GitHub, and assumes you're using the ADO2GH extension of the GitHub CLI.

Planning your migration

To plan your migration, ask yourself the following questions.

- How soon do we need to complete the migration?
- Do we understand what will be migrated?
- Who will run the migration?
- What organizational structure do we want in GitHub?

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

Migration timing is largely based on the number of pull requests in a repository. If you want to migrate 1,000 repositories, and each repository has 100 pull requests 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 on average, and 5,000 users, the migration will take much 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.

For migrations from ADO, GitHub Enterprise Importer only migrates Git repositories, including pull requests and some branch policies. Any other assets, such as pipelines, work items, artifacts, test plans, releases, and dashboards, will remain in ADO.

Because permissions work differently in GitHub than in ADO, GitHub Enterprise Importer does not attempt to migrate repository permissions from ADO. For more information, see "Configuring permissions."

Service hooks are not migrated from ADO, so you will need to recreate them separately.

1. Review the data that's migrated from Azure DevOps. 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?

To migrate a repository, you must be an organization owner for the destination organization, or an organization owner must grant you the migrator role.

1. Decide whether you want an organization owner of the destination organization 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."

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."

What organizational structure do we want in GitHub?

Next, plan the organizational structure you'll create in GitHub. ADO and GitHub have different ways of organizing an enterprise's work.

- ADO: Organization > team project > repositories
- GitHub: Enterprise > organization > repositories

> **Note:** The concept of a team project, which is used to group repositories in ADO, does not exist in GitHub. We do not recommend treating organizations in GitHub Enterprise Cloud as the equivalent of team projects in ADO.

After migrating to GitHub, you should have only one enterprise account and a small number of organizations owned by that enterprise. Each organization from ADO should correspond to a single organization on GitHub. We do not recommend creating an organization on GitHub for each team project on ADO.

This may result in a large list of ungrouped repositories within each organization. However, you can manage access to groups of repositories by creating teams. For more information, see "About teams."

If you want to break your migration effort into batches, the new structure can help you determine your batches. If you have more than one organization in ADO, and each organization's repositories are reasonably sized batches, consider batching by organization. You can use the GitHub CLI to generate a migration script for an entire organization on ADO.

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.

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. Create a test organization for your trial migrations.
2. Run the trial migrations. For more information, see "Preparing to run a migration with GitHub Enterprise Importer."
3. Complete the follow-up tasks described below for the trial migrations.
4. Ask users to validate the results of the migrations.
5. Resolve any issues uncovered by your trial migrations.
6. 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."
7. Run your production migrations. For more information, see "Migrating repositories from Azure DevOps to GitHub Enterprise Cloud."
8. 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
- Setting repository visibility
- Configuring permissions
- Reclaiming mannequins
- Configuring IP allow lists

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."

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 permissions

Because permissions work differently in GitHub than in ADO, GitHub Enterprise Importer does not attempt to migrate repository permissions from ADO.

If you used the ADO2GH CLI, GitHub Enterprise Importer will create two teams in GitHub for each team project in ADO. Each team is granted a different level of access to all repositories that originated from the team project.

Team	Access to migrated repositories
TEAM-PROJECT-Maintainers	Maintainer
TEAM-PROJECT-Admins	Admin

To give access to migrated repositories, you can add people to these teams. You can do this manually on GitHub, or if you chose to link the teams to Azure Active Directory (AAD) groups during your migration, by managing group membership in AAD. For more information about manually managing team membership, see "Adding organization members to a team."

If you aren't using the ADO2GH CLI, or if you require a permissions configuration that is more advanced than this default, configure permissions for your migrated repositories. You can modify the migration script to suit your needs, or you can manually configure permissions after your migration. For more information about managing access to repositories on GitHub, see "Repository roles for an organization."

1. Decide what permissions structure you require in GitHub.
2. If different than the default, make a plan for setting up team membership and permissions.

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."

Configuring IP allow lists

If you added the IP ranges for GitHub Enterprise Importer to the IP allow list for your destination organization, 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."