Migrating your enterprise to GitHub Actions

In this article

Learn how to plan a migration to GitHub Actions for your enterprise from another provider.

About enterprise migrations to GitHub Actions

To migrate your enterprise to GitHub Actions from an existing system, you can plan the migration, complete the migration, and retire existing systems.

This guide addresses specific considerations for migrations. For additional information about introducing GitHub Actions to your enterprise, see "Introducing GitHub Actions to your enterprise."

Planning your migration

Before you begin migrating your enterprise to GitHub Actions, you should identify which workflows will be migrated and how those migrations will affect your teams, then plan how and when you will complete the migrations.

Leveraging migration specialists

GitHub can help with your migration, and you may also benefit from purchasing GitHub Professional Services. For more information, contact your dedicated representative or GitHub's Sales team.

Identifying and inventorying migration targets

Before you can migrate to GitHub Actions, you need to have a complete understanding of the workflows being used by your enterprise in your existing system.

First, create an inventory of the existing build and release workflows within your enterprise, gathering information about which workflows are being actively used and need to migrated and which can be left behind.

Next, learn the differences between your current provider and GitHub Actions. This will help you assess any difficulties in migrating each workflow, and where your enterprise might experience differences in features. For more information, see "Migrating to GitHub Actions."

With this information, you'll be able to determine which workflows you can and want to

migrate to GitHub Actions.

Determine team impacts from migrations

When you change the tools being used within your enterprise, you influence how your team works. You'll need to consider how moving a workflow from your existing systems to GitHub Actions will affect your developers' day-to-day work.

Identify any processes, integrations, and third-party tools that will be affected by your migration, and make a plan for any updates you'll need to make.

Consider how the migration may affect your compliance concerns. For example, will your existing credential scanning and security analysis tools work with GitHub Actions, or will you need to use new tools?

Identify the gates and checks in your existing system and verify that you can implement them with GitHub Actions.

Identifying and validating migration tools

Automated migration tools can translate your enterprise's workflows from the existing system's syntax to the syntax required by GitHub Actions. Identify third-party tooling or contact your dedicated representative or GitHub's Sales team to ask about tools that GitHub can provide. For example, you can use the GitHub Actions Importer to plan, scope, and migrate your CI pipelines to GitHub Actions from various supported services. For more information, see "Automating migration with GitHub Actions Importer."

After you've identified a tool to automate your migrations, validate the tool by running the tool on some test workflows and verifying that the results are as expected.

Automated tooling should be able to migrate the majority of your workflows, but you'll likely need to manually rewrite at least a small percentage. Estimate the amount of manual work you'll need to complete.

Deciding on a migration approach

Determine the migration approach that will work best for your enterprise. Smaller teams may be able to migrate all their workflows at once, with a "rip-and-replace" approach. For larger enterprises, an iterative approach may be more realistic. You can choose to have a central body manage the entire migration or you can ask individual teams to self serve by migrating their own workflows.

We recommend an iterative approach that combines active management with self service. Start with a small group of early adopters that can act as your internal champions. Identify a handful of workflows that are comprehensive enough to represent the breadth of your business. Work with your early adopters to migrate those workflows to GitHub Actions, iterating as needed. This will give other teams confidence that their workflows can be migrated, too.

Then, make GitHub Actions available to your larger organization. Provide resources to help these teams migrate their own workflows to GitHub Actions, and inform the teams when the existing systems will be retired.

Finally, inform any teams that are still using your old systems to complete their migrations within a specific timeframe. You can point to the successes of other teams to reassure them that migration is possible and desirable.

Defining your migration schedule

After you decide on a migration approach, build a schedule that outlines when each of

your teams will migrate their workflows to GitHub Actions.

First, decide the date you'd like your migration to be complete. For example, you can plan to complete your migration by the time your contract with your current provider ends.

Then, work with your teams to create a schedule that meets your deadline without sacrificing their team goals. Look at your business's cadence and the workload of each individual team you're asking to migrate. Coordinate with each team to understand their delivery schedules and create a plan that allows the team to migrate their workflows at a time that won't impact their ability to deliver.

Migrating to GitHub Actions

When you're ready to start your migration, translate your existing workflows to GitHub Actions using the automated tooling and manual rewriting you planned for above.

You may also want to maintain old build artifacts from your existing system, perhaps by writing a scripted process to archive the artifacts.

Retiring existing systems

After your migration is complete, you can think about retiring your existing system.

You may want to run both systems side-by-side for some period of time, while you verify that your GitHub Actions configuration is stable, with no degradation of experience for developers.

Eventually, decommission and shut off the old systems, and ensure that no one within your enterprise can turn the old systems back on.