Introducing GitHub Actions to your enterprise

In this article

You can plan how to roll out GitHub Actions in your enterprise.

About GitHub Actions for enterprises

GitHub Actions is a continuous integration and continuous delivery (CI/CD) platform that allows you to automate your build, test, and deployment pipeline. With GitHub Actions, your enterprise can automate, customize, and execute your software development workflows like testing and deployments. For more information, see "About GitHub Actions for enterprises."

Before you introduce GitHub Actions to a large enterprise, you first need to plan your adoption and make decisions about how your enterprise will use GitHub Actions to best support your unique needs.

Governance and compliance

You should create a plan to govern your enterprise's use of GitHub Actions and meet your compliance obligations.

Determine which actions and reusable workflows your developers will be allowed to use. First, decide whether you'll allow third-party actions and reusable workflows that were not created by GitHub. You can configure the actions and reusable workflows that are allowed to run at the repository, organization, and enterprise levels and can choose to only allow actions that are created by GitHub. If you do allow third-party actions and reusable workflows, you can limit allowed actions to those created by verified creators or a list of specific actions and reusable workflows.

For more information, see "Managing GitHub Actions settings for a repository", "Disabling or limiting GitHub Actions for your organization", and "Enforcing policies for GitHub Actions in your enterprise."

Consider combining OpenID Connect (OIDC) with reusable workflows to enforce consistent deployments across your repository, organization, or enterprise. You can do this by defining trust conditions on cloud roles based on reusable workflows. For more information, see "Using OpenID Connect with reusable workflows."

You can access information about activity related to GitHub Actions in the audit logs for your enterprise. If your business needs require retaining this information longer than

audit log data is retained, plan how you'll export and store this data outside of GitHub. For more information, see "Exporting audit log activity for your enterprise" and "Streaming the audit log for your enterprise."

Security

You should plan your approach to security hardening for GitHub Actions.

Security hardening individual workflows and repositories

Make a plan to enforce good security practices for people using GitHub Actions features within your enterprise. For more information about these practices, see "Security hardening for GitHub Actions."

You can also encourage reuse of workflows that have already been evaluated for security. For more information, see "Innersourcing."

Securing access to secrets and deployment resources

You should plan where you'll store your secrets. We recommend storing secrets in GitHub, but you might choose to store secrets in a cloud provider.

In GitHub, you can store secrets at the repository or organization level. Secrets at the repository level can be limited to workflows in certain environments, such as production or testing. For more information, see "Using secrets in GitHub Actions."

You should consider adding manual approval protection for sensitive environments, so that workflows must be approved before getting access to the environments' secrets. For more information, see "Using environments for deployment."

Security considerations for third-party actions

There is significant risk in sourcing actions from third-party repositories on GitHub. If you do allow any third-party actions, you should create internal guidelines that encourage your team to follow best practices, such as pinning actions to the full commit SHA. For more information, see "Security hardening for GitHub Actions."

Innersourcing

Think about how your enterprise can use features of GitHub Actions to innersource automation. Innersourcing is a way to incorporate the benefits of open source methodologies into your internal software development cycle. For more information, see An introduction to innersource in GitHub Resources.

To share actions across your enterprise without publishing the actions publicly, you can store the actions in an internal repository, then configure the repository to allow access to GitHub Actions workflows in other repositories owned by the same organization or by any organization in the enterprise. For more information, see "Sharing actions and workflows with your enterprise."

With reusable workflows, your team can call one workflow from another workflow, avoiding exact duplication. Reusable workflows promote best practice by helping your team use workflows that are well designed and have already been tested. For more information, see "Reusing workflows."

To provide a starting place for developers building new workflows, you can use starter workflows. This not only saves time for your developers, but promotes consistency and best practice across your enterprise. For more information, see "Creating starter

workflows for your organization."

Whenever your workflow developers want to use an action that's stored in a private repository, they must configure the workflow to clone the repository first. To reduce the number of repositories that must be cloned, consider grouping commonly used actions in a single repository. For more information, see "About custom actions."

Managing resources

You should plan for how you'll manage the resources required to use GitHub Actions.

Runners

GitHub Actions workflows require runners. You can choose to use GitHub-hosted runners or self-hosted runners. GitHub-hosted runners are convenient because they are managed by GitHub, who handles maintenance and upgrades for you. However, you may want to consider self-hosted runners if you need to run a workflow that will access resources behind your firewall or you want more control over the resources, configuration, or geographic location of your runner machines. For more information, see "Using GitHub-hosted runners" and "About self-hosted runners."

If you are using self-hosted runners, you have to decide whether you want to use physical machines, virtual machines, or containers. Physical machines will retain remnants of previous jobs, and so will virtual machines unless you use a fresh image for each job or clean up the machines after each job run. If you choose containers, you should be aware that the runner auto-updating will shut down the container, which can cause workflows to fail. You should come up with a solution for this by preventing auto-updates or skipping the command to kill the container.

You also have to decide where to add each runner. You can add a self-hosted runner to an individual repository, or you can make the runner available to an entire organization or your entire enterprise. Adding runners at the organization or enterprise levels allows sharing of runners, which might reduce the size of your runner infrastructure. You can use policies to limit access to self-hosted runners at the organization and enterprise levels by assigning groups of runners to specific repositories or organizations. For more information, see "Adding self-hosted runners" and "Managing access to self-hosted runners using groups." You can also use policies to prevent people using repository-level self-hosted runners. For more information, see "Enforcing policies for GitHub Actions in your enterprise."

You should consider using autoscaling to automatically increase or decrease the number of available self-hosted runners. For more information, see "Autoscaling with self-hosted runners."

Finally, you should consider security hardening for self-hosted runners. For more information, see "Security hardening for GitHub Actions."

Storage

Artifacts enable you to share data between jobs in a workflow and store data once that workflow has completed. For more information, see "Storing workflow data as artifacts."

GitHub Actions also has a caching system that you can use to cache dependencies to speed up workflow runs. For more information, see "Caching dependencies to speed up workflows."

You can use policy settings for GitHub Actions to customize the storage of workflow artifacts, caches, and log retention. For more information, see "Enforcing policies for GitHub Actions in your enterprise."

Some storage is included in your subscription, but additional storage will affect your bill. You should plan for this cost. For more information, see "About billing for GitHub Actions."

Tracking usage

You should consider making a plan to track your enterprise's usage of GitHub Actions, such as how often workflows are running, how many of those runs are passing and failing, and which repositories are using which workflows.

You can see basic details of storage and data transfer usage of GitHub Actions for each organization in your enterprise via your billing settings. For more information, see "Viewing your GitHub Actions usage."

For more detailed usage data, you can use webhooks to subscribe to information about workflow jobs and workflow runs. For more information, see "About webhooks."

Make a plan for how your enterprise can pass the information from these webhooks into a data archiving system. You can consider using "CEDAR.GitHub.Collector", an open source tool that collects and processes webhook data from GitHub. For more information, see the `Microsoft/CEDAR.GitHub.Collector` repository.

You should also plan how you'll enable your teams to get the data they need from your archiving system.