# About permissions for GitHub Packages

**In this article**



Learn about how to manage permissions for your packages.

GitHub Packages is available with GitHub Free, GitHub Pro, GitHub Free for organizations, GitHub Team, GitHub Enterprise Cloud, GitHub Enterprise Server 3.0 or higher, and GitHub AE.

GitHub Packages is not available for private repositories owned by accounts using legacy per-repository plans. Also, accounts using legacy per-repository plans cannot access registries that support granular permissions, because these accounts are billed by repository. For the list of registries that support granular permissions, see "About permissions for GitHub Packages." For more information, see "GitHub's plans."

The permissions for packages can be scoped either to a user or an organization or to a repository.

## Granular permissions for user/organization-scoped packages

Packages with granular permissions are scoped to a personal account or organization. You can change the access control and visibility of the package separately from a repository that is connected (or linked) to a package.

The following GitHub Packages registries support granular permissions.

- Container registry
- npm registry
- NuGet registry
- RubyGems registry

## Permissions for repository-scoped packages

A repository-scoped package inherits the permissions and visibility of the repository in which the package is published. You can find a package scoped to a repository by going to the main page of the repository and clicking the **Packages** link to the right of the page. For more information, see "Connecting a repository to a package."

The following GitHub Packages registries **only** support repository-scoped permissions.

- Docker registry ( `docker.pkg.github.com` )

- Apache Maven registry
- Gradle registry

For other registries, you can choose to allow packages to be scoped to a user or an organization, or linked to a repository.

## Visibility and access permissions for packages

If a package belongs to a registry that supports granular permissions, anyone with admin permissions to the package can set the package to private or public, and can grant access permissions for the package that are separate from the permissions set at the organization and repository levels. For the list of registries that support granular permissions, see "About permissions for GitHub Packages."

In most registries, to pull a package, you must authenticate with a personal access token or `GITHUB_TOKEN`, regardless of whether the package is public or private. However, in the Container registry, public packages allow anonymous access and can be pulled without authentication or signing in via the CLI.

> **Note:** If you publish a package that is linked to a repository, the package inherits its permissions from the linked repository by default. To access the package's granular permissions settings, you must remove the package's inherited permissions. If you're the owner of an organization, you can disable the automatic inheritance of permissions for all new packages scoped to your organization. For more information, see "Configuring a package's access control and visibility" and "Configuring a package's access control and visibility."

When you publish a package, you automatically get admin permissions to the package. If you publish a package to an organization, anyone with the `owner` role in the organization also gets admin permissions to the package.

For packages scoped to a personal account, you can give any person an access role. For packages scoped to an organization, you can give any person or team in the organization an access role.

If you are using a GitHub Actions workflow to manage your packages, you can grant an access role to the repository the workflow is stored in by using the **Add Repository** button under "Manage Actions access" in the package's settings. For more information, see "Configuring a package's access control and visibility."

| Permission | Access description |
| --- | --- |
| Read | Can download package.<br>Can read package metadata. |
| Write | Can upload and download this package.<br>Can read and write package metadata. |
| Admin | Can upload, download, delete, and manage this package.<br>Can read and write package metadata.<br>Can grant package permissions. |

> **Note:** The ability for GitHub Actions workflows to delete and restore packages using the REST API is currently in public beta and subject to change.

For more information, see "Configuring a package's access control and visibility."

# About scopes and permissions for package registries

GitHub Packages only supports authentication using a personal access token (classic). For more information, see "Managing your personal access tokens."

To use or manage a package hosted by a package registry, you must use a personal access token (classic) with the appropriate scope, and your personal account must have appropriate permissions.

For example:

- To download and install packages from a repository, your personal access token (classic) must have the `read:packages` scope, and your user account must have read permission.
- To delete a package on GitHub Enterprise Cloud, your personal access token (classic) must at least have the `delete:packages` and `read:packages` scope. The `repo` scope is also required for repo-scoped packages. For more information, see "Deleting and restoring a package."

| Scope | Description | Required permission |
|---|---|---|
| `read:packages` | Download and install packages from GitHub Packages | read |
| `write:packages` | Upload and publish packages to GitHub Packages | write |
| `delete:packages` | Delete packages from GitHub Packages | admin |
| `repo` | Upload and delete packages (along with `write:packages`, or `delete:packages`) | write or admin |

**Note:** The ability for GitHub Actions workflows to delete and restore packages using the REST API is currently in public beta and subject to change.

When you create a GitHub Actions workflow, you can use the `GITHUB_TOKEN` to publish, install, delete, and restore packages in GitHub Packages without needing to store and manage a personal access token.

For more information, see:

- "Configuring a package's access control and visibility"
- "Publishing and installing a package with GitHub Actions"
- "Managing your personal access tokens"
- "Scopes for OAuth apps"

# About repository transfers

You can transfer a repository to another personal account or organization. For more information, see "Transferring a repository."

When you transfer a repository, GitHub may transfer the packages associated with the repository, depending on the registry the packages belong to.

- For registries that support granular permissions, packages are scoped to a personal account or organization, and the account associated with the package does not change when you transfer a repository. If you have linked a package to a repository, the link is removed when you transfer the repository to another user. Any codespaces or GitHub Actions workflows associated with the repository will lose access to the package. If the package inherited its access permissions from the linked repository, users will lose access to the package. For the list of these registries, see "Granular permissions for user/organization-scoped packages" above.
- For registries that only support repository-scoped permissions, packages are published directly to repositories, and GitHub transfers the packages associated with a repository as part of the repository transfer. All billable usage associated with the packages will subsequently be billed to the new owner of the repository. If the previous repository owner is removed as a collaborator on the repository, they may no longer be able to access the packages associated with the repository. For the list of these registries, see "Permissions for repository-scoped packages" above.

## Maintaining access to packages in GitHub Actions workflows

To ensure your workflows will maintain access to your packages, ensure that you're using the right access token in your workflow and that you've enabled GitHub Actions access to your package.

For more conceptual background on GitHub Actions or examples of using packages in workflows, see "Managing GitHub packages using GitHub Actions workflows."

### Access tokens

> **Note:** The ability for GitHub Actions workflows to delete and restore packages using the REST API is currently in public beta and subject to change.

- To publish, install, delete, and restore packages associated with the workflow repository, use `GITHUB_TOKEN`.
- To install packages associated with other private repositories that `GITHUB_TOKEN` can't access, use a personal access token (classic)

For more information about `GITHUB_TOKEN` used in GitHub Actions workflows, see "Automatic token authentication."

### GitHub Actions access for packages with granular permissions

To ensure your workflows have access to packages stored in registries that support granular permissions, you must give GitHub Actions access to the repositories where your workflow is run. You can find this setting on your package's settings page. For more information, see "Configuring a package's access control and visibility."