Introduction to GitHub Packages

In this article

GitHub Packages is a software package hosting service that allows you to host your software packages privately or publicly and use packages as dependencies in your projects.

About GitHub Packages

GitHub Packages is a platform for hosting and managing packages, including containers and other dependencies. GitHub Packages combines your source code and packages in one place to provide integrated permissions management, so you can centralize your software development on GitHub Enterprise Server.

You can integrate GitHub Packages with GitHub Enterprise Server APIs, GitHub Actions, and webhooks to create an end-to-end DevOps workflow that includes your code, CI, and deployment solutions.

GitHub Packages offers different package registries for commonly used package managers, such as npm, RubyGems, Apache Maven, Gradle, Docker, and NuGet. For more information on the different package registries that GitHub Packages supports, see "Working with a GitHub Packages registry."

You can view a package's README, as well as metadata such as licensing, download statistics, version history, and more on GitHub Enterprise Server. For more information, see "Viewing packages."

For more information about the configuration of GitHub Packages on GitHub Enterprise Server, see "Getting started with GitHub Packages for your enterprise."

Overview of package permissions

The permissions for a package are either inherited from the repository where the package is hosted, or can be defined for specific users or organizations. Some registries only support permissions inherited from a repository. For a list of these registries, see "About permissions for GitHub Packages." For more information on package access, see "Configuring a package's access control and visibility."

Overview of package visibility

You can publish packages in a public repository (public packages) to share with everyone on your enterprise, or in a private repository (private packages) to share with collaborators or an organization.

Supported clients and formats

GitHub Packages uses the native package tooling commands you're already familiar with to publish and install package versions.

Support for package registries

Language	Description	Package format	Package client
JavaScript	Node package manager	`package.json`	`npm`
Ruby	RubyGems package manager	`Gemfile`	`gem`
Java	Apache Maven project management and comprehension tool	`pom.xml`	`mvn`
Java	Gradle build automation tool for Java	`build.gradle` or `build.gradle.kts`	`gradle`
.NET	NuGet package management for .NET	`nupkg`	`dotnet` CLI
N/A	Docker container management	`Dockerfile`	`Docker`

Note: When enabling the Docker registry, we highly recommend also enabling subdomain isolation. For more information, see "Enabling subdomain isolation."

For more information about configuring your package client for use with GitHub Packages, see "Working with a GitHub Packages registry."

Authenticating to GitHub Packages

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

You need an access token to publish, install, and delete private, internal, and public packages.

You can use a personal access token (classic) to authenticate to GitHub Packages or the GitHub Enterprise Server API. When you create a personal access token (classic), you can assign the token different scopes depending on your needs. For more information about packages-related scopes for a personal access token (classic), see "About permissions for GitHub Packages."

To authenticate to a GitHub Packages registry within a GitHub Actions workflow, you can use:

- `GITHUB_TOKEN` to publish packages associated with the workflow repository.
- a personal access token (classic) with at least `read:packages` scope to install packages associated with other private repositories (which `GITHUB_TOKEN` can't access).

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

Managing packages

You can delete a package in the GitHub Enterprise Server user interface or using the REST API. For more information, see "Deleting and restoring a package" and the "Packages." For certain registries, you can use GraphQL to delete a version of a private package.

You cannot use the GitHub Packages GraphQL API with registries that support granular permissions. For the registries that **only** support repository-scoped permissions, and can be used with the GraphQL API, see "About permissions for GitHub Packages."

When you use the GraphQL API to query and delete private packages, you must use the same personal access token (classic) you use to authenticate to GitHub Packages.

For more information, see "Deleting and restoring a package" and "Forming calls with GraphQL."

You can configure webhooks to subscribe to package-related events, such as when a package is published or updated. For more information, see the "Webhook events and payloads."

Contacting support

If you need support for GitHub Packages, please contact your site administrators.