Working with the npm registry

In this article

Authenticating to GitHub Packages
Publishing a package
Publishing multiple packages to the same repository
Installing a package

You can configure npm to publish packages to GitHub Packages and to use packages stored on GitHub Packages as dependencies in an npm project.

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."

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 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).

Authenticating in a GitHub Actions workflow

This registry supports granular permissions. For registries that support granular permissions, if your GitHub Actions workflow is using a personal access token to authenticate to a registry, we highly recommend you update your workflow to use the

`GITHUB_TOKEN`. For guidance on updating your workflows that authenticate to a registry with a personal access token, see "Publishing and installing a package with GitHub Actions."

> **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.

You can use a `GITHUB_TOKEN` in a GitHub Actions workflow to delete or restore a package using the REST API, if the token has `admin` permission to the package. Repositories that publish packages using a workflow, and repositories that you have explicitly connected to packages, are automatically granted `admin` permission to packages in the repository.

For more information about the `GITHUB_TOKEN`, see "Automatic token authentication." For more information about the best practices when using a registry in actions, see "Security hardening for GitHub Actions."

You can also choose to give access permissions to packages independently for GitHub Codespaces and GitHub Actions. For more information, see "Configuring a package's access control and visibility" and "Configuring a package's access control and visibility."

Authenticating with a personal access token

You must use a personal access token (classic) with the appropriate scopes to publish and install packages in GitHub Packages. For more information, see "Introduction to GitHub Packages."

You can authenticate to GitHub Packages with npm by either editing your per-user *~/.npmrc* file to include your personal access token (classic) or by logging in to npm on the command line using your username and personal access token.

To authenticate by adding your personal access token (classic) to your *~/.npmrc* file, edit the *~/.npmrc* file for your project to include the following line, replacing TOKEN with your personal access token. Create a new *~/.npmrc* file if one doesn't exist.

```
//npm.pkg.github.com/:_authToken=TOKEN
```

To authenticate by logging in to npm, use the `npm login` command, replacing USERNAME with your GitHub username, TOKEN with your personal access token (classic), and PUBLIC-EMAIL-ADDRESS with your email address.

If you are using npm CLI version 9 or greater and are logging in or out of a private registry using the command line, you should use the `--auth-type=legacy` option to read in your authentication details from prompts instead of using the default login flow through a browser. For more information, see `npm-login`.

If GitHub Packages is not your default package registry for using npm and you want to use the `npm audit` command, we recommend you use the `--scope` flag with the namespace that hosts the package (the personal account or organization to which the package is scoped) when you authenticate to GitHub Packages.

```
$ npm login --scope=@NAMESPACE --auth-type=legacy --
registry=https://npm.pkg.github.com

> Username: USERNAME
> Password: TOKEN
```

Publishing a package

Note:

- Package names and scopes must only use lowercase letters.
- The tarball for an npm version must be smaller than 256MB in size.

The GitHub Packages registry stores npm packages within your organization or personal account, and allows you to associate a package with a repository. You can choose whether to inherit permissions from a repository, or set granular permissions independently of a repository.

When you first publish a package, the default visibility is private. To change the visibility or set access permissions, see "Configuring a package's access control and visibility." For more information on linking a published package with a repository, see "Connecting a repository to a package."

You can connect a package to a repository as soon as the package is published by including a `repository` field in the `package.json` file. You can also use this method to connect multiple packages to the same repository. For more information, see "Publishing multiple packages to the same repository."

Note: If you publish a package that is linked to a repository, the package automatically inherits the access permissions of the linked repository, and GitHub Actions workflows in the linked repository automatically get access to the package, unless your organization has disabled automatic inheritance of access permissions. For more information, see "Configuring a package's access control and visibility."

You can set up the scope mapping for your project using either a local `.npmrc` file in the project or using the `publishConfig` option in the `package.json`. GitHub Packages only supports scoped npm packages. Scoped packages have names with the format of `@NAMESPACE/PACKAGE-NAME`. Scoped packages always begin with an `@` symbol. You may need to update the name in your `package.json` to use the scoped name. For example, if you're the user `octocat` and your package is named `test`, you would assign the scoped package name as follows: `"name": "@octocat/test"`.

After you publish a package, you can view the package on GitHub. For more information, see "Viewing packages."

Publishing a package using a local *.npmrc* file

You can use an *.npmrc* file to configure the scope mapping for your project. In the *.npmrc* file, use the GitHub Packages URL and account owner so GitHub Packages knows where to route package requests. Using an *.npmrc* file prevents other developers from accidentally publishing the package to npmjs.org instead of GitHub Packages.

1. Authenticate to GitHub Packages. For more information, see "Authenticating to GitHub Packages."
2. In the same directory as your `package.json` file, create or edit an `.npmrc` file to include a line specifying GitHub Packages URL and the namespace where the package is hosted. Replace `NAMESPACE` with the name of the user or organization account to which the package will be scoped.

   ```
   @NAMESPACE:registry=https://npm.pkg.github.com
   ```

3. Add the *.npmrc* file to the repository where GitHub Packages can find your project. For more information, see "Adding a file to a repository."
4. Verify the name of your package in your project's `package.json`. The `name` field

must contain the scope and the name of the package. For example, if your package is called "test", and you are publishing to the "My-org" GitHub organization, the `name` field in your `package.json` should be `@my-org/test`.

5. Verify the `repository` field in your project's *package.json*. The `repository` field must match the URL for your GitHub repository. For example, if your repository URL is `github.com/my-org/test` then the repository field should be `https://github.com/my-org/test.git`.

6. Publish the package:

```
npm publish
```

Publishing a package using `publishConfig` in the `package.json` file

You can use `publishConfig` element in the `package.json` file to specify the registry where you want the package published. For more information, see "publishConfig" in the npm documentation.

1. Edit the `package.json` file for your package and include a `publishConfig` entry.

```
"publishConfig": {
  "registry": "https://npm.pkg.github.com"
},
```

2. Verify the `repository` field in your project's *package.json*. The `repository` field must match the URL for your GitHub repository. For example, if your repository URL is `github.com/my-org/test` then the repository field should be `https://github.com/my-org/test.git`.

3. Publish the package:

```
npm publish
```

Publishing multiple packages to the same repository

To publish multiple packages and link them to the same repository, you can include the URL of the GitHub repository in the `repository` field of the `package.json` file for each package. For more information, see "Creating a package.json file" and "Creating Node.js modules" in the npm documentation.

To ensure the repository's URL is correct, replace `REPOSITORY` with the name of the repository containing the package you want to publish, and `OWNER` with the name of the personal account or organization on GitHub that owns the repository.

GitHub Packages will match the repository based on the URL.

```
"repository":"https://github.com/OWNER/REPOSITORY",
```

Installing a package

You can install packages from GitHub Packages by adding the packages as dependencies in the `package.json` file for your project. For more information on using a `package.json` in your project, see "Working with package.json" in the npm documentation.

By default, you can add packages from one organization. For more information, see "Installing packages from other organizations."

You also need to add the *.npmrc* file to your project so that all requests to install packages will go through GitHub Packages. When you route all package requests through GitHub Packages, you can use both scoped and unscoped packages from *npmjs.org*. For more information, see "npm-scope" in the npm documentation.

1. Authenticate to GitHub Packages. For more information, see "Authenticating to GitHub Packages."
2. In the same directory as your `package.json` file, create or edit an `.npmrc` file to include a line specifying GitHub Packages URL and the namespace where the package is hosted. Replace `NAMESPACE` with the name of the user or organization account to which the package will be scoped.

   ```
   @NAMESPACE:registry=https://npm.pkg.github.com
   ```

3. Add the *.npmrc* file to the repository where GitHub Packages can find your project. For more information, see "Adding a file to a repository."
4. Configure `package.json` in your project to use the package you are installing. To add your package dependencies to the `package.json` file for GitHub Packages, specify the full-scoped package name, such as `@my-org/server`. For packages from *npmjs.com*, specify the full name, such as `@babel/core` or `lodash`. Replace `ORGANIZATION_NAME/PACKAGE_NAME` with your package dependency.

   ```
   {
     "name": "@my-org/server",
     "version": "1.0.0",
     "description": "Server app that uses the ORGANIZATION_NAME/PACKAGE_NAME
   package",
     "main": "index.js",
     "author": "",
     "license": "MIT",
     "dependencies": {
       "ORGANIZATION_NAME/PACKAGE_NAME": "1.0.0"
     }
   }
   ```

5. Install the package.

   ```
   npm install
   ```

Installing packages from other organizations

By default, you can only use GitHub Packages packages from one organization. If you'd like to route package requests to multiple organizations and users, you can add additional lines to your *.npmrc* file, replacing `NAMESPACE` with the name of the personal account or organization to which the package is scoped.

```
@NAMESPACE:registry=https://npm.pkg.github.com
```

```
@NAMESPACE:registry=https://npm.pkg.github.com
```

Legal