Connecting a repository to a package

In this article

You can connect a repository to a package on GitHub.com.

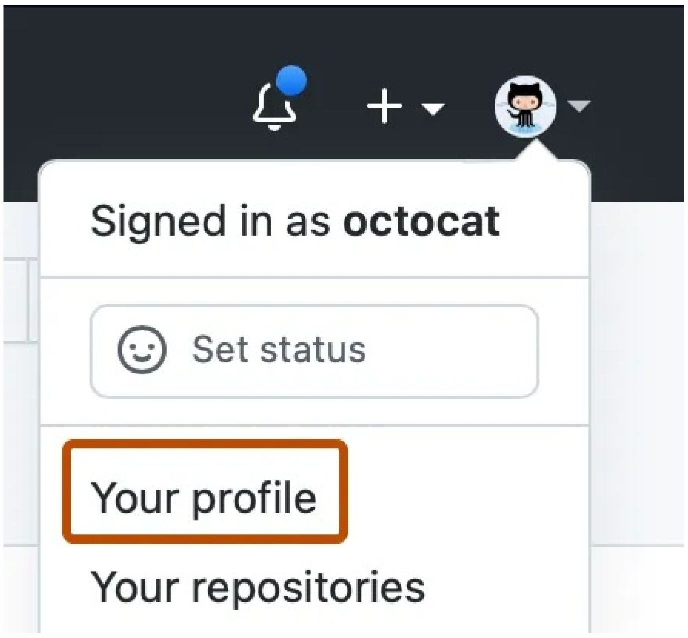
GitHub Packages is available with GitHub Free, GitHub Pro, GitHub Free for organizations, GitHub Team, GitHub Enterprise Cloud, GitHub Enterprise Server 3.0 or higher, and GitHub AE.

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."

When you publish a package that is scoped to a personal account or an organization, the package is not linked to a repository by default. If you connect a package to a repository, the package's landing page will show information and links from the repository, such as the README. You can also choose to have the package inherit its access permissions from the linked repository. For more information, see "Configuring a package's access control and visibility."

Connecting a repository to a user-scoped package on GitHub

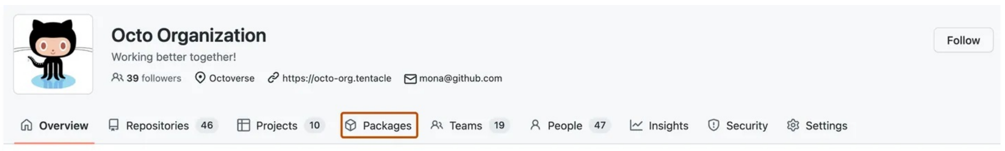
1. On GitHub, navigate to the main page of your personal account.
2. In the top right corner of GitHub.com, click your profile photo, then click **Your profile**.



3. On your profile page, in the header, click the **Packages** tab.
4. Search for and then click the name of the package that you want to manage.
5. Under your package versions, click **Connect repository**.
6. Select a repository to link to the package, then click **Connect repository**.

Connecting a repository to an organization-scoped package on GitHub

1. On GitHub, navigate to the main page of your organization.
2. Under your organization name, click the **Packages** tab.



3. Search for and then click the name of the package that you want to manage.
4. Under your package versions, click **Connect repository**.
5. Select a repository to link to the package, then click **Connect repository**.

Connecting a repository to a container image using the command line

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."

1. In your Dockerfile, add this line, replacing `OWNER` and `REPO` with your details:

   ```
   LABEL org.opencontainers.image.source=https://github.com/OWNER/REPO
   ```

   For example, if you're the user `octocat` and own `my-repo` you would add this line to your Dockerfile:

   ```
   LABEL org.opencontainers.image.source=https://github.com/octocat/my-repo
   ```

   For more information, see "LABEL" in the official Docker documentation and "Pre-defined Annotation Keys" in the `opencontainers/image-spec` repository.

2. Build your container image. This example builds an image from the Dockerfile in the current directory and assigns the image name `hello_docker`.

   ```
   docker build -t hello_docker .
   ```

3. Optionally, review the details of the Docker image you just created.

   ```
   $ docker images
   > REPOSITORY         TAG         IMAGE ID       CREATED          SIZE
   > hello_docker       latest      142e665b1faa   5 seconds ago    125MB
   > redis              latest      afb5e116cac0   3 months ago     111MB
   > alpine             latest      a6215f271958   5 months ago     5.29MB
   ```

4. Assign a name and hosting destination to your Docker image.

   ```
   docker tag IMAGE_NAME ghcr.io/NAMESPACE/NEW_IMAGE_NAME:TAG
   ```

   Replace `NAMESPACE` with the name of the personal account or organization to which you want the package to be scoped.

   For example:

   ```
   docker tag 38f737a91f39 ghcr.io/octocat/hello_docker:latest
   ```

5. If you haven't already, authenticate to the Container registry. For more information, see "Working with the Container registry."

   ```
   $ echo $CR_PAT | docker login ghcr.io -u USERNAME --password-stdin
   > Login Succeeded
   ```

6. Push your container image to the Container registry.

   ```
   docker push ghcr.io/NAMESPACE/IMAGE-NAME:TAG
   ```

   For example:

   ```
   docker push ghcr.io/octocat/hello_docker:latest
   ```