Allowing your codespace to access a private registry

In this article

You can allow GitHub Codespaces to access container images or other packages in a private registry.

About private registries and GitHub Codespaces

A registry is a secure space for storing, managing, and fetching container images or other packages. There are many examples of registries, such as:

- GitHub's Container registry, the Azure Container Registry, and DockerHub for container images
- The npm registry for Node.js packages.

Certain GitHub Packages registries, including the Container registry, can be configured to allow packages to be pulled seamlessly into GitHub Codespaces during codespace creation, without having to provide any authentication credentials.

To access other container image registries, you can create secrets in GitHub to store the access details, which will allow GitHub Codespaces to access images stored in that registry.

Accessing packages stored in registries with granular permissions

GitHub Packages registries that support granular permissions, including the Container registry, provide the easiest way for GitHub Codespaces to consume packages. For the list of GitHub Packages registries that support granular permissions and seamless GitHub Codespaces access, see "About permissions for GitHub Packages."

Accessing a package published to the same repository as the codespace

If you publish a package in the same repository that the codespace is being launched in, you will automatically be able to fetch that package on codespace creation. You won't have to provide any additional credentials, unless the **Inherit access from repo** option was unselected when the package was published.

Inheriting access from the repository from which a package was published

By default, the package inherits the access setting of the repository from which it was published. For example, if the repository is public, the package is also public. If the repository is private, the package is also private, but is accessible from the repository.

This behavior is controlled by the **Inherit access from repo** option. **Inherit access from repo** is selected by default when publishing via GitHub Actions, but not when publishing directly to a registry using a personal access token.

If the **Inherit access from repo** option was not selected when the package was published, you can manually add the repository to the published package's access controls. For more information, see "Configuring a package's access control and visibility."

Accessing a package published to the organization a codespace will be launched in

If you want a package to be accessible to all codespaces in an organization, we recommend that you publish the package with internal visibility. This will automatically make the package visible to all codespaces within the organization, unless the repository the codespace is launched from is public.

If the codespace is being launched from a public repository referencing an internal or private package, you must manually allow the public repository access to the internal package. This prevents the internal package from being accidentally leaked publicly. For more information, see "Configuring a package's access control and visibility."

Accessing a private package from a subset of repositories in an organization

If you want to allow a subset of an organization's repositories to access a package, or allow an internal or private package to be accessed from a codespace launched in a public repository, you can manually add repositories to a package's access settings. For more information, see "Configuring a package's access control and visibility."

Publishing a package from a codespace

Seamless access from a codespace to a registry is limited to pulling packages. If you want to publish a package from inside a codespace, you must use a personal access token (classic) with the `write:packages` scope.

We recommend publishing packages via GitHub Actions. For more information, see "Publishing Docker images" and "Publishing Node.js packages."

Accessing images stored in other registries

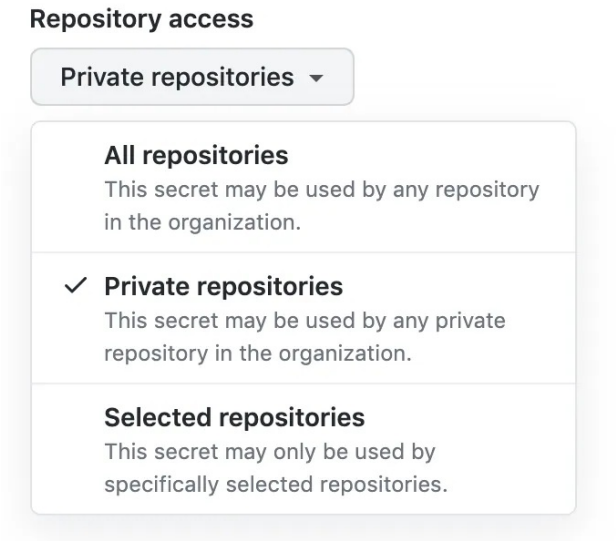
You can define secrets to allow GitHub Codespaces to access container image registries other than GitHub's Container registry. If you are accessing a container image from a registry that doesn't support seamless access, GitHub Codespaces checks for the presence of three secrets, which define the server name, username, and personal access token for a registry. If these secrets are found, GitHub Codespaces will make the registry available inside your codespace.

- `<*>_CONTAINER_REGISTRY_SERVER`
- `<*>_CONTAINER_REGISTRY_USER`
- `<*>_CONTAINER_REGISTRY_PASSWORD`

You can store secrets at the user, repository, or organization-level, allowing you to share them securely between different codespaces. When you create a set of secrets for a

private image registry, you need to replace the "<*>" in the name with a consistent identifier. For more information, see "Managing secrets for your codespaces" and "Managing secrets for your repository and organization for GitHub Codespaces."

If you are setting the secrets at the user or organization level, make sure to assign those secrets to the repository you'll be creating the codespace in by choosing an access policy from the dropdown list.

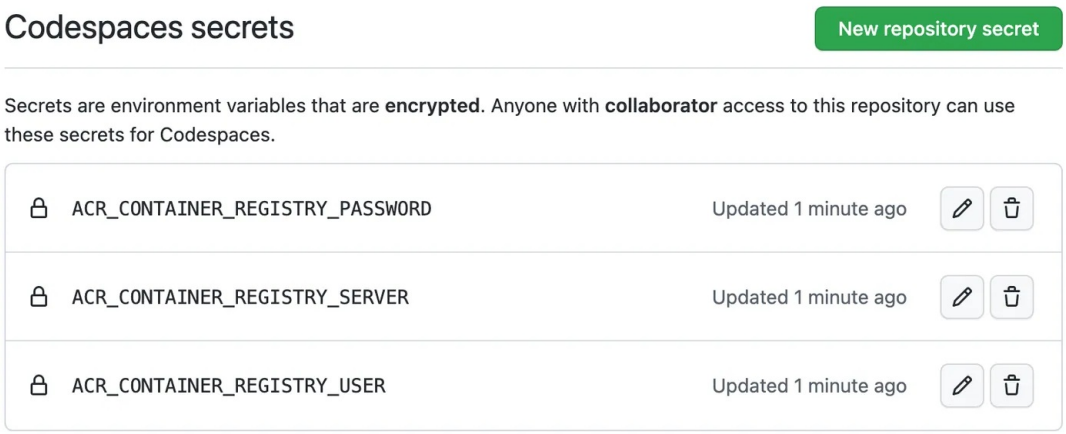


Example secrets

For a private image registry in Azure, you could create the following secrets:

```
ACR_CONTAINER_REGISTRY_SERVER = mycompany.azurecr.io
ACR_CONTAINER_REGISTRY_USER = acr-user-here
ACR_CONTAINER_REGISTRY_PASSWORD = <PERSONAL_ACCESS_TOKEN>
```

For information on common image registries, see "Common image registry servers." Note that accessing AWS Elastic Container Registry (ECR) is different.



Once you've added the secrets, you may need to stop and then start the codespace you are in for the new environment variables to be passed into the container. For more information, see "Using the Visual Studio Code Command Palette in GitHub Codespaces."

Accessing AWS Elastic Container Registry

To access AWS Elastic Container Registry (ECR), you can provide an AWS access key ID and secret key, and GitHub can retrieve an access token for you and log in on your behalf.

```
*_CONTAINER_REGISTRY_SERVER = <ECR_URL>
*_CONTAINER_REGISTRY_USER = <AWS_ACCESS_KEY_ID>
*_CONTAINER_REGISTRY_PASSWORD = <AWS_SECRET_KEY>
```

You must also ensure you have the appropriate AWS IAM permissions to perform the credential swap (e.g. `sts:GetServiceBearerToken`) as well as the ECR read operation (either `AmazonEC2ContainerRegistryFullAccess` or `ReadOnlyAccess`).

Alternatively, if you don't want GitHub to perform the credential swap on your behalf, you can provide an authorization token fetched via AWS's APIs or CLI.

```
*_CONTAINER_REGISTRY_SERVER = <ECR_URL>
*_CONTAINER_REGISTRY_USER = AWS
*_CONTAINER_REGISTRY_PASSWORD = <TOKEN>
```

Since these tokens are short lived and need to be refreshed periodically, we recommend providing an access key ID and secret.

While these secrets can have any name, so long as the `*_CONTAINER_REGISTRY_SERVER` is an ECR URL, we recommend using `ECR_CONTAINER_REGISTRY_*` unless you are dealing with multiple ECR registries.

For more information, see AWS ECR's "Private registry authentication documentation."

Common image registry servers

Some of the common image registry servers are listed below:

- DockerHub - `https://index.docker.io/v1/`
- GitHub Container Registry - `ghcr.io`
- Azure Container Registry - `<registry name>.azurecr.io`
- AWS Elastic Container Registry - `<aws_account_id>.dkr.ecr.<region>.amazonaws.com`
- Google Cloud Container Registry - `gcr.io` (US), `eu.gcr.io` (EU), `asia.gcr.io` (Asia)

Debugging private image registry access

If you are having trouble pulling an image from a private image registry, make sure you are able to run `docker login -u <user> -p <password> <server>`, using the values of the secrets defined above. If login fails, ensure that the login credentials are valid and that you have the appropriate permissions on the server to fetch a container image. If login succeeds, make sure that these values are copied appropriately into the right GitHub Codespaces secrets, either at the user, repository, or organization level and try again.