



# Allowing a prebuild to access other repositories

### In this article

Allowing a prebuild read access to external resources
Allowing a prebuild write access to external resources
Further reading

You can permit your prebuild to access other GitHub repositories so that it can be built successfully.

#### Who can use this feature

People with admin access to a repository can configure prebuilds for the repository.

Repository-level settings for GitHub Codespaces are available for all repositories owned by personal accounts.

For repositories owned by organizations, repository-level settings for GitHub Codespaces are available for organizations on GitHub Team and GitHub Enterprise plans. To access the settings, the organization or its parent enterprise must have added a payment method and set a spending limit for GitHub Codespaces. For more information, see "Choosing who owns and pays for codespaces in your organization" and "GitHub's plans."

By default, the GitHub Actions workflow for a prebuild configuration can only access its own repository contents. Your project may use additional resources, located elsewhere, to build the development environment.

## Allowing a prebuild read access to external resources *♂*

You can configure read access to other GitHub repositories, with the same repository owner, by specifying permissions in the devcontainer.json file used by your prebuild configuration. For more information, see "Managing access to other repositories within your codespace."

**Note**: You can only authorize read permissions in this way, and the owner of the target repository must be the same as the owner of the repository for which you're creating a prebuild. For example, if you're creating a prebuild configuration for the octo-org/octocat repository, then you'll be able to grant read permissions for other repositories, such as octo-org/octodemo, if this is specified in the devcontainer.json file, and provided you have the permissions yourself.

When you create or edit a prebuild configuration for a devcontainer.json file that sets up read access to other repositories with the same repository owner, you'll be prompted to grant these permissions when you click **Create** or **Update**. For more information, see "Configuring prebuilds."

### resources &

If your project requires write access to resources, or if the external resources reside in a repository with a different owner than the repository for which you are creating a prebuild configuration, you can use a personal access token to grant this access.

You will need to create a new personal account and then use this account to create a personal access token (classic) with the appropriate scopes.

1 Create a new personal account on GitHub.

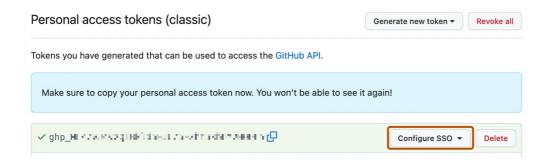
**Warning**: Although you can generate the personal access token (classic) using your existing personal account, we strongly recommend creating a new account with access only to the target repositories required for your scenario. This is because the access token's repository permission grants access to all of the repositories that the account has access to. For more information, see "Signing up for a new GitHub account" and "Security hardening for GitHub Actions."

- 2 Give the new account read access to the required repositories. For more information, see "Managing an individual's access to an organization repository."
- While signed into the new account, create a personal access token (classic) with the repo scope. Optionally, if the prebuild will need to download packages from the GitHub Container registry, also select the read:packages scope. For more information, see "Managing your personal access tokens."

repo	Full control of private repositories		
repo:status repo_deployment public_repo repo:invite	Access commit status Access deployment status Access public repositories Access repository invitations		
		security_events	Read and write security events
		workflow	Update GitHub Action workflows
		write:packages	Upload packages to GitHub Package Registry
read:packages	Download packages from GitHub Package Registry		
delete:packages	Delete packages from GitHub Package Registry		

If the prebuild will use a package from the GitHub Container registry, you will need to either grant the new account access to the package or configure the package to inherit the access permissions of the repository you are prebuilding. For more information, see "Configuring a package's access control and visibility."

4 Click **Configure SSO** and authorize the token for use with SAML single sign-on (SSO), so that it can access repositories that are owned by organizations with SSO enabled. For more information, see "Authorizing a personal access token for use with SAML single sign-on."



- **5** Copy the token string. You will assign this to a Codespaces repository secret.
- 6 Sign back into the account that has admin access to the repository.
- In the repository for which you want to create GitHub Codespaces prebuilds, create a new Codespaces repository secret called CODESPACES\_PREBUILD\_TOKEN, giving it the value of the token you created and copied. For more information, see "Managing secrets for your repository and organization for GitHub Codespaces."

The personal access token will be used for all subsequent prebuilds created for your repository. Unlike other Codespaces repository secrets, the CODESPACES\_PREBUILD\_TOKEN secret is only used for prebuilding and will not be available for use in codespaces created from your repository.

### Further reading @

- "Configuring prebuilds"
- "Troubleshooting prebuilds"

### Legal

© 2023 GitHub, Inc. <u>Terms</u> <u>Privacy</u> <u>Status</u> <u>Pricing</u> <u>Expert services</u> <u>Blog</u>