# Troubleshooting authentication to a repository

**In this article**



Learn how to troubleshoot common authentication issues when you clone, push to, or pull from a repository in a codespace.

When you create a codespace for a repository, you can generally use `git pull` and `git push` to pull and push changes to that repository without any additional authentication. However, occasionally you may see authentication errors when trying to run these operations.

You may also get errors if you are trying to interact with a repository other than the one from which you created the codespace.

## Authenticating to the repository you created the codespace from

If you're trying to push to or pull from the repository from which you created the codespace, but authentication is failing, you may see an error like `git@github.com: Permission denied (publickey)` or `Host key verification failed`.

You may see these errors if you use a dotfiles repository with GitHub Codespaces, and you have configured Git to use a protocol other than HTTPS for transferring data to the remote repository. For example, you may have configured Git to use SSH by including lines like the following in a config file in your dotfiles.

```
[url "git@github.com:"]
  insteadOf = https://github.com/
```

GitHub Codespaces uses the HTTPS protocol by default, and authenticates with a `GITHUB_TOKEN` configured with read and write access to the repository from which you created the codespace. We recommend you use the default HTTPS and `GITHUB_TOKEN` in your codespace. The permissions of the `GITHUB_TOKEN` are usually limited to just one repository, following the security principle of least privilege. SSH authentication does not have fine-grained repository permissions, so an accidental exposure of your SSH key could give someone access to all your repositories.

To use the default HTTPS, remove the conflicting configuration from your dotfiles. If your dotfiles repository contains an installation script in a recognized file such as `install.sh`, you can use logic like the following to exclude the configuration in codespaces.

```
if [ -z "$CODESPACES" ]; then
  git config --global url."git@github.com".insteadOf "https://github.com"
fi
```

If you're working in a codespace created from a repository you trust, and you need to use SSH, ensure that your codespace is set up to authenticate with an SSH key that is linked to your GitHub account. For more information, see "Generating a new SSH key and adding it to the ssh-agent."

## Authenticating to repositories that you didn't create the codespace from

The `GITHUB_TOKEN` in a codespace is configured with read and write access to the repository from which you created the codespace. By default, the token does not have access to other repositories. You may find you cannot clone a repository, or you cannot push to a repository you have cloned.

We do not recommend manually updating the value of the `GITHUB_TOKEN` in a codespace. If your project requires access to other repositories, you can give codespaces access to these repositories by listing additional permissions in your dev container configuration. This will allow users to authorize the additional permissions when they create a codespace. However, it will not change the permissions of an existing codespace. For more information, see "Managing access to other repositories within your codespace."

If you need access to another repository in an existing codespace, or if the permissions you need are specific to you and don't apply to other contributors, you can create a personal access token with access to the repository and add the token to your codespace. We recommend you limit the token's access by using a fine-grained personal access token, selecting only the repositories to which you need access, and giving the required access to the **Contents** permission only. For more information, see "Managing your personal access tokens."

You can then add the token as an environment variable in a codespace, or as a secret for GitHub Codespaces. If you create a secret, you should only allow certain trusted repositories to access the secret. When you add a new secret, you will be prompted to reload your existing codespace to pull in the new secret. For more information, see "Managing secrets for your codespaces."

To use the token to authenticate in your codespace, you have the following options.

- When you create the environment variable or secret, you can use the name `GH_TOKEN`. The `GH_TOKEN` variable is used by default in GitHub CLI operations, so you can clone the repository using the command `gh repo clone OWNER/REPO`.

  However, if you then try to push to the repository using `git push`, Git's credential helper will try to use the existing `GITHUB_TOKEN` to authenticate, and authentication will fail. You can override the helper, but this may introduce friction when you try to interact with the original repository from which you created the codespace.

- You can clone the repository with a URL that includes the access token. Replace `YOUR-VARIABLE` with the name of the environment variable or secret you created.

  ```
  git clone https://PAT:$YOUR-VARIABLE@github.com/OWNER/REPO`
  ```

  This will store the access token for the specific repository, so you will be able to push to and pull from the repository without overriding the existing credential helper.

  > **Note:** If you clone in this way, the token will be visible in your Git configuration. You should only use this method when working in a codespace created from a repository you trust, and you should limit the scope of the access token as much as possible.