Security in GitHub Codespaces

In this article

Overview of the GitHub Codespaces security architecture, with guidelines to help you maintain security and minimize the risk of attack.

Overview of codespace security

GitHub Codespaces is designed to be security hardened by default. Consequently, you will need to ensure that your software development practices do not risk reducing the security posture of your codespace.

This guide describes the way GitHub Codespaces keeps your development environment secure and provides some of the good practices that will help maintain your security as you work. As with any development tool, remember that you should only open and work within repositories you know and trust.

Environment isolation

GitHub Codespaces is designed to keep your codespaces separate from each other, with each using its own virtual machine and network.

Isolated virtual machines

Each codespace is hosted on its own newly-built virtual machine (VM). Two codespaces are never co-located on the same VM.

Every time you restart a codespace, it's deployed to a new VM with the latest available security updates.

Isolated networking

Each codespace has its own isolated virtual network. We use firewalls to block incoming connections from the internet and to prevent codespaces from communicating with each other on internal networks. By default, codespaces are allowed to make outbound connections to the internet.

Authentication

You can connect to a codespace using a web browser or from Visual Studio Code. If you connect from VS Code, you are prompted to authenticate with GitHub Enterprise Cloud.

Every time a codespace is created or restarted, it's assigned a new GitHub token with an automatic expiry period. This period allows you to work in the codespace without needing to reauthenticate during a typical working day, but reduces the chance that you

will leave a connection open when you stop using the codespace.

The token's scope will vary depending on the access you have to the repository where the codespace was created:

- **If you have write access to the repository**: The token will be scoped for read/write access to the repository.
- **If you only have read access to the repository**: The token will only allow the code to be cloned from the source repository. If you make a commit in the codespace, or push a new branch, GitHub Codespaces automatically creates a fork of the repository, or links the codespace to an existing fork if you already have one for the upstream repository. The token is updated to have read and write access to the fork. For more information, see "Using source control in your codespace."
- **If you've authorized your codespace to access other repositories**: The token will be scoped for read or read/write access to the source repository and to any other repositories to which you've authorized access. For more information, see "Managing access to other repositories within your codespace."

Codespace connections

You can connect to your codespace using the TLS encrypted tunnel provided by the GitHub Codespaces service. Only the creator of a codespace can connect to a codespace. Connections are authenticated with GitHub Enterprise Cloud.

If you need to allow external access to services running on a codespace, you can enable port forwarding for private or public access.

Port forwarding

If you need to connect to a service (such as a development web server) running within your codespace, you can configure port forwarding to make the service available on the internet.

Organization owners can restrict the ability to make forward ports available publicly or within the organization. For more information, see "Restricting the visibility of forwarded ports."

Privately forwarded ports: Are accessible on the internet, but only the codespace creator can access them, after authenticating to GitHub Enterprise Cloud.

Publicly forwarded ports within your organization: Are accessible on the internet, but only to members of the same organization as the codespace, after authenticating to GitHub Enterprise Cloud.

Publicly forwarded ports: Are accessible on the internet, and anyone on the internet can access them. No authentication is needed to access public forwarded ports.

All forwarded ports are private by default, which means that you will need to authenticate before you can access the port. Access to a codespace's private forwarded ports is controlled by authentication cookies with a 3-hour expiry period. When the cookie expires, you will need to reauthenticate.

A public forwarded port will automatically revert back to private when you remove and re-add the port, or if you restart the codespace.

You can use the "Ports" panel to configure a port for public or private access, and can stop port forwarding when it's no longer required. For more information, see "Forwarding ports in your codespace."

Good security practices for your codespaces

Codespaces are designed to be security hardened by default. To help maintain this posture, we recommend that you follow good security practices during your development procedures:

- As with any development tool, remember that you should only open and work within repositories you know and trust.
- Before you add new dependencies to the codespace, check whether they are well-maintained, and if they release updates to fix any security vulnerabilities found in their code.

Using secrets to access sensitive information

Always use secrets when you want to use sensitive information (such as access tokens) in a codespace. You can access your secrets as environment variables in the codespace, including from the terminal. For example, you can launch a terminal within your codespace and use `echo $SECRET_NAME` to see the value of a secret.

The secret values are copied to environment variables whenever the codespace is resumed or created and are also synced when they are changed.

Secrets are not copied into the environment if you don't have write access to the codespace's repository.

For more information on secrets, see:

- "Managing secrets for your codespaces"
- "Managing secrets for your repository and organization for GitHub Codespaces"

Working with other people's contributions and repositories

When you create a codespace from a PR branch from a fork, the token in the codespace will vary depending on whether the repository is public or private:

- For a private repository, the codespace is granted access to both the fork and parent.
- For a public repository, the codespace will only have access to the fork and opening PRs on the parent.

We also further protect you in these scenarios by not injecting any of your codespace secrets into the environment. For more information, see "Managing secrets for your codespaces."

> **Note:** The scope of the token in the codespace can change if you create a codespace from a fork to which you only have read access, then make a commit or push a new branch in the codespace. In this situation, as with any other repository, GitHub Codespaces automatically creates a new fork, or links your codespace to an existing fork owned by your account, and updates the token to have read and write access to the newly linked fork. For more information, see "Using source control in your codespace."
>
> When GitHub Codespaces links your codespace to an existing fork, this existing fork can be either a fork of the fork from which you created a codespace, or your own fork of the shared upstream repository.

Additional good practices

There are some additional good practices and risks that you should be aware of when using GitHub Codespaces.

Understanding a repository's devcontainer.json file

When you create a codespace, if a `devcontainer.json` file is found for your repository, it is parsed and used to configure your codespace. The `devcontainer.json` file can contain powerful features, such as installing third-party extensions and running arbitrary code supplied in a `postCreateCommand`.

For more information, see "Introduction to dev containers."

Granting access through features

Certain development features can potentially add risk to your environment. For example, commit signing, secrets injected into environment variables, authenticated registry access, and packages access can all present potential security issues. We recommend that you only grant access to those who need it and adopt a policy of being as restrictive as possible.

Using extensions

Any additional VS Code extensions that you've installed can potentially introduce more risk. To help mitigate this risk, ensure that the you only install trusted extensions, and that they are always kept up to date.

Using Settings Sync

VS Code's Settings Sync can allow potentially malicious content to transfer across devices. By default, Settings Sync is disabled for codespaces opened in the browser. If you're creating a codespace for a repository whose contents you do not trust, you should open the codespace in the browser and leave Settings Sync turned off.

If you have enabled Settings Sync in your user preferences, and want to allow changes to your settings to sync from your codespaces to other instances of VS Code, we recommend you add a selected list of trusted repositories, rather than trusting all repositories. When you create codespaces from trusted repositories, changes you make to your settings in the codespaces are synced to your cached settings in the cloud, from which they can transfer to your devices. For more information about managing Settings Sync, see "Personalizing GitHub Codespaces for your account."