Understanding the codespace lifecycle

In this article

You can develop in a GitHub Codespaces environment and maintain your data throughout the entire codespace lifecycle.

This article explains the stages in the life of a codespace, from creation to deletion. If you have read the "Quickstart for GitHub Codespaces" article and you now want to start using GitHub Codespaces for your own work, see the articles under "Developing in a codespace."

About the lifecycle of a codespace

The lifecycle of a codespace begins when you create a codespace and ends when you delete it. You can disconnect and reconnect to an active codespace without affecting its running processes. You may stop and restart a codespace without losing changes that you have made to your project.

Creating a codespace

When you want to work on a project, you can choose to create a new codespace or open an existing codespace. You might want to create a new codespace from a branch of your repository each time you develop in GitHub Codespaces or keep a long-running codespace for a feature. If you're starting a new project, you might want to create a codespace from a template and publish to a repository on GitHub later. For more information, see "Creating a codespace for a repository" and "Creating a codespace from a template."

There are limits to the number of codespaces you can create, and the number of codespaces you can run at the same time. These limits vary based on a number of factors. If you reach the maximum number of codespaces and try to create another, a message is displayed telling you that you must remove an existing codespace before you can create a new one. Similarly, if you reach the maximum number of active codespaces and you try to start another, you are prompted to stop one of your active codespaces.

If you choose to create a new codespace each time you work on a project, you should regularly push your changes so that any new commits are on GitHub. If you choose to use a long-running codespace for your project, you should pull from your repository's default branch each time you start working in your codespace so that your environment

has the latest commits. This workflow is very similar to if you were working with a project on your local machine.

To speed up codespace creation, repository administrators can enable GitHub Codespaces prebuilds for a repository. For more information, see "About GitHub Codespaces prebuilds."

Saving changes in a codespace

When you connect to a codespace through the web, auto-save is enabled automatically for the web editor and configured to save changes after a delay. When you connect to a codespace through Visual Studio Code running on your desktop, you must enable auto-save. For more information, see Save/Auto Save in the Visual Studio Code documentation.

Your work will be saved on a virtual machine in the cloud. You can close and stop a codespace and return to the saved work later. If you have unsaved changes, your editor will prompt you to save them before exiting. However, if your codespace is deleted, then your work will be deleted too. To persist your work, you will need to commit your changes and push them to your remote repository, or publish your work to a new remote repository if you created your codespace from a template. For more information, see "Using source control in your codespace."

Timeouts for GitHub Codespaces

If you leave your codespace running without interaction, or if you exit your codespace without explicitly stopping it, the codespace will timeout after a period of inactivity and stop running. By default, a codespace will timeout after 30 minutes of inactivity, but you can customize the duration of the timeout period for new codespaces that you create. For more information about setting the default timeout period for your codespaces, see "Setting your timeout period for GitHub Codespaces." For more information about stopping a codespace, see "Stopping a codespace."

When a codespace times out, your data is preserved from the last time your changes were saved. For more information, see "Saving changes in a codespace."

Rebuilding a codespace

You can rebuild your codespace to implement changes to your dev container configuration. For most uses, you can create a new codespace as an alternative to rebuilding a codespace. By default, when you rebuild your codespace, GitHub Codespaces will reuse images from your cache to speed up the rebuild process. Alternatively, you can perform a full rebuild, which clears your cache and rebuilds the container with fresh images.

> **Note:** When you rebuild the container in a codespace, changes you have made outside the `/workspaces` directory are cleared. Changes you have made inside the `/workspaces` directory, which includes the clone of the repository or template from which you created the codespace, are preserved over a rebuild. For more information, see "Deep dive into GitHub Codespaces."

For more information, see "Introduction to dev containers" and "Rebuilding the container in a codespace."

Stopping a codespace

You can stop a codespace at any time. When you stop a codespace, any running

processes are stopped. Any saved changes in your codespace will still be available when you next start it. The terminal history is preserved, but the visible contents of the terminal window are not preserved between codespace sessions.

If you do not explicitly stop a codespace, it will continue to run until it times out from inactivity. Closing a codespace does not stop the codespace. For example, if you're using a codespace in the VS Code web client and you close the browser tab, the codespace remains running on the remote machine. For information about timeouts, see "Understanding the codespace lifecycle."

Only running codespaces incur CPU charges. A stopped codespace incurs only storage costs.

You may want to stop and restart a codespace to apply changes to it. For example, if you change the machine type used for your codespace, you will need to stop and restart it for the change to take effect. You can also stop your codespace and choose to restart or delete it if you encounter an error or something unexpected. For more information, see "Stopping and starting a codespace."

Deleting a codespace

You can create a codespace for a particular task and then safely delete the codespace after you push your changes to a remote branch.

If you try to delete a codespace with unpushed git commits, your editor will notify you that you have changes that have not been pushed to a remote branch. You can push any desired changes and then delete your codespace, or continue to delete your codespace and any uncommitted changes. You can also export your code to a new branch without creating a new codespace. For more information, see "Exporting changes to a branch."

Codespaces that have been stopped and remain inactive for a specified period of time will be deleted automatically. By default, inactive codespaces are deleted after 30 days, but you can customize your codespace retention period. For more information, see "Configuring automatic deletion of your codespaces."

If you create a codespace, it will continue to accrue storage charges until it is deleted, irrespective of whether it is active or stopped. For more information, see "About billing for GitHub Codespaces." Deleting a codespace does not reduce the current billable amount for GitHub Codespaces, which accumulates during each monthly billing cycle. For more information, see "Viewing your GitHub Codespaces usage."

For more information on deleting a codespace, see "Deleting a codespace."

Losing the connection while using GitHub Codespaces

GitHub Codespaces is a cloud-based development environment and requires an internet connection. If you lose connection to the internet while working in a codespace, you will not be able to access your codespace. However, any uncommitted changes will be saved. When you have access to an internet connection again, you can connect to your codespace in the exact same state that it was left in. If you have an unstable internet connection, you should commit and push your changes often.

If you know that you will often be working offline, you can use your `devcontainer.json` file with the "Dev Containers" extension for VS Code to build and attach to a local development container for your repository. For more information, see Developing inside a container in the Visual Studio Code documentation.