Using GitHub Codespaces with GitHub CLI

In this article

You can work with GitHub Codespaces directly from your command line by using `gh`, the GitHub command line interface.

About GitHub CLI

GitHub CLI is an open source tool for using GitHub from your computer's command line. When you're working from the command line, you can use the GitHub CLI to save time and avoid switching context. For more information, see "About GitHub CLI."

You can work with GitHub Codespaces in the GitHub CLI to:

- List all of your codespaces
- Create a new codespace
- View details of a codespace
- Stop a codespace
- Delete a codespace
- Rename a codespace
- Rebuild a codespace
- SSH into a codespace
- Open a codespace in Visual Studio Code
- Open a codespace in JupyterLab
- Copy a file to/from a codespace
- Modify ports in a codespace
- Access codespace logs
- Access remote resources
- Change the machine type of a codespace

Installing GitHub CLI

For installation instructions for GitHub CLI, see the GitHub CLI repository.

Using GitHub CLI

If you have not already done so, run `gh auth login` to authenticate with your GitHub account.

To use `gh` to work with GitHub Codespaces, type `gh codespace SUBCOMMAND` or its alias `gh cs SUBCOMMAND`.

As an example of a series of commands you might use to work with GitHub Codespaces, you could:

- List your current codespaces, to check whether you have a codespace for a particular repository:
  `gh codespace list`
- Create a new codespace for the required repository branch:
  `gh codespace create -r github/docs -b main`
- SSH into the new codespace:
  `gh codespace ssh -c octocat-literate-space-parakeet-7gwrqp9q9jcx4vq`
- Forward a port to your local machine:
  `gh codespace ports forward 8000:8000 -c octocat-literate-space-parakeet-7gwrqp9q9jcx4vq`

`gh` commands for GitHub Codespaces

The sections below give example commands for each of the available operations.

For a complete reference of `gh` commands for GitHub Codespaces, including details of all available options for each command, see the GitHub CLI online help for "gh codespace." Alternatively, on the command line, use `gh codespace --help` for general help or `gh codespace SUBCOMMAND --help` for help with a specific subcommand.

> **Note**: The `-c CODESPACE_NAME` flag, used with many commands, is optional. If you omit it a list of codespaces is displayed for you to choose from.

List all of your codespaces

```
gh codespace list
```

The list includes the unique name of each codespace, which you can use in other `gh codespace` commands.

An asterisk at the end of the branch name for a codespace indicates that there are uncommitted or unpushed changes in that codespace.

Create a new codespace

```
gh codespace create -r OWNER/REPO_NAME [-b BRANCH]
```

For more information, see "Creating a codespace for a repository."

View details of a codespace

```
gh codespace view
```

After running this command you are prompted to choose one of your existing codespaces. The following information is then displayed:

- Name of the codespace
- State (for example, "Available" or "Shutdown")

- Repository
- Git status
- Path to the dev container configuration file used to create the codespace
- Machine type
- Idle timeout
- Date and time the codespace was created
- Retention period

For more information, see the GitHub CLI reference.

Stop a codespace

```
gh codespace stop -c CODESPACE-NAME
```

For more information, see "Deep dive into GitHub Codespaces."

Delete a codespace

```
gh codespace delete -c CODESPACE-NAME
```

For more information, see "Deleting a codespace."

Rename a codespace

```
gh codespace edit -c CODESPACE-NAME -d DISPLAY-NAME
```

For more information, see "Renaming a codespace."

Rebuild a codespace

```
gh codespace rebuild
```

To perform a full rebuild, add `--full` at the end of this command. For more information, see "Rebuilding the container in a codespace."

When you use this command to rebuild a codespace, it uses the `devcontainer.json` file that is currently saved in the codespace's system. This happens regardless of whether or not the current state of the file has been saved in source control. For more information, see "Introduction to dev containers."

SSH into a codespace

To run commands on the remote codespace machine, from your terminal, you can SSH into the codespace.

```
gh codespace ssh -c CODESPACE-NAME
```

Note: The codespace you connect to must be running an SSH server. The default container image includes an SSH server, which is started automatically. If your codespaces are not created from the default image, you can install and start an SSH server by adding the following to the `features` object in your `devcontainer.json` file.

```
"features": {
```

```
    // ...
    "ghcr.io/devcontainers/features/sshd:1": {
        "version": "latest"
    },
    // ...
}
```

For more information about the `devcontainer.json` file and the default container image, see "Introduction to dev containers."

GitHub Codespaces creates a local SSH key automatically to provide a seamless authentication experience. For more information on connecting with SSH, see `gh codespace ssh`.

Open a codespace in Visual Studio Code

```
gh codespace code -c CODESPACE-NAME
```

You must have VS Code installed on your local machine. For more information, see "Using GitHub Codespaces in Visual Studio Code."

Open a codespace in JupyterLab

```
gh codespace jupyter -c CODESPACE-NAME
```

The JupyterLab application must be installed in the codespace you are opening. The default container image includes JupyterLab, so codespaces created from the default image will always have JupyterLab installed. For more information about the default image, see "Introduction to dev containers" and the `devcontainers/images` repository. If you're not using the default image in your dev container configuration, you can install JupyterLab by adding the `ghcr.io/devcontainers/features/python` feature to your `devcontainer.json` file. You should include the option `"installJupyterlab": true`. For more information, see the README for the `python` feature, in the `devcontainers/features` repository.

Copy a file to/from a codespace

```
gh codespace cp [-r] SOURCE(S) DESTINATION
```

Use the prefix `remote:` on a file or directory name to indicate that it's on the codespace. As with the UNIX `cp` command, the first argument specifies the source and the last specifies the destination. If the destination is a directory, you can specify multiple sources. Use the `-r` (recursive) flag if any of the sources is a directory.

The location of files and directories on the codespace is relative to the home directory of the remote user.

Examples

- Copy a file from the local machine to the `$HOME` directory of a codespace:

  `gh codespace cp myfile.txt remote:`

- Copy a file to the directory in which a repository is checked out in a codespace:

  `gh codespace cp myfile.txt remote:/workspaces/REPOSITORY-NAME`

- Copy a file from a codespace to the current directory on the local machine:

  `gh codespace cp remote:myfile.txt .`

- Copy three local files to the `$HOME/temp` directory of a codespace:

  `gh codespace cp a1.txt a2.txt a3.txt remote:temp`

- Copy three files from a codespace to the current working directory on the local machine:

  `gh codespace cp remote:a1.txt remote:a2.txt remote:a3.txt .`

- Copy a local directory into the `$HOME` directory of a codespace:

  `gh codespace cp -r mydir remote:`

- Copy a directory from a codespace to the local machine, changing the directory name:

  `gh codespace cp -r remote:mydir mydir-localcopy`

For more information about the `gh codespace cp` command, including additional flags you can use, see the GitHub CLI manual.

Modify ports in a codespace

You can forward a port on a codespace to a local port. The port remains forwarded as long as the process is running. To stop forwarding the port, press `Control` + `C`.

```
gh codespace ports forward CODESPACE-PORT_NAME:LOCAL-PORT-NAME -c CODESPACE-NAME
```

To see details of forwarded ports enter `gh codespace ports` and then choose a codespace.

You can set the visibility of a forwarded port. There are three visibility settings:

- `private` - Visible only to you. This is the default setting when you forward a port.
- `org` - Visible to members of the organization that owns the repository.
- `public` - Visible to anyone who knows the URL and port number.

```
gh codespace ports visibility CODESPACE-PORT:private|org|public -c CODESPACE-NAME
```

You can set the visibility for multiple ports with one command. For example:

```
gh codespace ports visibility 80:private 3000:public 3306:org -c CODESPACE-NAME
```

For more information, see "Forwarding ports in your codespace."

Access codespace logs

You can see the creation log for a codespace. After entering this command you will be asked to enter the passphrase for your SSH key.

```
gh codespace logs -c CODESPACE-NAME
```

For more information about the creation log, see "GitHub Codespaces logs."

Access remote resources

You can use the GitHub CLI extension to create a bridge between a codespace and your local machine, so that the codespace can access any remote resource that is accessible from your machine. For more information on using the extension, see "Using GitHub CLI to access remote resources."

> **Note**: The GitHub CLI extension is currently in beta and subject to change.

Change the machine type of a codespace

```
gh codespace edit -m MACHINE-TYPE-NAME
```

For more information, see the "GitHub CLI" tab of "Changing the machine type for your codespace."