Create a repository

To put your project up on GitHub, you will need to create a repository for it to live in.

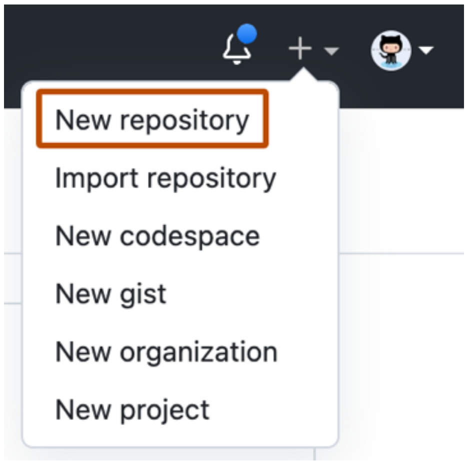
Create a repository

You can store a variety of projects in GitHub repositories, including open source projects. With open source projects, you can share code to make better, more reliable software. You can use repositories to collaborate with others and track your work. For more information, see "About repositories." To learn more about open source projects, visit OpenSource.org.

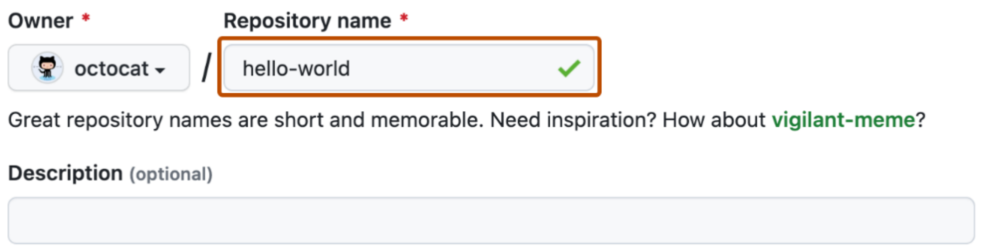
Notes:

- You can create public repositories for an open source project. When creating your public repository, make sure to include a license file that determines how you want your project to be shared with others. For more information on open source, specifically how to create and grow an open source project, we've created Open Source Guides that will help you foster a healthy open source community by recommending best practices for creating and maintaining repositories for your open source project.
- You can also take a free GitHub Skills course on maintaining open source communities.
- You can also add community health files to your repositories, to set guidelines on how to contribute, keep your repositories safe, and much more. For more information, see "Creating a default community health file."

1. In the upper-right corner of any page, use the drop-down menu, and select **New repository**.



2. Type a short, memorable name for your repository. For example, "hello-world".



Great repository names are short and memorable. Need inspiration? How about vigilant-meme?

3. Optionally, add a description of your repository. For example, "My first repository on GitHub."
4. Choose a repository visibility. For more information, see "About repositories."
5. Select **Initialize this repository with a README**.
6. Click **Create repository**.

Congratulations! You've successfully created your first repository, and initialized it with a *README* file.

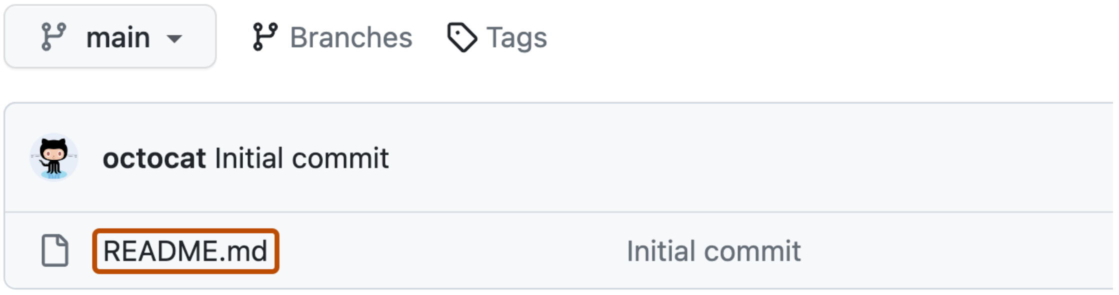
Commit your first change

A commit is like a snapshot of all the files in your project at a particular point in time.

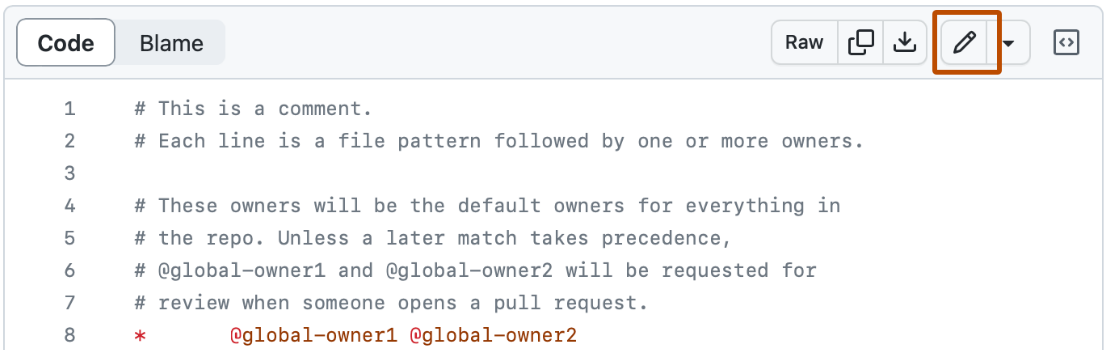
When you created your new repository, you initialized it with a *README* file. *README* files are a great place to describe your project in more detail, or add some documentation such as how to install or use your project. The contents of your *README* file are automatically shown on the front page of your repository.

Let's commit a change to the README file.

1. In your repository's list of files, select **README.md**.

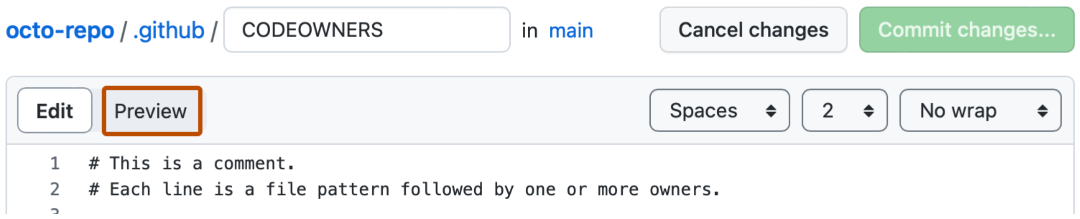


2. In the upper right corner of the file view, click to open the file editor.



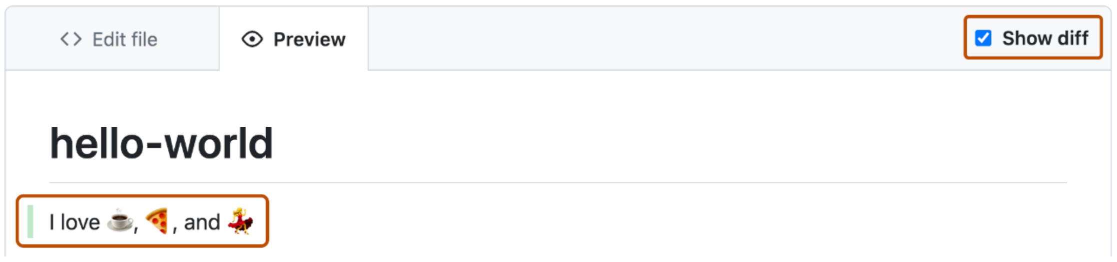
```
# This is a comment.
# Each line is a file pattern followed by one or more owners.

# These owners will be the default owners for everything in
# the repo. Unless a later match takes precedence,
# @global-owner1 and @global-owner2 will be requested for
# review when someone opens a pull request.
*       @global-owner1 @global-owner2
```

3. In the text box, type some information about yourself.
4. Above the new content, click **Preview**.

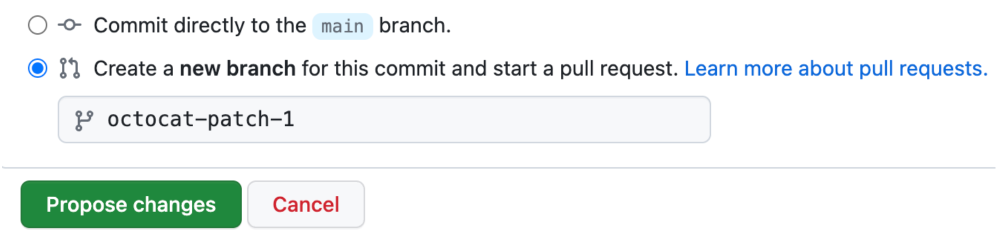


```
# This is a comment.
# Each line is a file pattern followed by one or more owners.
```

5. Review the changes you made to the file. If you select **Show diff**, you will see the new content in green.



6. Click **Commit changes...**
7. In the "Commit message" field, type a short, meaningful commit message that describes the change you made to the file. You can attribute the commit to more than one author in the commit message. For more information, see "Creating a commit with multiple authors."
8. Below the commit message fields, decide whether to add your commit to the current branch or to a new branch. If your current branch is the default branch, you should choose to create a new branch for your commit and then create a pull request. For more information, see "Creating a pull request."


Commit directly to the main branch.
Create a **new branch** for this commit and start a pull request. Learn more about pull requests.
octocat-patch-1
Propose changes
Cancel

9. Click **Commit changes** or **Propose changes**.

Cloning a repository

When you create a repository on GitHub.com, it exists as a remote repository. You can clone your repository to create a local copy on your computer and sync between the two locations.

About cloning a repository

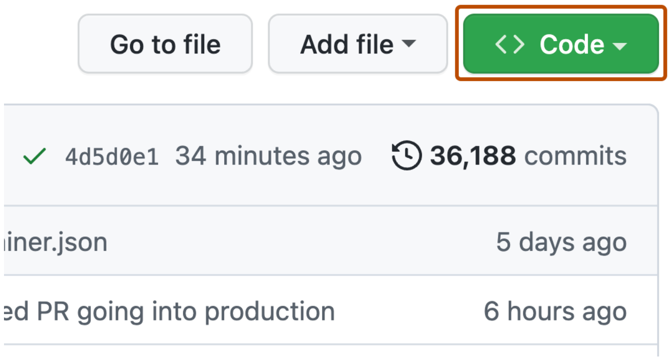
You can clone a repository from GitHub.com to your local computer, or to a codespace, to make it easier to fix merge conflicts, add or remove files, and push larger commits. When you clone a repository, you copy the repository from GitHub.com to your local machine, or to a remote virtual machine when you create a codespace. For more information about cloning to a codespace, see "Creating a codespace for a repository."

Cloning a repository pulls down a full copy of all the repository data that GitHub.com has at that point in time, including all versions of every file and folder for the project. You can push your changes to the remote repository on GitHub.com, or pull other people's changes from GitHub.com. For more information, see "Using Git".

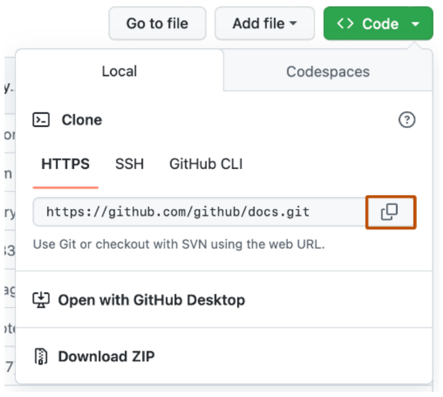
You can clone your existing repository or clone another person's existing repository to contribute to a project.

Cloning a repository

1. On GitHub.com, navigate to the main page of the repository.
2. Above the list of files, click **Code**.



3. Copy the URL for the repository.
   - To clone the repository using HTTPS, under "HTTPS", click .
   - To clone the repository using an SSH key, including a certificate issued by your organization's SSH certificate authority, click **SSH**, then click .
   - To clone a repository using GitHub CLI, click **GitHub CLI**, then click .



```
https://github.com/github/docs.git
```

Use Git or checkout with SVN using the web URL.

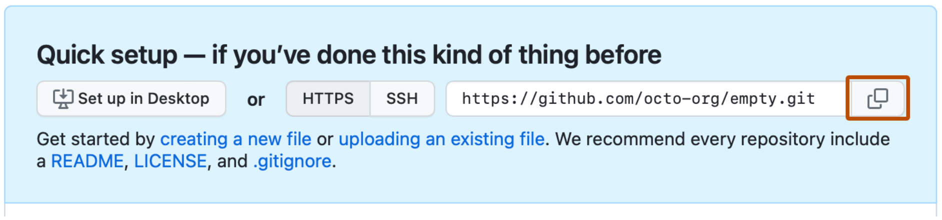
4. Open Git Bash.
5. Change the current working directory to the location where you want the cloned directory.
6. Type `git clone`, and then paste the URL you copied earlier.
   ```
   git clone https://github.com/YOUR-USERNAME/YOUR-REPOSITORY
   ```
7. Press **Enter** to create your local clone.

```
8. $ git clone https://github.com/YOUR-USERNAME/YOUR-REPOSITORY
9. > Cloning into `Spoon-Knife`...
10.> remote: Counting objects: 10, done.
11.> remote: Compressing objects: 100% (8/8), done.
12.> remove: Total 10 (delta 1), reused 10 (delta 1)
   > Unpacking objects: 100% (10/10), done.
```

Cloning an empty repository

An empty repository contains no files. It's often made if you don't initialize the repository with a README when creating it.

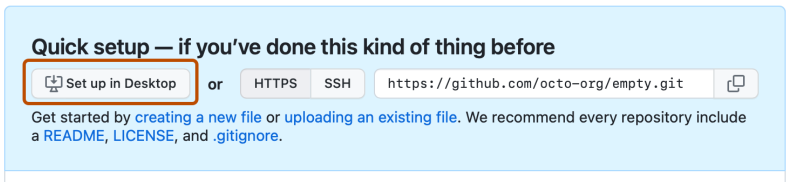
1. On GitHub.com, navigate to the main page of the repository.
2. To clone your repository using the command line using HTTPS, under "Quick setup", click . To clone the repository using an SSH key, including a certificate issued by your organization's SSH certificate authority, click **SSH**, then click .

   ```
   https://github.com/octo-org/empty.git
   ```

   Get started by creating a new file or uploading an existing file. We recommend every repository include a README, LICENSE, and .gitignore.

   Alternatively, to clone your repository in Desktop, click **Set up in Desktop** and follow the prompts to complete the clone.

   ```
   https://github.com/octo-org/empty.git
   ```

   Get started by creating a new file or uploading an existing file. We recommend every repository include a README, LICENSE, and .gitignore.

3. Open Git Bash.
4. Change the current working directory to the location where you want the cloned directory.
5. Type `git clone`, and then paste the URL you copied earlier.
   ```
   git clone https://github.com/YOUR-USERNAME/YOUR-REPOSITORY
   ```
6. Press **Enter** to create your local clone.

```
7. $ git clone https://github.com/YOUR-USERNAME/YOUR-REPOSITORY
8. > Cloning into `Spoon-Knife`...
9. > remote: Counting objects: 10, done.
10.> remote: Compressing objects: 100% (8/8), done.
11.> remove: Total 10 (delta 1), reused 10 (delta 1)
   > Unpacking objects: 100% (10/10), done.
```

Troubleshooting cloning errors

When cloning a repository it's possible that you might encounter some errors.

If you're unable to clone a repository, check that:

- You can connect using HTTPS. For more information, see "Troubleshooting cloning errors."
- You have permission to access the repository you want to clone. For more information, see "Troubleshooting cloning errors."
- The default branch you want to clone still exists. For more information, see "Troubleshooting cloning errors."

Deleting a repository

You can delete any repository or fork if you're either an organization owner or have admin permissions for the repository or fork. Deleting a forked repository does not delete the upstream repository.

Only members with owner privileges for an organization or admin privileges for a repository can delete an organization repository. If **Allow members to delete or transfer repositories for this organization** has been disabled, only organization owners can delete organization repositories. For more information, see "Repository roles for an organization."

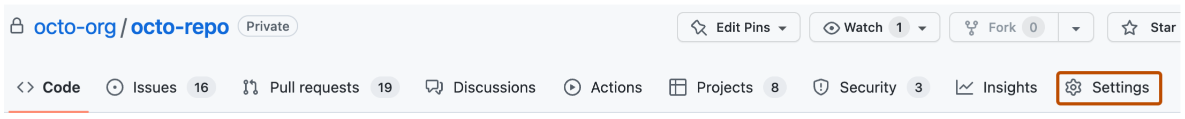
Deleting a public repository will not delete any forks of the repository.

Warnings:

- Deleting a repository will **permanently** delete release attachments and team permissions. This action **cannot** be undone.
- Deleting a private repository will delete all forks of the repository.

Some deleted repositories can be restored within 90 days of deletion. For more information, see "Restoring a deleted repository."

1. On GitHub.com, navigate to the main page of the repository.
2. Under your repository name, click **Settings**. If you cannot see the "Settings" tab, select the dropdown menu, then click **Settings**.



3. In the "Danger Zone" section, click **Delete this repository**.
4. **Read the warnings**.
5. To verify that you're deleting the correct repository, in the text box, type the name of the repository you want to delete.
6. Click **I understand the consequences, delete this repository**.

Restoring a deleted repository

You can restore some deleted repositories to recover their contents.

Who can use this feature

Anyone can restore deleted repositories that were owned by their own personal account. Organization owners can restore deleted repositories that were owned by the organization.

About repository restoration

A deleted repository can be restored within 90 days, unless the repository was part of a fork network that is not currently empty. A fork network consists of a parent repository, the repository's forks, and forks of the repository's forks. If your repository was part of a fork network, it cannot be restored unless every other repository in the network is deleted or has been detached from the network. For more information about forks, see "About forks."

If you want to restore a repository that was part of a fork network that is not currently empty, you can contact GitHub Support.

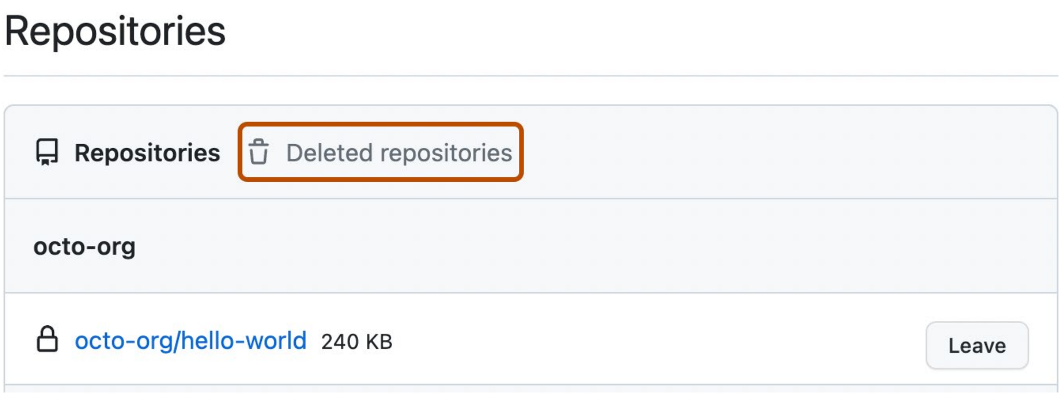
It can take up to an hour after a repository is deleted before that repository is available for restoration.

Restoring a repository will not restore release attachments or team permissions. Issues that are restored will not be labeled.

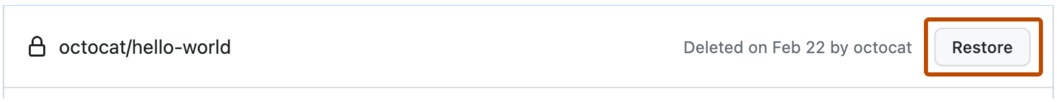
Restoring a deleted repository that was owned by a personal account

1. In the upper-right corner of any page, click your profile photo, then click **Settings**.

2. In the "Code planning, and automation" section of the sidebar, click **Repositories**.
3. Under "Repositories", click **Deleted repositories**.



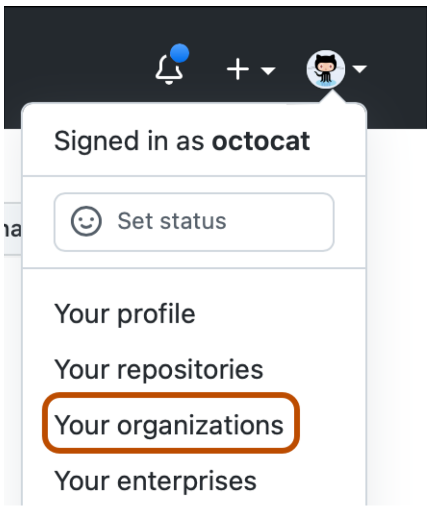
4. Next to the repository you want to restore, click **Restore**.



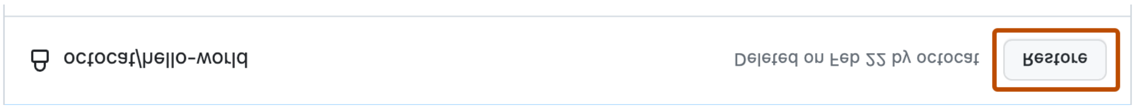
5. Read the warning, then click **I understand, restore this repository**.

Restoring a deleted repository that was owned by an organization

1. In the top right corner of GitHub.com, click your profile photo, then click **Your organizations**.



2. Next to the organization, click **Settings**.
3. In the left sidebar, click **Deleted repositories**.
4. Next to the repository you want to restore, click **Restore**.



5. Read the warning, then click **I understand, restore this repository**.

UPDATED: JULY 2023