# Create a repo

In this article

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

GitHub CLI   Web browser

Create a repository 🔗

You can store a variety of projects in GitHub Enterprise Server repositories, including innersource projects. With innersource, you can share code to make better, more reliable software. For more information on innersource, see GitHub's white paper "An introduction to innersource."

1. In the upper-right corner of any page, use the + drop-down menu, and select **New repository**.
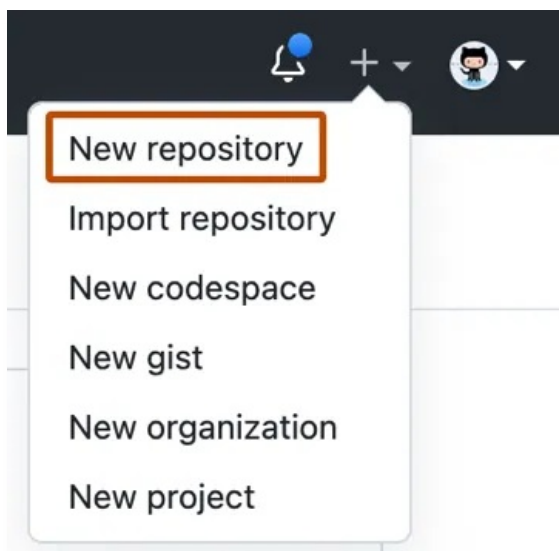
2. Type a short, memorable name for your repository. For example, "hello-world".

3. Optionally, add a description of your repository. For example, "My first repository on GitHub Enterprise Server."

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.

> To learn more about GitHub CLI, see "[About GitHub CLI](#)."

1. In the command line, navigate to the directory where you would like to create a local clone of your new project.

2. To create a repository for your project, use the `gh repo create` subcommand. When prompted, select **Create a new repository on GitHub from scratch** and enter the name of your new project. If you want your project to belong to an organization instead of to your personal account, specify the organization name and project name with `organization-name/project-name`.

3. Follow the interactive prompts. To clone the repository locally, confirm yes when asked if you would like to clone the remote project directory.

4. Alternatively, to skip the prompts supply the repository name and a visibility flag (`--public` , `--private` , or `--internal`). For example, `gh repo create project-name --public` . To clone the repository locally, pass the `--clone` flag. For more information about possible arguments, see the [GitHub CLI manual](#).

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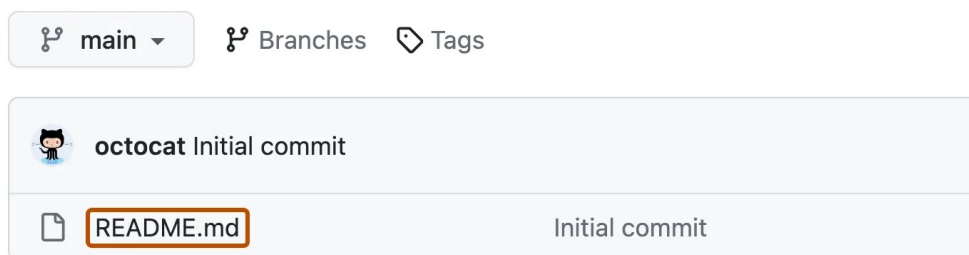
2. In the upper right corner of the file view, click 🖉 to open the file editor.

```
71 lines (61 sloc)   2.44 KB                    Raw   Blame   ✎  ▾   ⧉   🗑

    1   # For most projects, this workflow file will not need changing; you simply need
    2   # to commit it to your repository.
    3   #
```

③ In the text box, type some information about yourself.

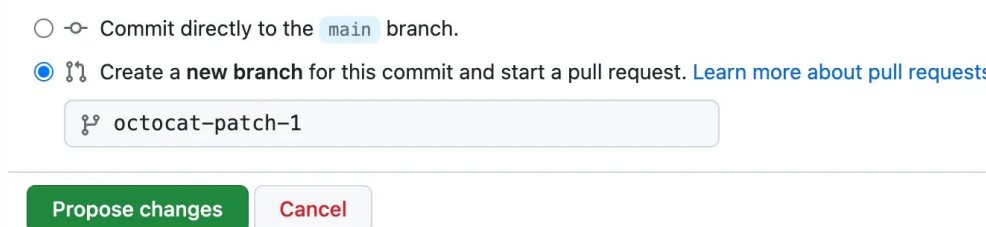④ Above the new content, click **Preview changes**.

octo-repo / .github / CODEOWNERS   in main

    <> Edit file   ⊙ Preview changes

    1   # This is a comment.
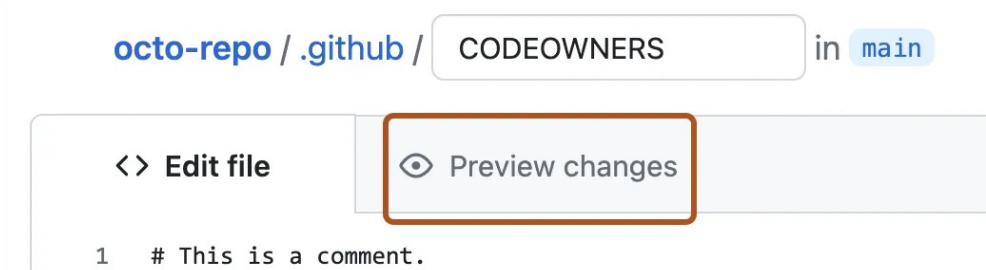
⑤ Review the changes you made to the file. If you select **Show diff**, you will see the
new content in green.

    <> Edit file   ⊙ Preview   ☑ Show diff

    hello-world

    I love ☕, 🍕, and 🐎
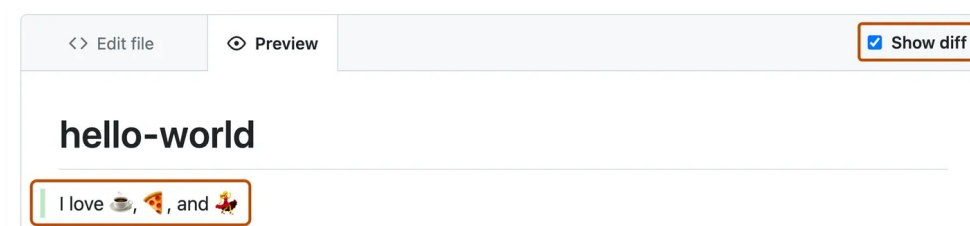
⑥ In the "Commit message" field, type a short, meaningful commit message that
describes the change you made to the file. You can attribute the commit to more
than one author in the commit message. For more information, see "Creating a
commit with multiple authors."

⑦ 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]

⑧ Click **Commit changes** or **Propose changes**.

Now that you have created a project, you can start committing changes.

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. Follow these steps to
add a *README* file.

1. In the command line, navigate to the root directory of your new project. (This directory was created when you ran the `gh repo create` command.)

2. Create a *README* file with some information about the project.

```
echo "info about this project" >> README.md
```

3. Enter `git status`. You will see that you have an untracked `README.md` file.

```
$ git status

Untracked files:
  (use "git add <file>..." to include in what will be committed)
  README.md

nothing added to commit but untracked files present (use "git add" to track)
```

4. Stage and commit the file.

```
git add README.md && git commit -m "Add README"
```

5. Push the changes to your branch.

```
git push --set-upstream origin HEAD
```

Next steps 🔗

You have now created a repository, including a *README* file, and created your first commit on your GitHub Enterprise Server instance.

- You can now clone a GitHub repository to create a local copy on your computer. From your local repository you can commit, and create a pull request to update the changes in the upstream repository. For more information, see "Cloning a repository" and "Set up Git."

- You can find interesting projects and repositories on GitHub and make changes to them by creating a fork of the repository. Forking a repository will allow you to make changes to another repository without affecting the original. For more information, see "Fork a repo."

- Each repository on GitHub is owned by a person or an organization. You can interact with the people, repositories, and organizations by connecting and following them on GitHub Enterprise Server. For more information, see "Be social."

- GitHub has a great support community where you can ask for help and talk to people from around the world. Join the conversation on GitHub Community.