What is GitHub?

In this unit, we review the following learning objectives:

- Brief overview of the GitHub Enterprise Platform
- How to create a repository
- Adding files to a repository
- How to search for repositories
- Introduction to gists and wikis

GitHub

GitHub is a cloud-based platform that uses Git, a distributed version control system, at its core. The GitHub platform simplifies the process of collaborating on projects and provides a website, command-line tools, and overall flow that allows developers and users to work together.

As we learned earlier, GitHub provides an AI powered developer platform to build, scale, and deliver secure software. Let's break down each one of the core pillars of the GitHub Enterprise platform, AI, Collaboration, Productivity, Security, and Scale.

AI

Generative AI is dramatically transforming software development as we speak. The GitHub Enterprise platform is enhancing collaboration through AI-powered pull requests and issues, productivity through Copilot, and security by automating security checks faster.

Collaboration

Collaboration is at the core of everything GitHub does. We know inefficient collaboration results in wasted time and money. We counteract that with a suite of seamless tools that allow collaboration to happen effortlessly.

Repositories, Issues, Pull Requests, and other tools help to enable developers, project managers, operation leaders, and others at the same company. It enables them to work faster together, cut down approval times, and ship more quickly.

Productivity

Productivity is accelerated with automation that the GitHub Enterprise Platform provides. With built-in CI/CD (Continuous Integration and Continuous Delivery) tools directly integrated into the workflow, the platform gives users the ability to set tasks and forget them, taking care of routine administration and speeding up day-to-day work. This gives your developers more time to focus on what matters most, creating innovative solutions.

Security

GitHub focuses on integrating security directly into the development process from the start. GitHub Enterprise platform includes native, first-party security features that minimize security risk with a built-in security solution. Plus, your code remains private within your organization. At the same time, you're able to take advantage of security overview and Dependabot.

GitHub has continued to make investments to ensure that our features are enterprise-ready. Microsoft and highly regulated industries trust GitHub, and we meet global compliance requirements.

Scale

GitHub is the largest developer community of its kind with real-time data on over 100M+ developers, 330M+ repositories, and countless deployments. We've been able to understand the shifting needs of developers and make changes to our product to match.

This has translated into an incredible scale that is unmatched and unparalleled by any other company on the planet. Everyday we're gaining more insights from this impressive community and evolving the platform to meet their needs.

In essence, the GitHub Enterprise Platform focuses on the developer experience. It has the scale to provide industry-changing insights, collaboration capabilities for transformative efficiency, the tools for increased productivity, security at every step, and AI to power it all to new heights in a single, integrated platform.

Now let's get into the backbone of GitHub, repositories.

Introduction to repositories

Let's first review:

- What is a repository?
- How to create a repository
- Adding files to a repository
- How to search for repositories
- Introduction to gists, wikis, and GitHub pages

What is a repository?

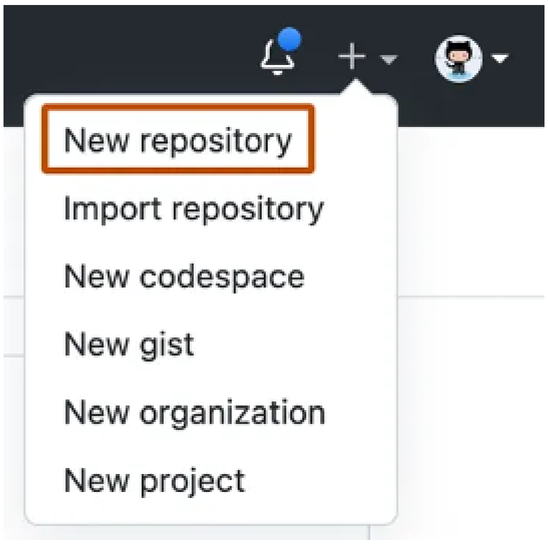
A repository contains all of your project's files and each file's revision history. It's one of the essential parts that helps you collaborate with people. You can use repositories to manage your work, track changes, store revision history, and work with others. Before we dive too deep, let's first start with how to create a repository.

How to create a repository

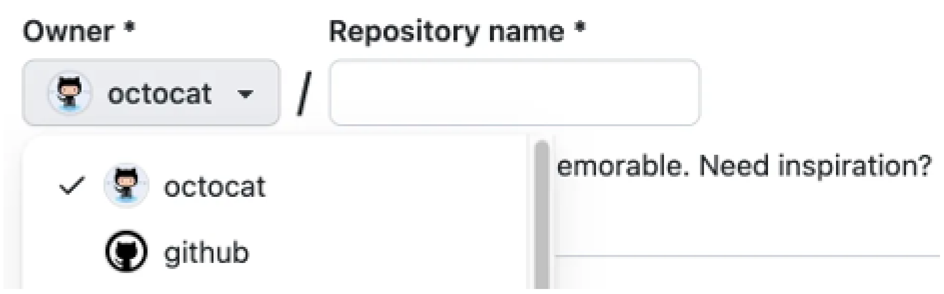
You can create a new repository on your personal account or any organization where you have sufficient permissions.

Let's tackle creating a repository from github.com.

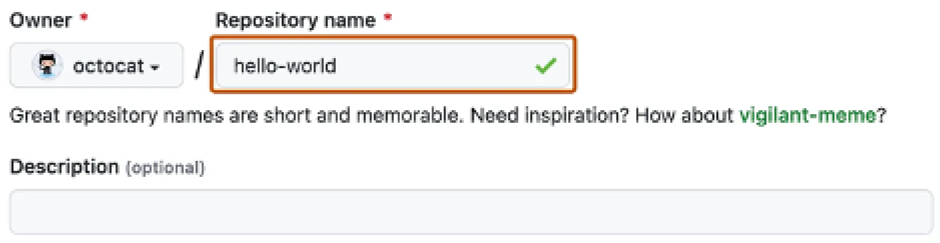
1. In the upper-right corner of any page, use the drop-down menu, and select **New repository**.



2. Use the **Owner** drop-down menu to select the account you want to own the repository.



3. Type a name for your repository, and an optional description.



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

4. Choose a repository visibility.
    - **Public repositories** are accessible to everyone on the internet.
    - **Private repositories** are only accessible to you, people you explicitly share access with, and, for organization repositories, certain organization members.
5. Select **Create repository** and congratulations! You just created a repository!

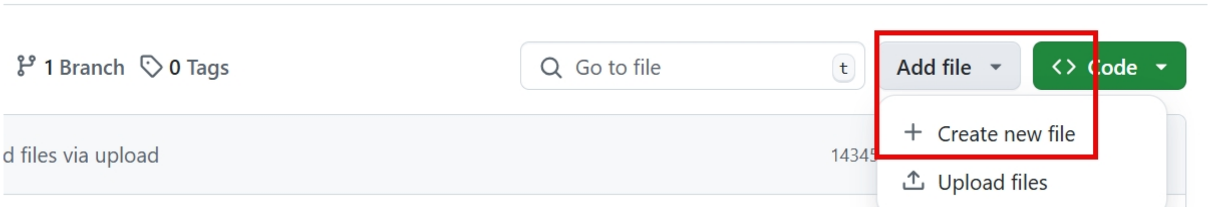
Next up, let's review how to add files to your repository.

How to add a file to your repository

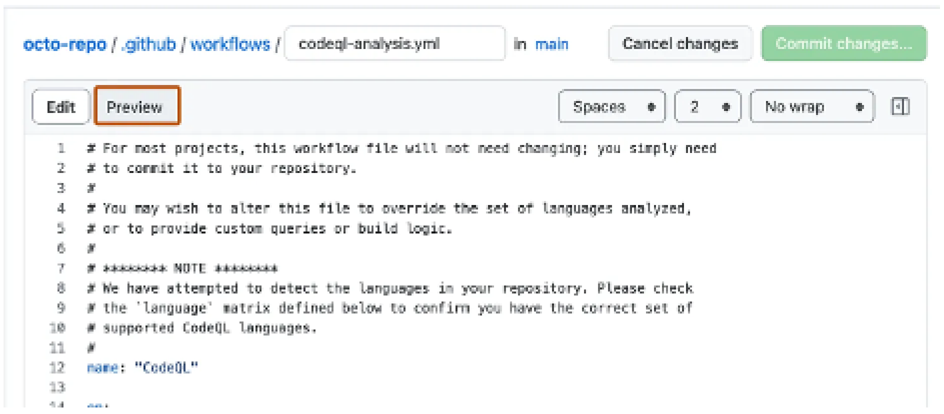
Files in GitHub can do a handful of things, but the main purpose of files is to store data and information about your project. It's worth knowing in order to add a file to a repository that you must first have minimum **Write** access within the repository you want to add a file.

Let's review how to add a file to your repository.

1. On GitHub.com, navigate to the main page of the repository.
2. In your repository, browse to the folder where you want to create a file by selecting the **creating a new file** link or **uploading an existing file**.
3. Once added, above the list of files select the **Add file ∇** drop-down menu. Then select **Create new file**.



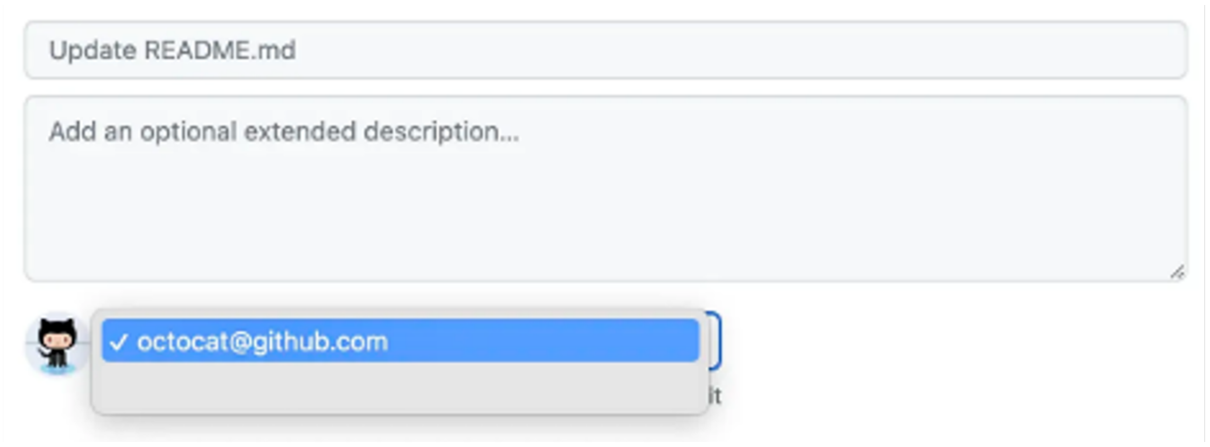
4. In the file name field, type the name and extension for the file. To create subdirectories, type the / directory separator.
5. In the file contents text box, type **content** for the file.
6. To review the new content, above the file contents, select **Preview**.



```
# For most projects, this workflow file will not need changing; you simply need
# to commit it to your repository.
#
# You may wish to alter this file to override the set of languages analyzed,
# or to provide custom queries or build logic.
#
# ******** NOTE ********
# We have attempted to detect the languages in your repository. Please check
# the 'language' matrix defined below to confirm you have the correct set of
# supported CodeQL languages.
#
name: "CodeQL"

```

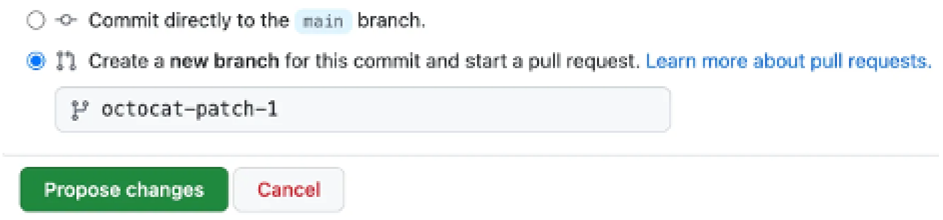
7. Select **Commit changes**.
8. In the **Commit message** field, type a short and meaningful commit message that describes the change you made to the file. You can attribute the commit to more than one author in the commit message.
9. If you have more than one email address associated with your account on GitHub.com, select the email address drop-down menu. Then select the email address to use as the Git author email address. Only verified email addresses appear in this drop-down menu. If you enabled email address privacy, then *[username]@users.noreply.github.com* is the default commit author email address.


Update README.md

Add an optional extended description...

✓ octocat@github.com

10. 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.


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

11. Select **Commit changes** or **Propose changes**.

Congratulations, you just created a new file in your repository! You have also created a new branch and made a commit.

Before we review branches and commits in the next unit, let’s quickly review gists, wikis, and GitHub pages because they're similar to repositories.

What are gists

Now that we have a good understanding of repositories, we can review gists. Similarly to repositories, gists are a simplified way to share code snippets with others.

Every gist is a Git repository, which you can fork and clone and be made either public or secret. Public gists are displayed publicly where people can browse new ones as they’re created. Public gists are also searchable. Conversely, secret gists aren't searchable, but they aren’t entirely private. If you send the URL of a secret gist to a friend, they'll be able to see it.

To learn more about gists, see the linked article in our Resources section at the end of this module titled *Creating Gists*.

What are wikis?

Every repository on GitHub.com comes equipped with a section for hosting documentation, called a wiki. You can use your repository's wiki to share long-form content about your project, such as how to use it, how you designed it, or its core principles. While a README file quickly tells what your project can do, you can use a wiki to provide additional documentation.

It’s worth a reminder that if your repository is private, only people who have at least read access to your repository will have access to your wik

Components of the GitHub flow

In this unit, we're reviewing the following components of the GitHub flow:

- Branches
- Commits
- Pull Requests
- The GitHub Flow

What are branches

In the last section, we created a new file and a new branch in your repositories.

Branches are an essential part to the GitHub experience because they're where we can make changes without affecting the entire project we're working on.

Your branch is a safe place to experiment with new features or fixes. If you make a mistake, you can revert your changes or push more changes to fix the mistake. Your changes won't update on the default branch until you merge your branch.

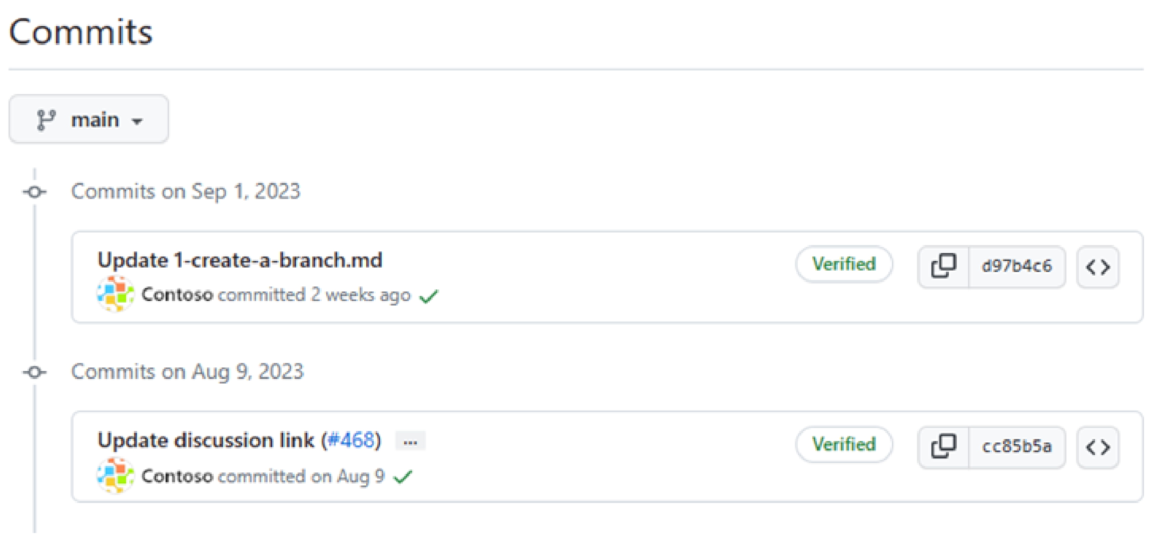
Note

Alternatively, you can create a new branch and check it out by using git in a terminal. The command would be git checkout -b newBranchName

What are commits

In the previous unit, you added a new file into the repository by pushing a commit. Let’s briefly review what commits are.

A **commit** is a change to one or more files on a branch. Every time a commit is created, it's assigned a unique ID and tracked along with the time and contributor. Commits provide a clear audit trail for anyone reviewing the history of a file or linked item, such as an issue or pull request.


Commits

main

Commits on Sep 1, 2023

Update 1-create-a-branch.md
Contoso committed 2 weeks ago ✓
Verified d97b4c6

Commits on Aug 9, 2023

Update discussion link (#468)
Contoso committed on Aug 9 ✓
Verified cc85b5a

Within a git repository, a file can exist in several valid states as it goes through the version control process. The primary states for a file in a Git repository are **Untracked** and **Tracked**.

Untracked: An initial state of a file when it isn't yet part of the Git repository. Git is unaware of its existence.

Tracked: A tracked file is one that Git is actively monitoring. It can be in one of the following substates:

- **Unmodified:** The file is tracked, but it hasn't been modified since the last commit.
- **Modified:** The file has been changed since the last commit, but these changes aren't yet staged for the next commit.
- **Staged:** The file has been modified, and the changes have been added to the staging area (also known as the index). These changes are ready to be committed.
- **Committed:** The file is in the repository's database. It represents the latest committed version of the file.

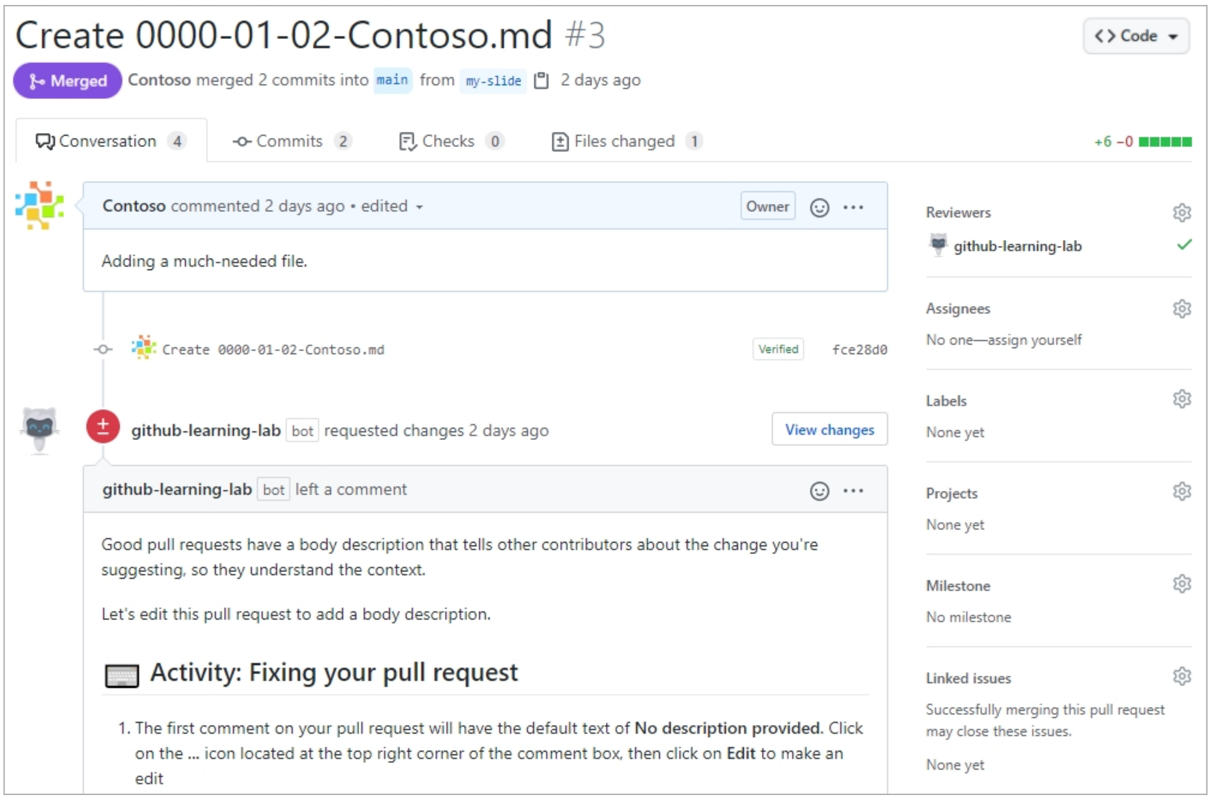
These states and substates are important to collaborating with your team to know where each and every commit is in the process of your project. Now let's move on to pull requests.

What are pull requests?

A **pull request** is the mechanism used to signal that the commits from one branch are ready to be merged into another branch.

The team member submitting the **pull request** asks one or more reviewers to verify the code and approve the merge. These reviewers have the opportunity to comment on changes, add their own, or use the pull request itself for further discussion.

Once the changes have been approved (if required), the pull request's source branch (the compare branch) is merged into the base branch.


Create 0000-01-02-Contoso.md #3

Merged Contoso merged 2 commits into main from my-slide 2 days ago

Contoso commented 2 days ago • edited

Adding a much-needed file.

Create 0000-01-02-Contoso.md Verified fce28d0

github-learning-lab bot requested changes 2 days ago

github-learning-lab bot left a comment

Good pull requests have a body description that tells other contributors about the change you're suggesting, so they understand the context.

Let's edit this pull request to add a body description.

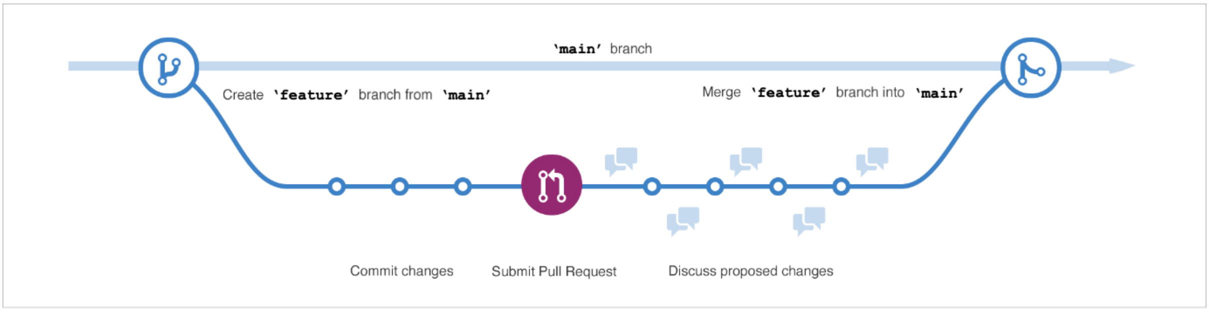
Activity: Fixing your pull request

1. The first comment on your pull request will have the default text of **No description provided.** Click on the ... icon located at the top right corner of the comment box, then click on **Edit** to make an edit

Reviewers: github-learning-lab ✓
Assignees: No one—assign yourself
Labels: None yet
Projects: None yet
Milestone: No milestone
Linked issues: Successfully merging this pull request may close these issues. None yet

Now that we know of all the ingredients, let's review the GitHub flow.

The GitHub flow



The GitHub flow can be defined as a lightweight workflow that allows for safe experimentation. You can test new ideas and collaboration with your team by using branching, pull requests, and merging.

Now that we know the basics of GitHub we can walk through the GitHub flow and its components.

1. Start by creating a branch so that the changes, features, and fixes you create don't affect the main branch.
2. Next, make your changes. We recommend deploying changes to your feature branch before merging into the main branch. Doing so ensures the changes are valid in a production environment.
3. Now, create a pull request to ask collaborators for feedback. Pull request review is so valuable that some repositories require an approving review before pull requests can be merged.
4. Then review and implement your feedback from your collaborators.
5. Once you feel great about your changes, it's time to get your pull request approved and merge it into the main branch.
6. Finally, you can delete your branch. Deleting your branch signals your work on the branch is complete and prevents you or others from accidentally using old branches.

That’s it, you’ve been through a GitHub flow cycle

GitHub is a collaborative platform

Collaboration is at the core of everything GitHub does. We went over repositories in the first unit of the module and learned that repositories help you organize your project and its files. In the last unit, we learned about pull requests, which is a way to keep track of changes made to your project.

In this unit, we're learning about issues and discussions. These are two other pieces that contribute to the collaborative nature of the GitHub Enterprise Platform.

Issues

GitHub Issues were created to track ideas, feedback, tasks, or bugs for work on GitHub. Issues can be created in various ways, so you can choose the most convenient method for your workflow.

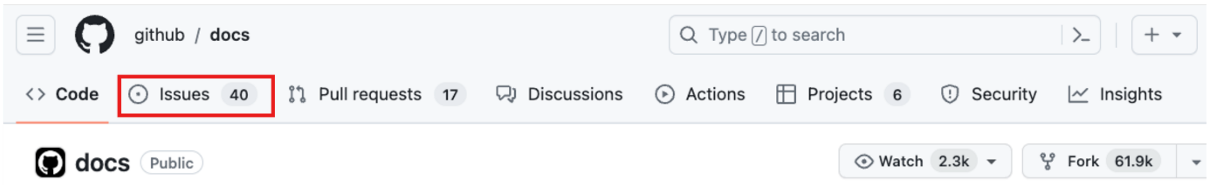
For this walkthrough, we'll go over how to create an issue from a repository. But issues can also be created from:

- An item in a task list.

- A note in a project.
- A comment in an issue or pull request.
- A specific line of code.
- A URL query.

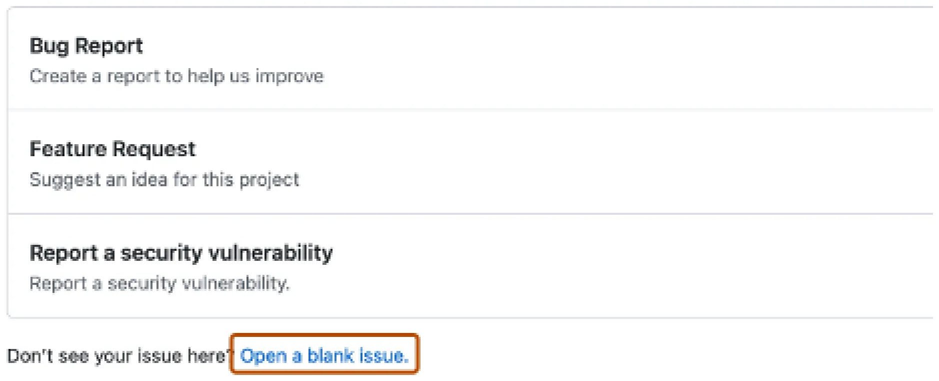
Creating an issue from a repository

1. On GitHub.com, navigate to the main page of the repository.
2. Under your repository name, select **Issues**.



3. Select **New issue**.
4. If your repository uses issue templates, next to the type of issue you'd like to open select **Get started**.

If the type of issue you'd like to open isn't included in the available options, select **Open a blank issue**. If not using templates, skip to Step 5.



Bug Report
Create a report to help us improve

Feature Request
Suggest an idea for this project

Report a security vulnerability
Report a security vulnerability.

Don't see your issue here? Open a blank issue.

5. In the **Add a title** field, enter a title for your issue.
6. In the **Add a description** field, type a description of your issue.
7. If you're a project maintainer, you can assign the issue to someone, add it to a project board, associate it with a milestone, or apply a label.
8. When you're finished, select **Submit new issue**.

Some conversations are more suitable for GitHub Discussions. You can use GitHub Discussions to ask and answer questions, share information, make announcements, and conduct or participate in conversations about a project.

In the next section, we'll review Discussions and how to best utilize the feature.

Discussions

Discussions are for conversations that need to be accessible to everyone and aren't related to code. Discussions enable fluid, open conversation in a public forum.

In this section we go over:

- Enabling a discussion in your repository.
- Creating a new discussion and various discussion categories.

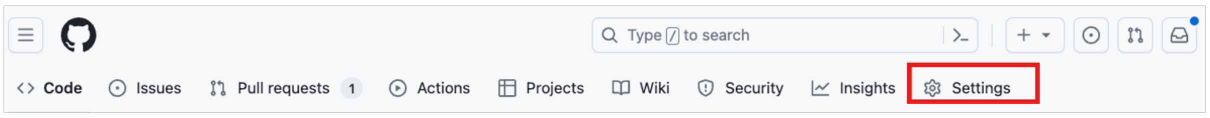
Let’s dive into enabling a discussion in your repository.

Enabling a discussion in your repository

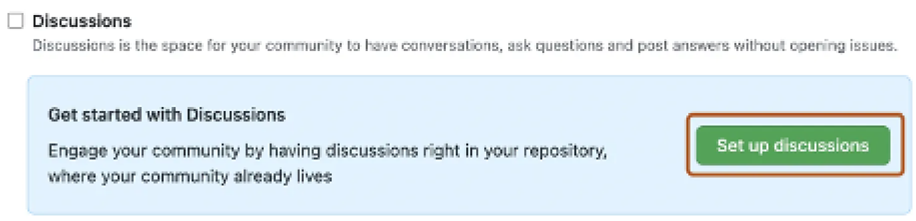
Repository owners and people with Write access can enable GitHub Discussions for a community on their public and private repositories. The visibility of a discussion is inherited from the repository the discussion is created in.

When you first enable GitHub Discussions, you're invited to configure a welcome post.

1. On GitHub.com, navigate to the main page of the repository.
2. Under your repository name, select **Settings**.



3. Scroll down to the **Features** section and under **Discussions**, select **Setup discussions**.


Discussions

Discussions is the space for your community to have conversations, ask questions and post answers without opening issues.

Get started with Discussions

Engage your community by having discussions right in your repository, where your community already lives

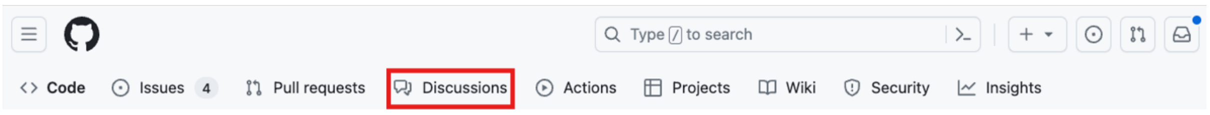
4. Under **Start a new discussion**, edit the template to align with the resources and tone you want to set for your community.
5. Select **Start discussion**.

You're now ready to create a new discussion.

Create a new discussion

Any authenticated user who can view the repository can create a discussion in that repository. Similarly, since organization discussions are based on a source repository, any authenticated user who can view the source repository can create a discussion in that organization.

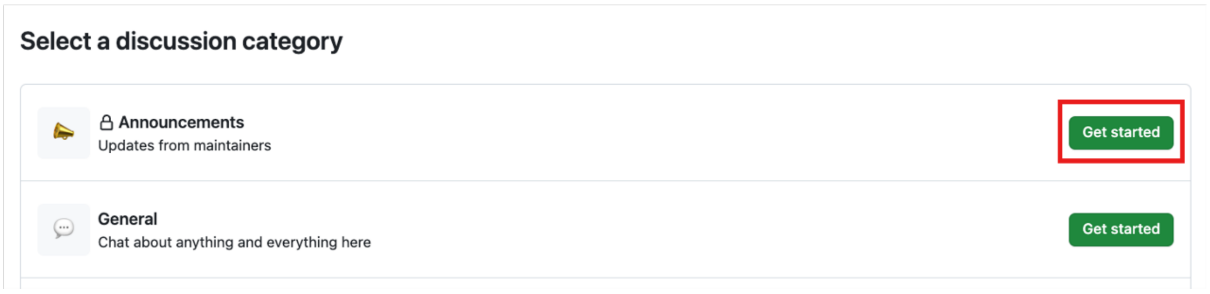
1. On GitHub.com, navigate to the main page of the repository or organization where you want to start a discussion.
2. Under your repository or organization name, select **Discussions**.



3. On the right side of the page, select **New discussion**.

Category	**Purpose**	**Format**
📣 Announcements	Updates and news from project maintainers	Announcement
# General	Anything and everything relevant to the project	Open-ended discussion
💡 Ideas	Ideas to change or improve the project	Open-ended discussion
🗳 Polls	Polls with multiple options for the community to vote for and discuss	Polls
🙏 Q&A	Questions for the community to answer, with a question/answer format	Question and Answer
🙌 Show and tell	Creations, experiments, or tests relevant to the project	Open-ended discussion

4. Select a discussion category by selecting **Get started**. All discussions must be created in a category. For repository discussions, people with maintain or admin permissions to the repository define the categories for discussions in that repository.


Select a discussion category

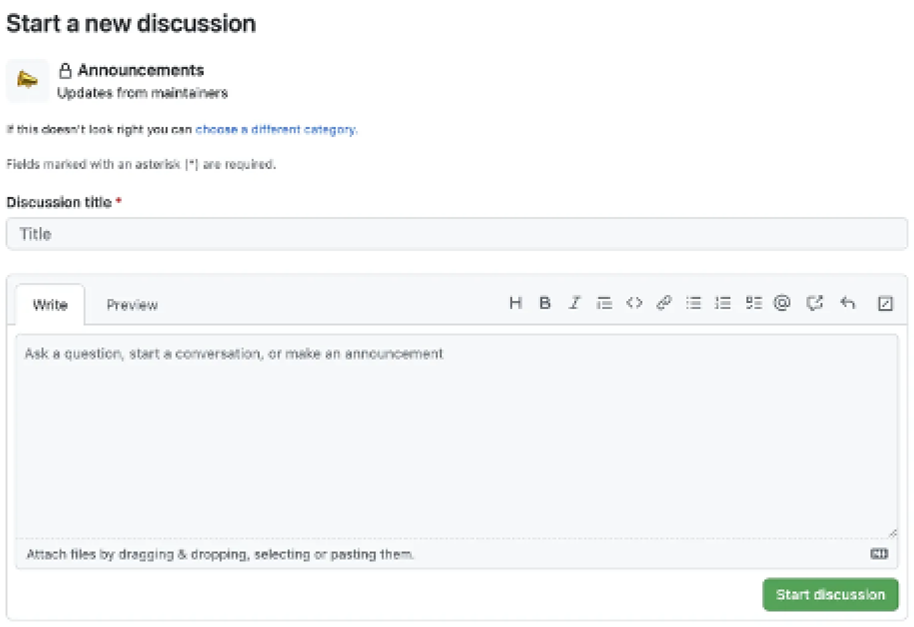
Announcements
Updates from maintainers

General
Chat about anything and everything here

Each category must have a unique name, emoji pairing, and a detailed description stating its purpose. Categories help maintainers organize how conversations are filed. They're customizable to help distinguish categories that are Q&A or more open-ended conversations. The following table shows the default categories for discussions and their purpose.

Expand table

1. Under **Discussion title** enter a title for your discussion, and under **Write** enter the body of your discussion.


Start a new discussion

Announcements
Updates from maintainers

If this doesn't look right you can choose a different category.

Fields marked with an asterisk (*) are required.

Discussion title *

Title

Ask a question, start a conversation, or make an announcement

Attach files by dragging & dropping, selecting or pasting them.

2. Select **Start discussion**.

That covers a little about how GitHub inspires collaboration. Now let's move to how you can manage notifications, subscribe to threads, and get started with GitHub pages.

GitHub platform management

Now that you know the basics of the GitHub platform, let’s go over some platform management.

In this unit, we'll cover:

- Managing notifications and subscriptions.
- Subscribing to threads and finding threads where you're mentioned.
- Publicizing your project or organization on GitHub pages.

Managing notifications and subscriptions

You can choose to receive ongoing updates about specific activity on GitHub.com through a subscription. Notifications are the updates that you receive for specific activity to which you're subscribed.

Subscription options

You can choose to subscribe to notifications for:

- A conversation in a specific issue, pull request, or gist.
- CI activity, such as the status of workflows in repositories set up with GitHub Actions.
- Repository issues, pull requests, releases, security alerts, or discussions (if enabled).
- All activity in a repository.

In some instances, you're automatically subscribed to conversations on GitHub. Examples include opening a pull request or issue, commenting on a thread, or being assigned to an issue or pull request.

If you're no longer interested in a conversation, you can unsubscribe, unwatch, or customize the types of notifications you'll receive in the future.

If you're ever interested in issues that mention a certain user, you can use *mentions:* as the qualifier to find those specific issues.

What are GitHub Pages?

To round out our journey of GitHub, let’s tackle GitHub pages. You can use GitHub Pages to publicize and host a website about yourself, your organization, or your project directly from a repository on GitHub.com.

GitHub Pages is a static site-hosting service that takes HTML, CSS, and JavaScript files straight from a repository on GitHub. Optionally, you can run the files through a build process and publish a website. Edit and push your changes, and your project is live for the public in a visually organized way.

Next up, we'll walk through an exercise to get you started with GitHub. In the next exercise, you'll:

- Create a new repository.
- Create a new branch.
- Commit a file.
- Open a pull request.
- And merge a pull request.

Next unit: Exercise - A guided tour of GitHub