UNIT-3

Uzma Sulthana

CONTENT

- **Working with Pull Requests-** Understanding a Pull Request, Pushing Code to GitHub, opening a Pull Request, writing a Great Pull Request, Reviewing a Pull Request.

Manage and Contribute to large projects:

- **Exploring and Contributing to OSS -** Exploring GitHub, Finding Places to Contribute, Surveying a Project for Contribution, Setting Contributor Expectations.
- **Starting Your Own OSS-** Creating an Open Source Repository, Making a Repository Public, Enforcing a Code of Conduct, Writing a README.md File, Writing Good Documentation, Managing Issues, Ending Your Project

UNDERSTANDING A PULL REQUEST

- A pull request is a request to the maintainer of a repository to pull in some code.
- When you write some code that you want to contribute to a repository, you create and submit a pull request.
- Your code contains some proposed changes to the target repository.
- A pull request is your way of offering these changes to the maintainer of the repository.
- It gives the repository maintainers an opportunity to review the changes and either accept them, reject them, or ask for more changes to be made.

PUSHING CODE TO GITHUB

To push code to GitHub, you need a repository.

The first thing to do is create a new branch.

Creating a new branch before writing new code is standard operating procedure with Git.

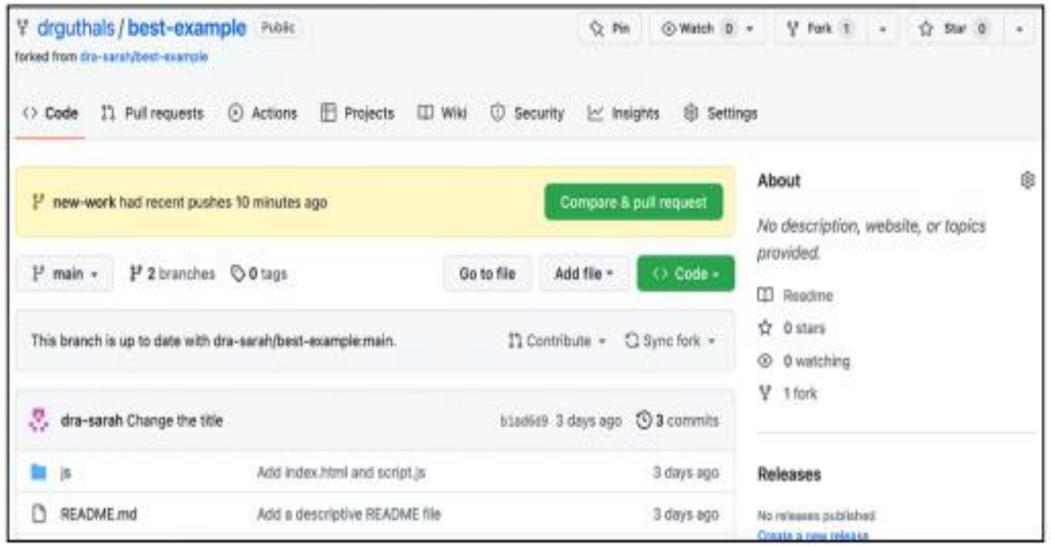
A pull request is always associated with a branch.

- $ **git checkout -b new-work** (create a new branch when starting new work. I name the branch new-work for this example, but feel free to name it whatever you want by replacing new-work with your own branch name in the following command)
- $ **echo "\n## Installation" >> README.md**
- $ **git add -A**
- **$ git commit -m "Add text to the README“**
- **$ git push -u origin new-work**

OPENING A PULL REQUEST

Before you can open a pull request, your GitHub.com repository must have at least one branch other than the default branch.

When you visit the repository on GitHub.com, you see a new message in the at the top of the Code tab, as shown in Figure 8-1.


drguthals / best-example Public

forked from dra-sarah/best-example

new-work had recent pushes 10 minutes ago

This branch is up to date with dra-sarah/best-example:main.

dra-sarah Change the title b1ad6d9 3 days ago 3 commits

js	Add index.html and script.js	3 days ago
README.md	Add a descriptive README file	3 days ago

About

No description, website, or topics provided.

Releases

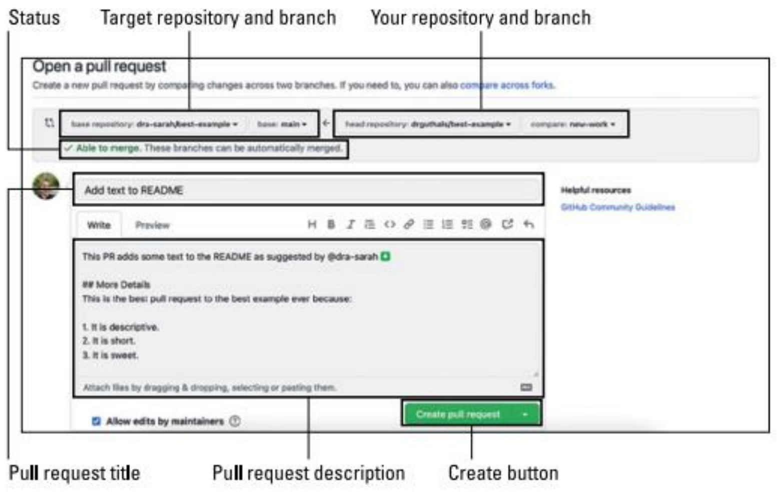
No releases published

FIGURE 8-1: Repository home page with recently pushed branches listed.

OPENING A PULL REQUEST

- Click the Compare & Pull Request button to navigate to the Open a Pull Request page, as shown in Figure 8-2.
- The target branch (the branch you want to pull your changes into) is the default branch for the repo. Your branch is listed next to the target branch, and a status of whether your branch can be merged into the target branch is next to that.
- The pull request title is the same as the most recent commit message — in this example Add text to README — and your description is blank.

Figure 8-2 shows an example description.



Open a pull request

Create a new pull request by comparing changes across two branches. If you need to, you can also compare across forks.

base repository: dra-sarah/best-example ▾ base: main ▾ ← head repository: drguthals/best-example ▾ compare: new-work ▾

✓ Able to merge. These branches can be automatically merged.

Add text to README

This PR adds some text to the README as suggested by @dra-sarah

More Details
This is the best pull request to the best example ever because:

1. It is descriptive.
2. It is short.
3. It is sweet.

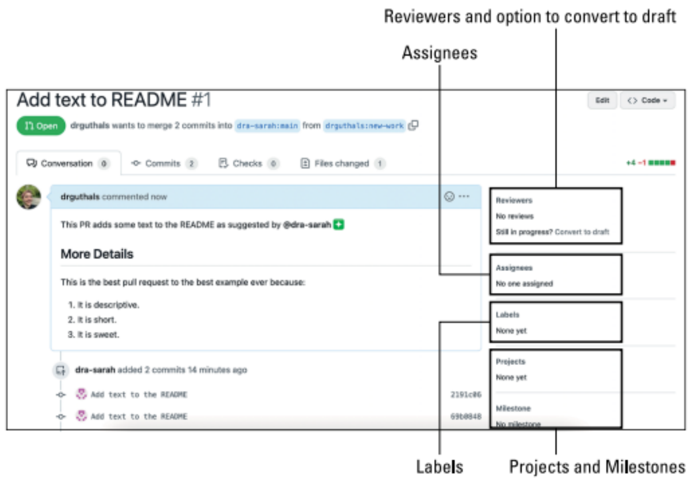
Attach files by dragging & dropping, selecting or pasting them.

☑ Allow edits by maintainers

FIGURE 8-2: The Open a Pull Request page.

OPENING A PULL REQUEST

- Click the Create Pull Request button, and you go to the repository with the new pull request open.
- Here you can see that the account dra-sarah, or any accounts who have been added as collaborators to this repository, can assign reviewers, assign assignees, apply labels, connect to a project, or add to a milestone.
- Figure 8-3 shows the pull request opened on the dra-sarah repository.



Add text to README #1

drguthals wants to merge 2 commits into dra-sarah:main from drguthals:new-work

drguthals commented now

This PR adds some text to the README as suggested by @dra-sarah

More Details

This is the best pull request to the best example ever because:

1. It is descriptive.
2. It is short.
3. It is sweet.

dra-sarah added 2 commits 14 minutes ago

Add text to the README 2191c86

Add text to the README 69b0848

Reviewers
No reviews
Still in progress? Convert to draft

Assignees
No one assigned

Labels
None yet

Projects
None yet

Milestone
No milestone

FIGURE 8-3: The pull request opened on the dra-sarah/best-example repository.

ADDING REVIEWERS

- The first field, Reviewers, lets the repository owner or contributors specify one or more people to review your pull request.
- **To add reviewers if you are the owner or a contributor on a repository:**
- 1. Click Reviewers to see a list of people you can mention.
  - For repositories with a large number of users, you can start typing to filter the set of users.
- 2. Click each user to add them to the list of reviewers.
  - When you add a reviewer, they're immediately notified when you finish creating the pull request.

SPECIFYING ASSIGNEES

- **The next option after Reviewers is an option to specify assignees.**
- An assignee is the person who should take action on the pull request.
- Often, a pull request represents a work in progress and not the final result of some work.
- If more work needs to be done on a pull request, you'd assign the pull request to the person that should do the work.
- **To specify assignees:**
- 1. Click Assignees to see a list of assignees.
  - The assignee dialog box works just like the reviewers dialog box, described in the preceding section. It allows you to select one or more assignees.
- 2. Click each user to add them to the list of assignees.

SPECIFYING LABELS, SPECIFYING PROJECTS AND MILESTONES

Specifying Labels

- Labels provide convenient grouping and context to help you decide what to work on next or what to review next.
- The set of labels you can use on issues and pull requests are the same, but some labels make more sense for issues than pull requests and vice versa.
- For example, many repositories have a "**ready for review**" label specifically for pull requests.

Specifying projects and milestones

- The last two options allow you to specify the project board and milestone that this pull request belongs to.

WRITING A GREAT PULL REQUEST

1. Knowing your audience
2. Making the purpose clear
3. Keeping it focused
4. Explaining the why
5. A picture is worth a thousand words
6. Including a call to action

KNOWING YOUR AUDIENCE

- **Know Your Primary Audience:**
  - The main focus should be on the reviewers and maintainers who decide whether the pull request (PR) will be merged.
  - Make their review process as efficient as possible by being clear and concise.
- **Consider Secondary Audiences:**
  - Beyond maintainers, others (like future contributors or curious users) may read the PR.
  - In open-source projects, this could include anyone globally, so clarity is essential.

- **Use Respectful Language:**

  •Maintain a tone that is respectful, friendly, and inclusive to ensure your PR is welcoming and accessible to diverse audiences.

MAKING THE PURPOSE CLEAR

- Make sure to be concise and informative.

- For example,
- the summary should make the purpose of the pull request clear.
- The summary is the only part shown on the page that lists pull requests.
- It needs to be scannable.

Here are some examples of good pull request summaries:

- » Adds the About page to the website
- » Minimizes boilerplate setup code for JavaScript libraries
- » Extracts and isolates error handling from GitStore internals

Here are some bad examples taken from my own repositories:

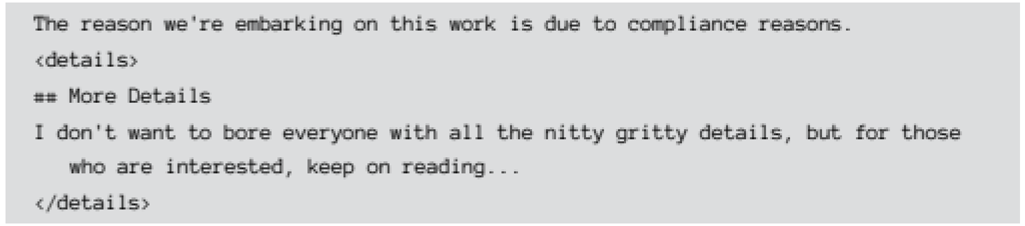
- » Teams are forever
- » Typo
- » Small changes

KEEPING IT FOCUSED

- Much like a commit, a pull request should not contain a bunch of unrelated changes.
- A pull request may consist of multiple commits, but they should all be related to the task at hand.
- You can often tell that a pull request is doing too much when writing a concise description of what the pull request accomplishes is difficult.
- Even if the pull request is focused on a single major change, keep the pull request to a manageable size.
- Reviewing a very large pull request is difficult.
- If the pull request addresses a very large task, break down the task into smaller steps and submit pull requests for each step.

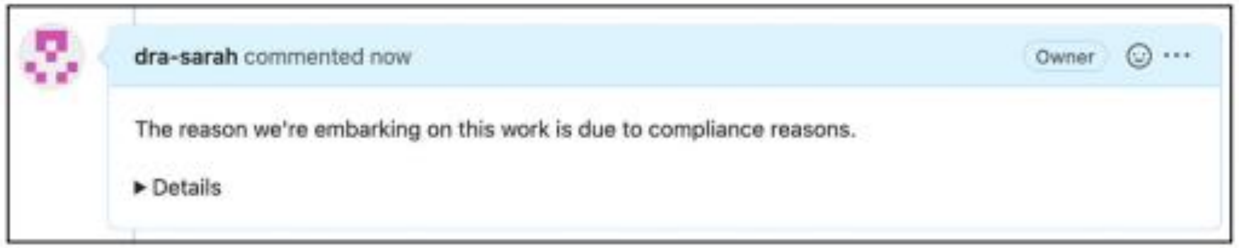
EXPLAINING THE WHY

- You also need to explain why you're taking on this work.
- The pull request description is an opportunity to provide links to other documents that provide more context.
- You can't assume everyone will be familiar with the history.
- If you have a lot of context to share, you can provide a brief summary followed by more details within a <details> tag.
- For example, if you add a pull request comment with the following text:



```
The reason we're embarking on this work is due to compliance reasons.
<details>
## More Details
I don't want to bore everyone with all the nitty gritty details, but for those
   who are interested, keep on reading...
</details>
```

GitHub displays the details section collapsed by default, as shown in Figure 8-4.



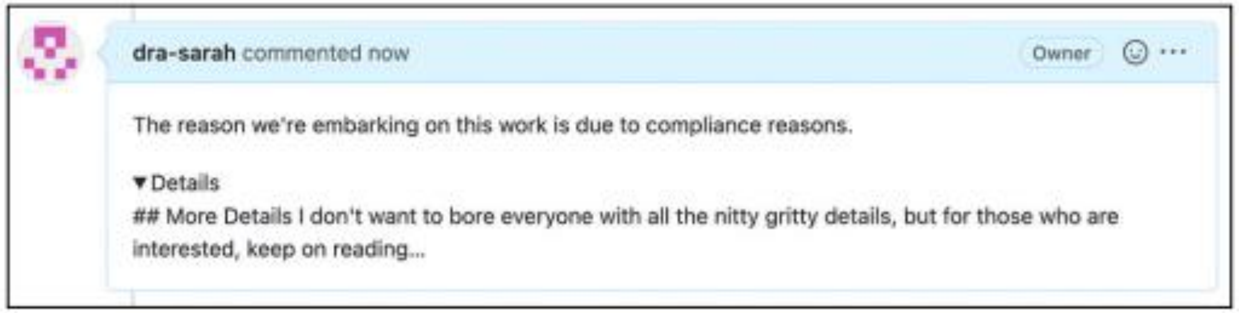
dra-sarah commented now

The reason we're embarking on this work is due to compliance reasons.

▶ Details

FIGURE 8-4: Details section collapsed.

Click the details label to expand the details section, as shown in Figure 8-5.



dra-sarah commented now

The reason we're embarking on this work is due to compliance reasons.

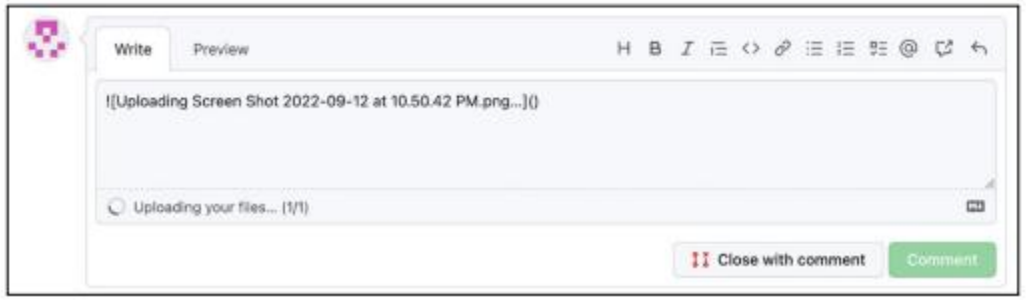
▼ Details

More Details I don't want to bore everyone with all the nitty gritty details, but for those who are interested, keep on reading...

FIGURE 8-5: Details section expanded.

A PICTURE IS WORTH A THOUSAND WORDS

- GitHub supports adding images to a pull request description by dragging and dropping an image. When you drag and then drop the image on the text field,
- GitHub uploads the image and replaces it with the Markdown for rendering an image.
- Figure 8-6 shows an upload in progress after I dragged an image into a pull request comment field.



![Uploading Screen Shot 2022-09-12 at 10.50.42 PM.png...]()

Uploading your files... (1/1)

FIGURE 8-6: Uploading an image to a pull request.

Visit https://github.com/dra-sarah/best-example/pull/1-issuecomment-1244931357 to see this comment. It's very meta as it's a screenshot of the same repository it's a comment on.

INCLUDING A CALL TO ACTION

- You need to be very clear about what feedback you want from others on the pull request.
- For example, if the pull request is a work in progress, make that clear
- from the start so that people don’t waste time reviewing a pull request that isn’t ready for review.
- To make that clear, follow the conventions of the repository. Following the conventions is important so that others know what is expected of them with respect to your pull request.

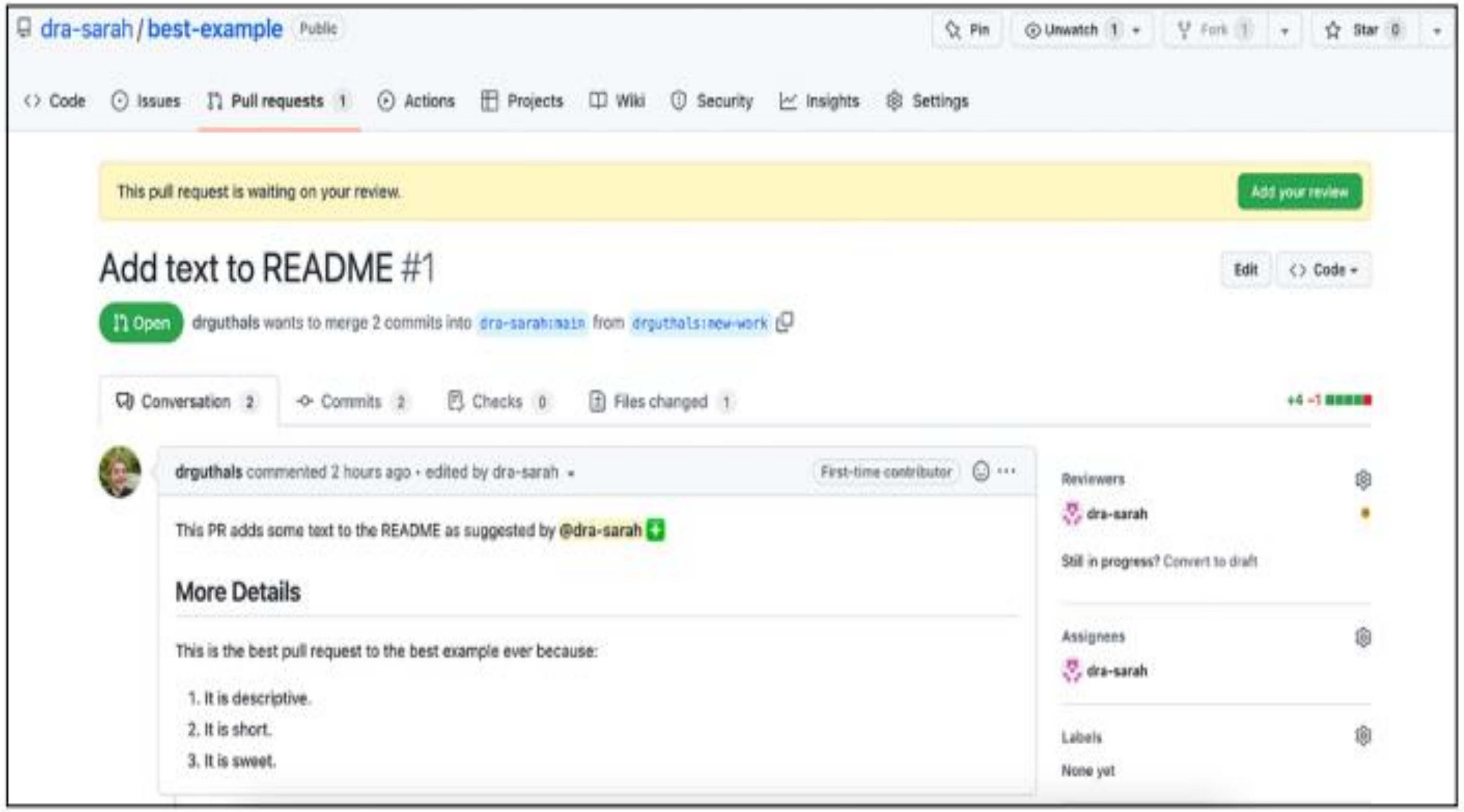
“How to write the perfect pull request” at https://blog.github.com/2015-01-21-how-to-write-the-perfect-pull-request/.

REVIEWING A PULL REQUEST

1. Reviewing the Conversation tab
2. Reviewing the changed files
3. Commenting on code
4. Suggesting changes
5. Finishing the review .

REVIEWING A PULL REQUEST

- Pull requests have two parties involved: the person who writes and opens the pull request, and the person (or people) who reviews it.
- Reviewing a pull request is a very active activity.
- It doesn’t serve the folks submitting pull requests if you just take a cursory look at it, comment LGTM (Looks Good To Me), and don’t provide quality feedback.
- Sometimes, a cursory look is all you have time for.
- In that case, make it clear what you did and didn’t review and suggest that someone else provide a more detailed review.
- When someone adds you as a reviewer, you’ll receive a notification about
- the request, typically via email or on your notification bell at the top right of GitHub.com.
- If you visit the pull request, you see a banner message, as shown in Figure 8-7.



This pull request is waiting on your review.

Add text to README #1

drguthals wants to merge 2 commits into `dra-sarah:main` from `drguthals:new-work`

drguthals commented 2 hours ago • edited by dra-sarah

This PR adds some text to the README as suggested by @dra-sarah

More Details

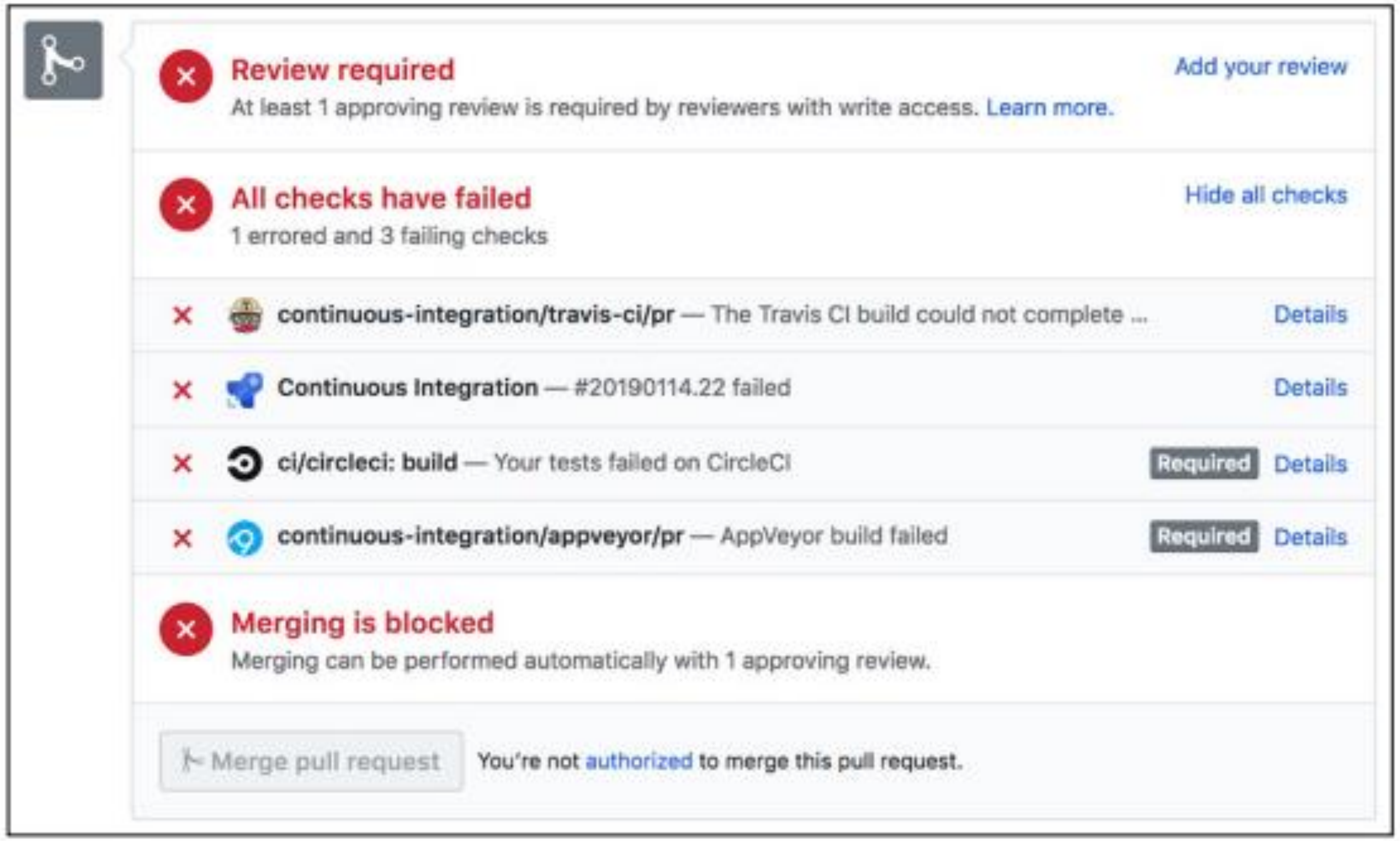
This is the best pull request to the best example ever because:

1. It is descriptive.
2. It is short.
3. It is sweet.

FIGURE 8-7: Message requesting your review on this pull request.

1.REVIEWING THE CONVERSATION TAB

- When you review a pull request, the first thing to do is read through the contents of the Conversation tab for the pull request.
- At the bottom of the conversation is a section that displays the checks that GitHub runs against the repository if there are any.
- If no checks are set up, you see the message Continuous integration has not been set up.
- The status of these checks is the next thing you should check.
- If any of these tests fail, you shouldn't spend any more time reviewing the code
- Figure 8-8 shows an example from the GitHub Desktop open source project of a pull request that fails the continuous integration (CI) builds.



Review required

At least 1 approving review is required by reviewers with write access. Learn more.

Add your review

All checks have failed

1 errored and 3 failing checks

Hide all checks

✗	**continuous-integration/travis-ci/pr** — The Travis CI build could not complete ...		Details
✗	**Continuous Integration** — #20190114.22 failed		Details
✗	**ci/circleci: build** — Your tests failed on CircleCI	Required	Details
✗	**continuous-integration/appveyor/pr** — AppVeyor build failed	Required	Details

Merging is blocked

Merging can be performed automatically with 1 approving review.

Merge pull request

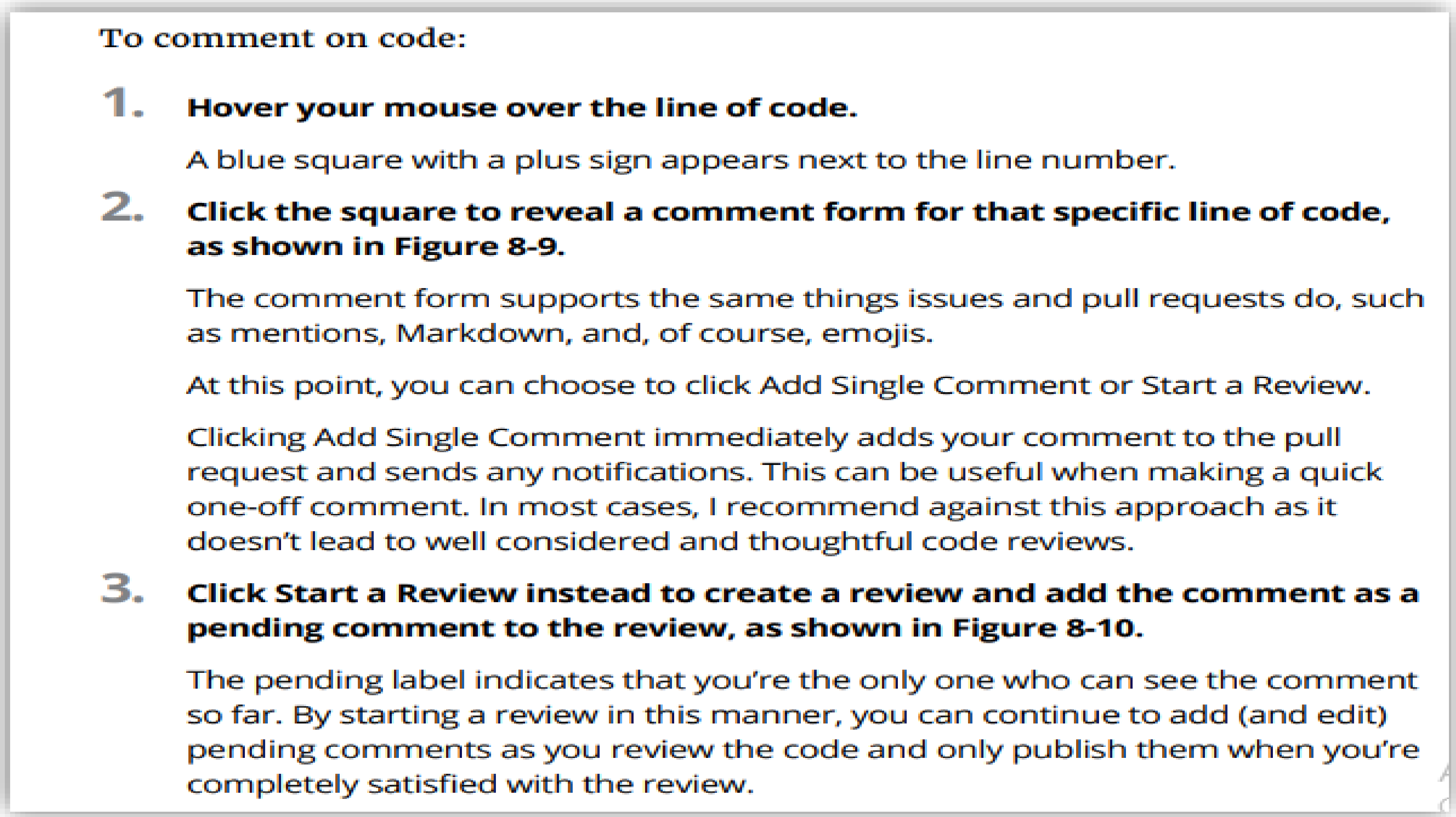
You're not authorized to merge this pull request.

FIGURE 8-8:
Pull request with failed checks.

2. REVIEWING THE CHANGED FILES

- **Clarity:** Is the code easy to read and understand? Could it be made more clear? Are there appropriate comments throughout the code? Obtuse, hard-to-understand code becomes a maintenance nightmare down the road.
- **Correctness:** Does the code do what the pull request says it does? Are there any glaring bugs? Are there any errors of omission? Are there any tests missing?
- **Security:** Related to the previous item, a security review requires a specific mindset. Ideally, you work with security experts who can help review the code for security. The idea here is to think about all the ways a bad actor could attempt to attack the code. There are many frameworks for doing security review, such as STRIDE. You should also think about how bad actors can use the code to harm other users. Does the code protect users privacy? Does it ask for consent to take actions on behalf of users?
- **Conventions and idioms:** Just because code is correct, it doesn’t mean it’s necessarily idiomatic. A code review is a good place to teach and learn conventions and idioms specific to a project.

3. COMMENTING ON CODE



To comment on code:

1. **Hover your mouse over the line of code.**

   A blue square with a plus sign appears next to the line number.

2. **Click the square to reveal a comment form for that specific line of code, as shown in Figure 8-9.**

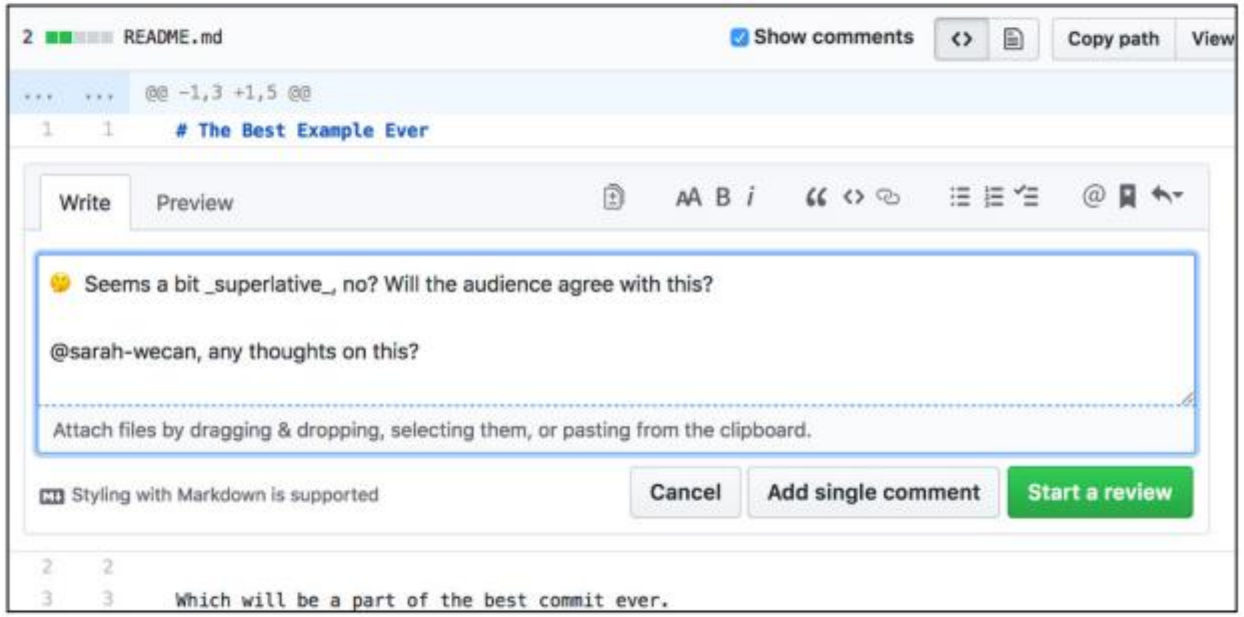
   The comment form supports the same things issues and pull requests do, such as mentions, Markdown, and, of course, emojis.

   At this point, you can choose to click Add Single Comment or Start a Review.

   Clicking Add Single Comment immediately adds your comment to the pull request and sends any notifications. This can be useful when making a quick one-off comment. In most cases, I recommend against this approach as it doesn't lead to well considered and thoughtful code reviews.

3. **Click Start a Review instead to create a review and add the comment as a pending comment to the review, as shown in Figure 8-10.**

   The pending label indicates that you're the only one who can see the comment so far. By starting a review in this manner, you can continue to add (and edit) pending comments as you review the code and only publish them when you're completely satisfied with the review.



2 README.md

Show comments | Copy path | View

```
... ... @@ -1,3 +1,5 @@
1   1   # The Best Example Ever
```

Write | Preview

Seems a bit _superlative_, no? Will the audience agree with this?

@sarah-wecan, any thoughts on this?

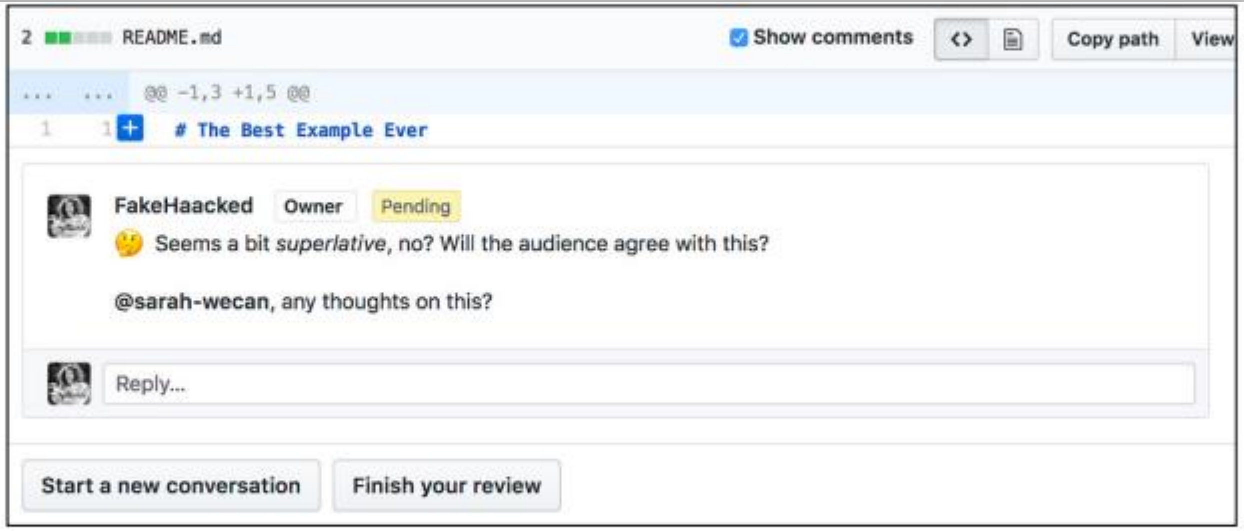
Attach files by dragging & dropping, selecting them, or pasting from the clipboard.

Styling with Markdown is supported

Cancel | Add single comment | Start a review

```
2   2
3   3   Which will be a part of the best commit ever.
```

FIGURE 8-9: Pull Request Comment form for a line of code.



2 README.md

Show comments | Copy path | View

```
... ... @@ -1,3 +1,5 @@
1   1   # The Best Example Ever
```

FakeHaacked Owner Pending

Seems a bit *superlative*, no? Will the audience agree with this?

@sarah-wecan, any thoughts on this?

Reply...

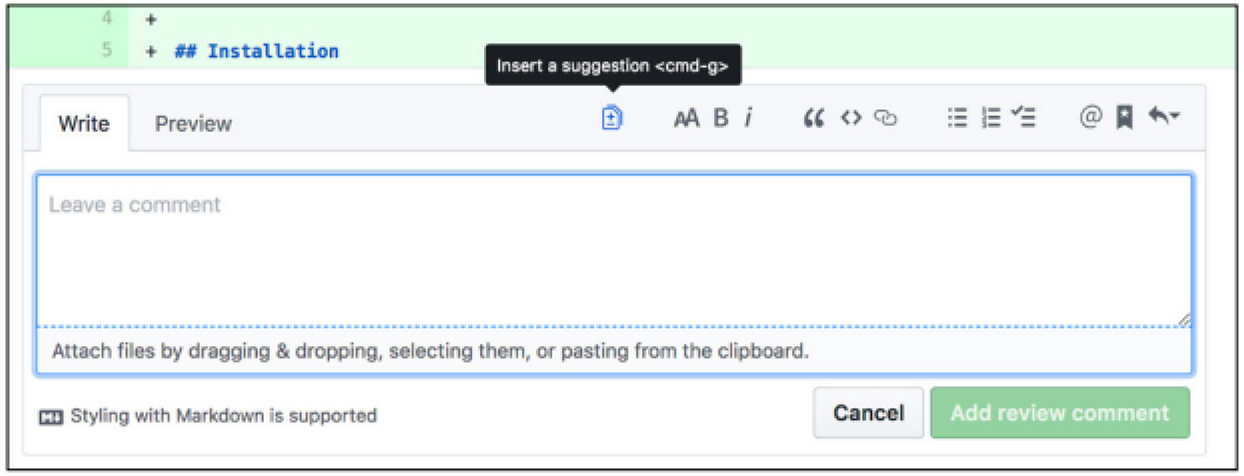
Start a new conversation | Finish your review

FIGURE 8-10: Pending comment in a pull request review.

4. SUGGESTING CHANGES

To suggest a change:

1. **Opening the comment form for the line of code you want to change.**
When you bring up the comment form, you see an icon with a + and - symbol. When you hover over the symbol, the tooltip describes the purpose of this button (see Figure 8-11). The tooltip also describes a keyboard shortcut you can use to suggest a change, CMD-G.
2. **Click the Suggest Changes button.**
GitHub adds a suggestion block into the body of your comment with the current contents of the line you want to change.
3. **Change the contents of the block to suggest what the final result should be.**
Figure 8-12 shows an example where I suggest a change to add the word "Instructions" to the line.
4. **Click the Add Review Comment button to save your comment to this review.**


```
4  +
5  + ## Installation
```

Insert a suggestion <cmd-g>

Write Preview

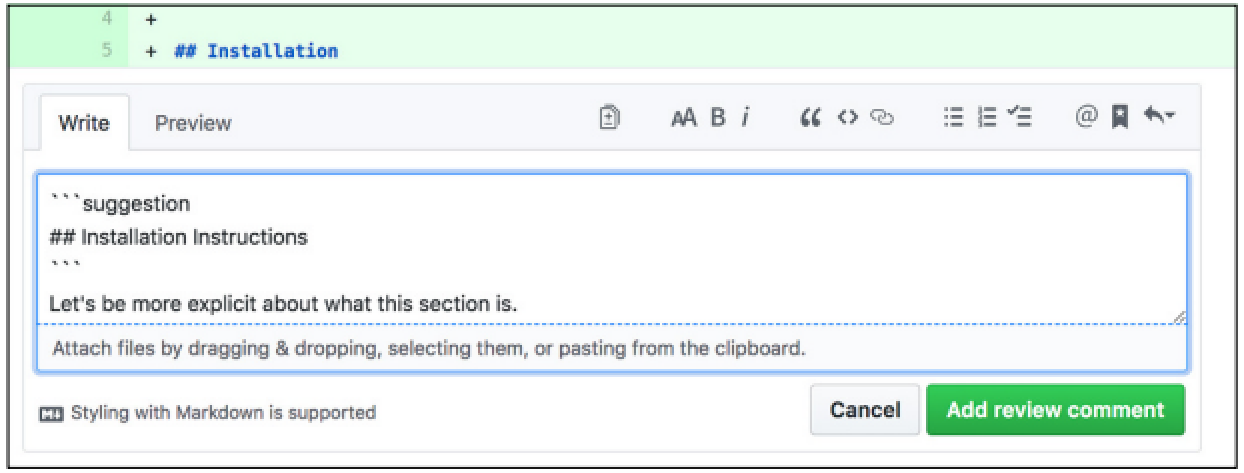
Leave a comment

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

Styling with Markdown is supported

Cancel Add review comment

FIGURE 8-11: Comment form with the Suggest changes button.


```
4  +
5  + ## Installation
```

Write Preview

~~~
```suggestion
## Installation Instructions
```

Let's be more explicit about what this section is.
~~~

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

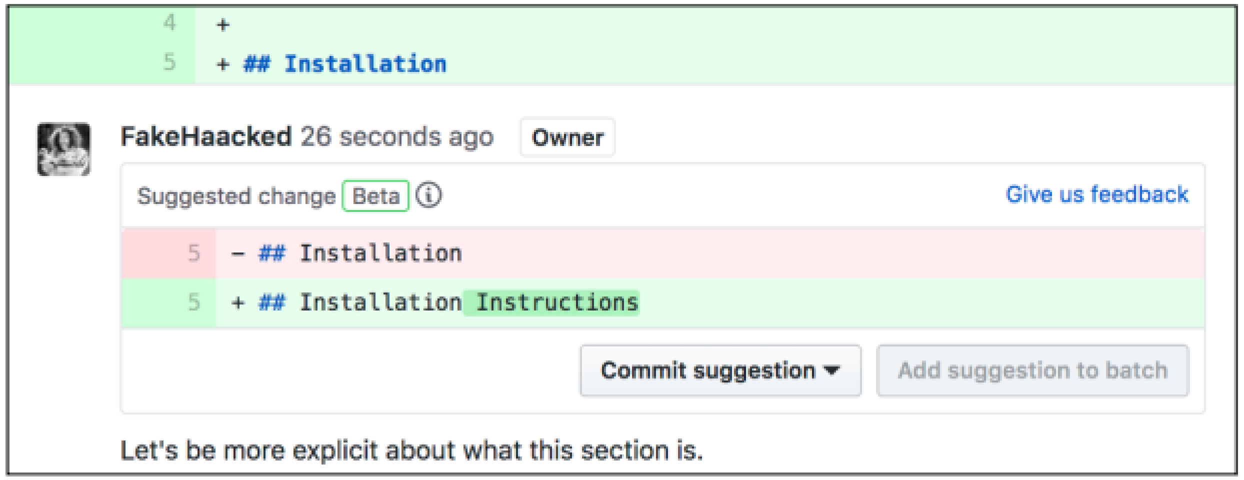
Styling with Markdown is supported

Cancel Add review comment

FIGURE 8-12: Comment with a suggested change.

5. FINISHING THE REVIEW

After all your suggestions, we can finish the review so the author receives all your valuable feedback and can start to address it.
To finalize the review, click the Finish Your Review button at the bottom of any pending comment (refer to Figure 8-10).



```
4  +
5  + ## Installation
```

FakeHaacked 26 seconds ago Owner

Suggested change Beta — Give us feedback

```
5  - ## Installation
5  + ## Installation Instructions
```

Let's be more explicit about what this section is.

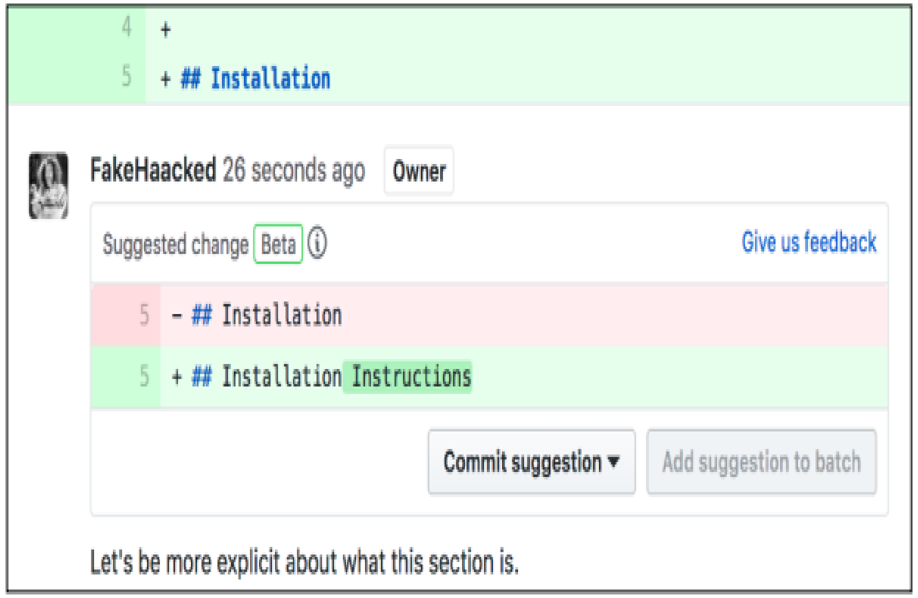
FIGURE 8-13: Comment with a suggested change.

5. FINISHING THE REVIEW

You see a **form** where you can write your overall summary about the pull request.

- This form is an opportunity to bring up any review comments that are not specific to any lines of code. It is also a good opportunity to offer some general praise, raise broad concerns, suggest follow-up actions, and so on.
- After you type your comment, check one of the options to indicate your position

on the pull request. Figure 8-14 shows the comment form with the review options.



```
4  +
5  + ## Installation
```

FakeHaacked 26 seconds ago Owner

Suggested change Beta

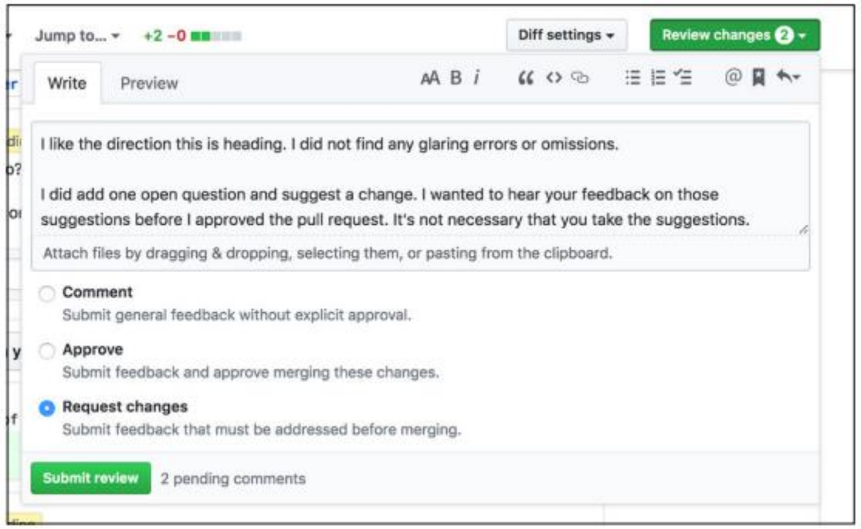
Give us feedback

```
5  - ## Installation
5  + ## Installation Instructions
```

Commit suggestion ▾ | Add suggestion to batch

Let's be more explicit about what this section is.

FIGURE 8-13: Comment with a suggested change.



I like the direction this is heading. I did not find any glaring errors or omissions.

I did add one open question and suggest a change. I wanted to hear your feedback on those suggestions before I approved the pull request. It's not necessary that you take the suggestions.

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

- Comment — Submit general feedback without explicit approval.
- Approve — Submit feedback and approve merging these changes.
- [✓ Request changes] — Submit feedback that must be addressed before merging.

Submit review 2 pending comments

FIGURE 8-14: Comment with a suggested change.

CONTENT

Manage and Contribute to large projects:

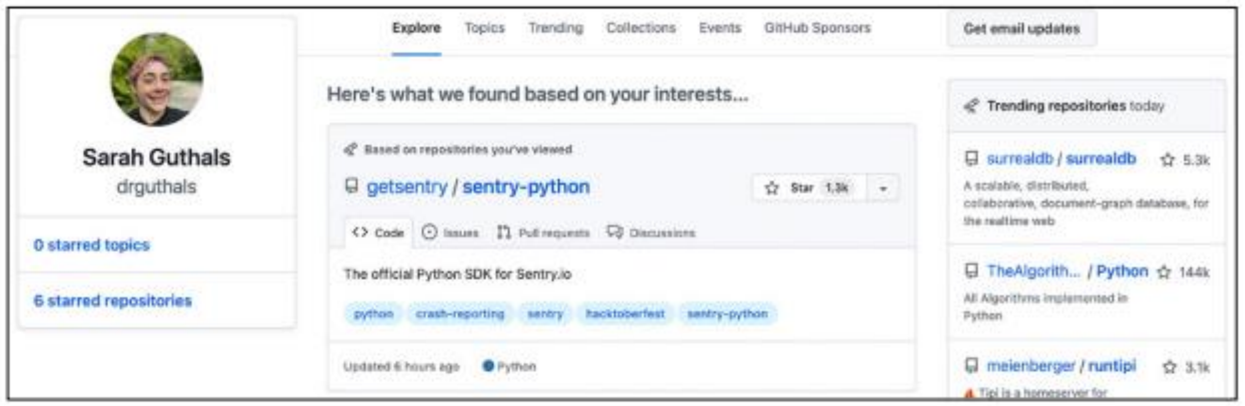
- **Exploring and Contributing to OSS**
  - Exploring GitHub
  - Finding Places to Contribute
  - Surveying a Project for Contribution
  - Setting Contributor Expectations.
- **Starting Your Own OSS**
  - Creating an Open Source Repository
  - Making a Repository Public
  - Enforcing a Code of Conduct
  - Writing a README.md File
  - Writing Good Documentation
  - Managing Issues
  - Ending Your Project

EXPLORING AND CONTRIBUTING TO OSS

- open source software (OSS), contributing to open source is a fantastic way to continue your development as a software developer.
- It gives you the opportunity to work with technologies that you may not otherwise work with in your day job.
- It connects you with a large community of developers working on a diverse array of challenges.
- Many of these developers are happy to share their knowledge with folks looking to learn something new.

1. EXPLORING GITHUB

- The Explore page on GitHub, located at https://github.com/explore, is a great starting point to discover repositories, topics, collections, and OSS projects that may align with your interests. It contains a mix of human and algorithmically curated content.
- The Explore page displays a selection of repositories based on your interests.
- This selection is an algorithmically curated page of recommendations specific to you.
- Figure 9-1 shows a portion of the Explore page.



Sarah Guthals
drguthals
0 starred topics
6 starred repositories

Here's what we found based on your interests...

Based on repositories you've viewed
getsentry / sentry-python
The official Python SDK for Sentry.io
Updated 6 hours ago · Python

Trending repositories today
surrealdb / surrealdb ☆ 5.3k
A scalable, distributed, collaborative, document-graph database, for the realtime web
TheAlgorith... / Python ☆ 144k
All Algorithms implemented in Python
meienberger / runtipi ☆ 3.1k
Tipi is a homeserver for

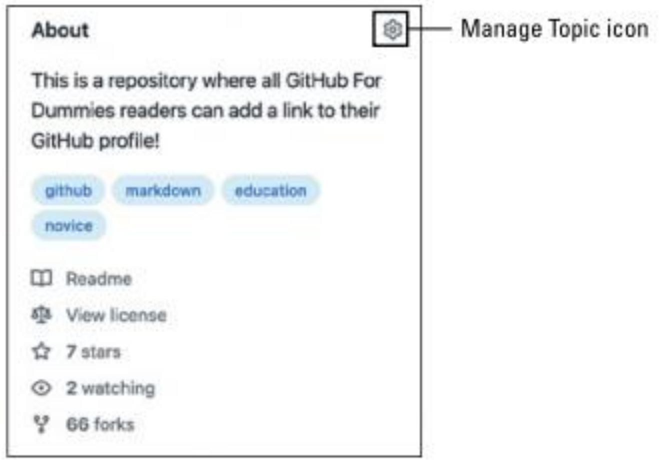
FIGURE 9-1: The Explore page on GitHub.

1. EXPLORING GITHUB

- Exploring topics
- Trending repositories
- Exploring collections
- Exploring events
- Exploring GitHub Sponsors

1. EXPLORING GITHUB

- **Exploring topics**
  - The Popular Topics section lists the most popular topics on GitHub.
  - A topic is a user-applied category for a repository. A topic gives people more information about what a repository is about. A repository owner can specify multiple topics for a repository.
  - The concept is very similar to tags or issue labels.
  - Figure 9-2 shows the list of topics for the thewecanzone/GitHubForDummies Readers repository.
  - Admins for the repository see a Manage Topics icon that lets them add or remove topics for the repository


About

This is a repository where all GitHub For Dummies readers can add a link to their GitHub profile!

github markdown education novice

Readme

View license

7 stars

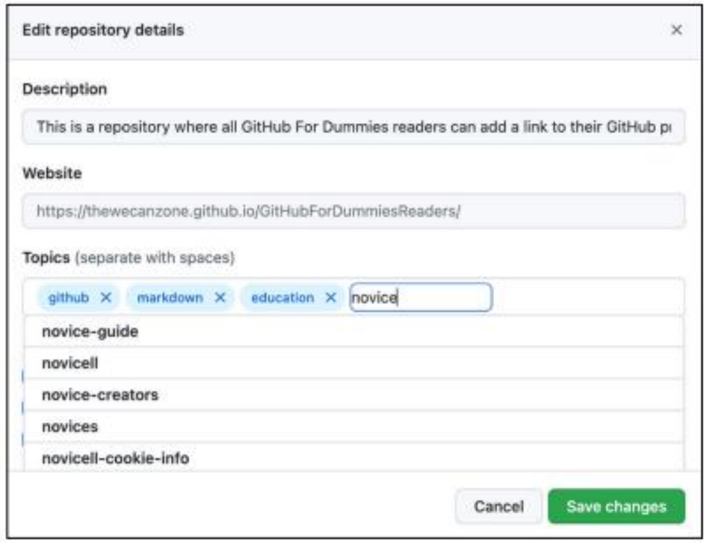
2 watching

66 forks

FIGURE 9-2: List of topics for a repository.

1. EXPLORING GITHUB

- **Exploring topics**
  - When you add topics, you type a portion of a topic name, and GitHub offers suggestions based on the topics that others use within GitHub.
  - Figure 9-3 shows an
  - example where I typed novice, and GitHub suggests other topics with the word novice in them.


Edit repository details

Description

This is a repository where all GitHub For Dummies readers can add a link to their GitHub pı

Website

https://thewecanzone.github.io/GitHubForDummiesReaders/

Topics (separate with spaces)

github ✕ markdown ✕ education ✕ novice

novice-guide
novicell
novice-creators
novices
novicell-cookie-info

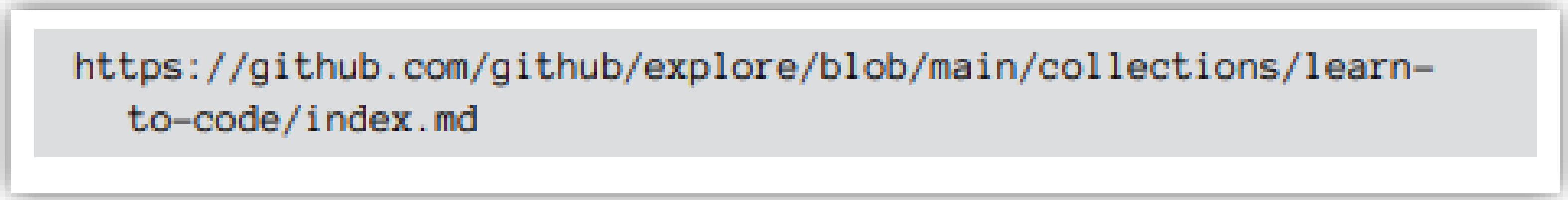
FIGURE 9-3: A list of suggested topics.

1. EXPLORING GITHUB

- **Trending repositories**
  - The next section on the Explore page includes a list of the top 25 trending repositories.
  - Click the heading to visit https://github.com/trending, a page where you can discover repositories and people that are trending across GitHub.
  - This page lets you filter trending repositories based on the **primary language** of the repository.
  - If you're interested in the trending JavaScript libraries, click the Other Languages button and select JavaScript.
  - The Trending button at the top of the page defaults to today, but you can click it to see what's been trending for the week and month.
  - To determine what's trending, GitHub looks at a variety of data points, such as stars, forks, commits, follows, and pageviews.
  - GitHub weighs these data points appropriately and factors in how recent the events were, not just total numbers.

1. EXPLORING GITHUB

- **Exploring collections**
  - The next three sections are curated collections. At the time of writing, they are currently Pixel Art Tools, Game Engineers, and Made in Brazil.
  - Each collection is fully curated by a human.
  - A collection may contain a list of web pages and repositories.
  - The goal is to be a great starting point for learning a particular subject in depth by listing websites for further reading and repositories with related code.
  - For example, to edit The Learn to Code collection, suggest an edit to this file in the github/explore repository:



```
https://github.com/github/explore/blob/main/collections/learn-
   to-code/index.md
```

1. EXPLORING GITHUB

Exploring events

- The Events section lists a selection of upcoming GitHub affiliated events, such as **GitHub Universe** (**GitHub's yearly flagship conference**), **GitHub Satellite** (**smaller GitHub community events hosted around the world**), and **All Things Open** (**an event promoting open source**).
- Check this page often to find out about future events, which can be a great way to connect with the larger GitHub community.

1. EXPLORING GITHUB

- **Exploring GitHub Sponsors**

- The GitHub Sponsors section takes you to https://github.com/sponsors/ explore where you can find a list of all of the open source projects and maintainers that create dependencies you have in your repositories.
- This is a great way to monetarily give back to the projects and people that make what you do possible.

Getting by with help from your friends

2. FINDING PLACES TO CONTRIBUTE

There are many valid reasons for contributing to open source.
One or more of the following may apply:

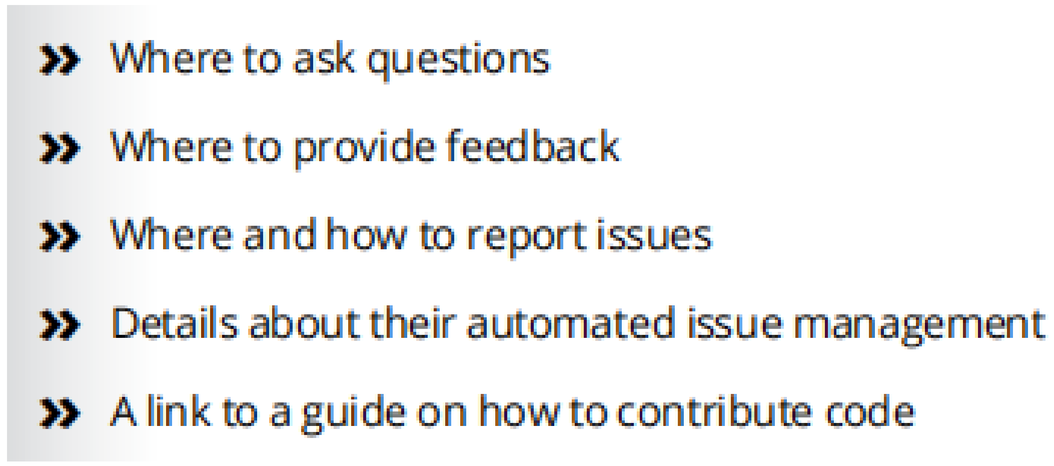
- I want to get involved in OSS in general and learn how to contribute.
- I want to help improve a project that I use in my day-to-day work.
- I want to support a project that does good in the world.
- I want to expand my skills by working in a technology I don't normally get to.
- I'm just bored and want to contribute to something cool.

3.SURVEYING A PROJECT FOR CONTRIBUTION

1. Reading the contributing guide
2. Reading the contributing code guide
3. Reading the code of conduct

3.SURVEYING A PROJECT FOR CONTRIBUTION

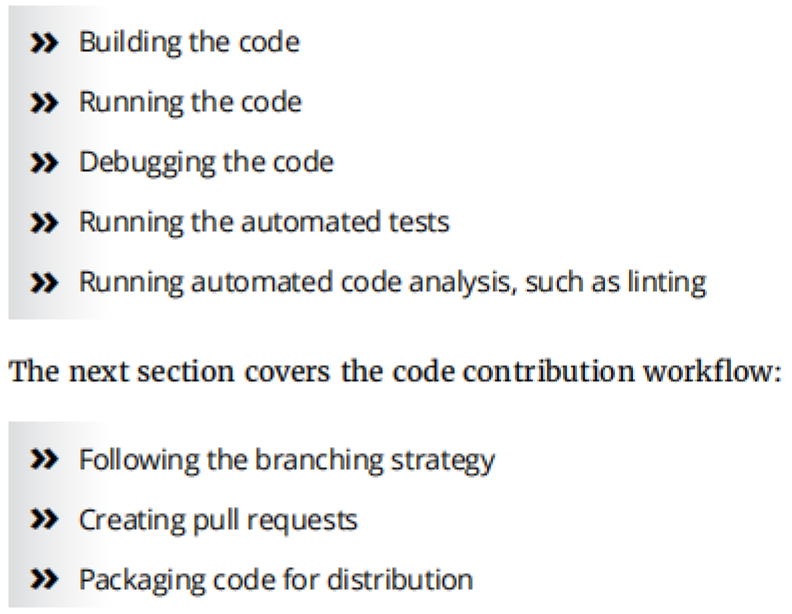
1. **Reading the contributing guide**



- Where to ask questions
- Where to provide feedback
- Where and how to report issues
- Details about their automated issue management
- A link to a guide on how to contribute code

3.SURVEYING A PROJECT FOR CONTRIBUTION

2. Reading the contributing code guide:



- Building the code
- Running the code
- Debugging the code
- Running the automated tests
- Running automated code analysis, such as linting

The next section covers the code contribution workflow:

- Following the branching strategy
- Creating pull requests
- Packaging code for distribution

3.SURVEYING A PROJECT FOR CONTRIBUTION

3. Reading the code of conduct

1. The projects now also include a **CODE_OF_CONDUCT.md** file.
2. This file lays out expectations for those who participate in the project.
3. It sets the behavioral norms for the project and a resolution process in cases where violations of the norm occur.
4. VS Code uses the Microsoft Open Source Code of Conduct located at **https://opensource.microsoft.com/codeofconduct.**
5. Adding a code of conduct to your own repository is easy.
6. If you add a new file on GitHub using the browser and name the file CODE_OF_CONDUCT.md, GitHub displays a code of conduct selection drop-down list with two choices: **the Contributor Covenant and the Citizen Code of Conduct.**

4. SETTING CONTRIBUTOR EXPECTATIONS.

1. They won’t fix every issue
2. They won’t merge every pull request
3. They don’t owe you anything

4. SETTING CONTRIBUTOR EXPECTATIONS.

1. They won’t fix every issue:

- Many, if not most, projects have limited resources, and their priorities may not

align with your priorities.

- It’s important to keep all that in mind when you file an issue.
- On the one hand, opening a good issue takes time and energy.
- A well_x0002_written issue that thoroughly describes a problem with clear steps to reproduce

the issue is very valuable to a project, so it’s understandable that people who write issues feel invested in them.

4. SETTING CONTRIBUTOR EXPECTATIONS.

2. They won't merge every pull request

- Submitting a pull request is the culmination of a lot of work.
- To submit a proper pull request, a developer had to spend the time to get the code working on their own machine, understand the code well enough to make a change, write the change, and then submit it.
- Being disappointed when the repository maintainers then reject the pull request is understandable.
- Remember, though, they're under no obligation to accept your pull request.
- Yes, you put a lot of time into the pull request, but they'll have to own the code change for the lifetime of the project.

4. Setting Contributor Expectations.

3. They don’t owe you anything

- One of the biggest challenges of being a maintainer of a popular open source project is the sense of entitlement expressed by many people who use the software
- In most cases, the maintainers are all volunteers working on the project on the side.
- Sometimes, they’re paid employees working on an open source project.
- But chances are, you are not the one paying them.