Style guide

In this article

Follow this guide to make sure GitHub's documentation stays consistent and follows clear patterns that our readers can understand.

Articles in the "Contributing to GitHub Docs" section refer to the documentation itself and are a resource for GitHub staff and open source contributors.

Note: These guidelines are specific to GitHub's documentation. For general style questions or guidance on topics not covered here, see the Microsoft Style Guide. For markup specific to source content on docs.github.com, see "Using Markdown and Liquid in GitHub Docs." For any questions about the GitHub brand, see our "GitHub Brand Guide."

The GitHub Docs approach to style

- Our style guide aims for simplicity. Guidelines should be easy to apply to a range of scenarios.
- Decisions aren't about what's right or wrong according to the rules of grammar or the style guide, but about what's best for our users. We're flexible and open to change while maintaining consistency.
- To scale the style guide as our team and documentation sets grow, and to create high-quality, meaningful content that serves users, we focus our attention on high-impact, high-value scenarios rather than attempting to comprehensively cover every style question.
- Consistency and grammatical correctness are important, but not as important as clarity and meaning.
- When making a style or structure decision, we consider the flow of information within the unit of content and the context of the information.
- When a question specific to help documentation isn't covered by the style guide, we think it through using these principles, then make a decision. If a reviewer asks about it, we're prepared to discuss the decision.

Audit log events

We document each of the events that may appear in the audit logs for each type of account: user, organization, and enterprise.

- "Security log events"
- "Audit log events for your organization"
- "Audit log events for your enterprise"

When writing the description for an audit log event, describe the event that took place in a way that applies to all versions, using past tense and passive voice. Do not begin the sentence with phrases that are already implied by the context of the article, such as "Triggered by."

- **Use**: The visibility of a repository was changed.
- **Use**: Secret scanning was enabled for all new repositories.
- **Avoid**: An organization owner disabled a two-factor authentication requirement for the organization.
- **Avoid**: Triggered when a user updates which repositories a codespace can access.

Callouts

Callouts highlight important information that customers need to know. We use standard formatting and colors for different types of callouts across doc sets.

Use callouts sparingly for high-value information - do not include general information, permissions, or prerequisites in callouts. Do not include more than two bullet points in a callout.

There are three types of in-content callouts: notes, warnings, and danger notices.

Formatting callouts

Each callout starts with text indicating the type of callout (e.g. **Warning**:) to orient the reader (whether accessing the site visually or with a screen reader) and helps every user gauge the importance and necessity of the information in the callout.

Notes are rendered in blue `{% note %}` tags.

- Notes provide useful information or reminders for the user, but the information is not required to follow. Notes may not be relevant or necessary to every user.
- Precede note content with `**Note:**`.

Warnings and danger notices are rendered in red `{% warning %}` tags.

- Warnings are potentially dangerous actions that a user should heed before continuing with a task. They are often non-optional steps.
  - Precede content with `**Warning:**`.
- Danger notices are dangerous actions that a user should exercise extreme caution before performing. They often involve the potential for data loss or other destructive actions.
  - Precede content with `**Danger:**`.

For more information on formatting callouts, see “Callouts” in "Using Markdown and Liquid in GitHub Docs."

Buttons

Landing pages and some articles have buttons that take people to relevant content in other articles or on other GitHub webpages. Buttons should be used when someone needs to navigate to another page to complete the task being described. For example, "Setting up a trial of GitHub Enterprise Cloud" has a button that takes people to the trial sign up page since that is the next step in the process of setting up a trial. The "Migrations documentation" landing page uses a button to direct people to the article that most people will need to read to start a migration.

If a button encourages people to navigate away from the GitHub Docs site, follow the call to action (CTA) button guidelines. If you want to include another type of button on a landing page or article, open an issue in the `docs-strategy` repository to share your use case for approval by the Content Strategy team.

Call to action (CTA) buttons

CTA buttons emphasize a link that we expect or encourage people to navigate to after reading an article or as part of completing the task that an article describes. CTAs should only take people to GitHub-owned domains. For example, the CTA in "Getting started with GitHub Copilot" links to the GitHub Copilot settings menu on GitHub.com.

Only include a CTA button if navigating to the link supports user needs. Do not use CTA buttons solely for marketing GitHub features or products. In the above example, someone who wants to try GitHub Copilot must navigate to the Copilot settings menu and would likely want to after reading the article. In contrast, even though someone might use Copilot as part of writing code that they then create a pull request for, we would not add a "Try GitHub Copilot" CTA to "Creating a pull request" since Copilot is not connected to the user needs of "Creating a pull request." Most people will create pull requests without using Copilot. But people visiting articles about getting started with Copilot are probably interested in trying Copilot if they are not already using it. So we add the CTA button to help people get where they are trying to go.

Style your CTAs using the following format.

```
<a href="https://github.com/DESTINATION/URL" target="_blank" class="btn btn-primary mt-3 mr-3 no-underline"><span>Try PRODUCT NAME</span> {% octicon "link-external" height:16 %}</a>
```

Code

Code blocks

Keep lines in code samples to about 60 characters, to avoid requiring readers to scroll horizontally in the code block. Locate explanatory text before the code block, rather than using comments inside the code block. See "Using Markdown and Liquid in GitHub Docs" for more information on the syntax and formatting of code blocks.

Within code blocks:

- Specify the language of the sample after the first code fence. For a list of all supported languages, see "Code languages" in the `github/docs` repository.
- Do not use HTML to style or markup a code block.
- Style any placeholders that people need to replace with their own values in all caps.
  - **Use:** `git checkout -b BRANCH-NAME`
  - **Avoid:** `git checkout -b <branch-name>`
- Only use `$` before the command itself if you’re showing the command’s output in the same block.
  - If you show a command and the command's output, do not make the code block copyable.
- If your code example includes `{` or `}` that should render, wrap that section in `{% raw %}` `{% endraw %}` to disable Liquid processing for that section.
  - **Use**:

    ```
    GITHUB_TOKEN: {% raw %}${{ secrets.GITHUB_TOKEN }}{% endraw %}
    ```

  - **Avoid**:

    ```
    GITHUB_TOKEN: ${{ secrets.GITHUB_TOKEN }}
    ```

- If your code example includes content that should be parsed, wrap that section in `<pre>` `</pre>` tags to parse rather than escape the content in the section.

Commands

Use inline code blocks to refer to short command names.

- **Use:** To check the status of a running cluster, use the `ghe-cluster-status` command.

Use command blocks for longer or more complex commands.

- **Use:** Enable maintenance mode according to your scheduled window by connecting to the administrative shell of any cluster node and running:

```
ghe-cluster-maintenance -s
```

Avoid inline links in command names.

Examples

When code examples refer to a larger file, show the relevant section of the file, so that users understand how to edit their own code in context.

- **Use:**

```
on:
  schedule:
    - cron:  "40 19 * * *"
```

- **Avoid:**

```
schedule:
  - cron:  "40 19 * * *"
```

File names and directory names

Use backticks to format references to file names and directory names in a monospaced font. If a file type generally follows a specific capitalization convention, such as all caps for README files, use the established convention.

- **Use:** In your `README.md` file, add info about your repository.
- **Use:** In your `.github/workflows/` directory, create the `example-workflow.yml` file.
- **Avoid:** In your *.github/workflows/* directory, create the `example-workflow.yml` file.
- **Avoid:** Delete the **example.js** file.

Indentation

In YAML examples, such as actions and workflow files, use two spaces to indent lines within nested lists and block sequences.

- **Use:**

```
    steps:
      - uses: actions/checkout@v4
      - name: Setup Python
        uses: actions/setup-python@v4
        with:
          python-version: ${{ matrix.python }}
```

To indent reusables, see `data/reusables/README.md`.

Scheduled workflows

Workflow runs are delayed when too many workflows run at once. Since many users copy code from GitHub Docs, we should use examples that guide users away from congested times.

- Do not use examples that run on the hour. (The Actions team reports that times at the start of the hour, especially UTC midnight, are disproportionately overloaded.)
- Do not use examples that run more frequently than necessary. For example, instead

of running every 5 minutes, consider if the example makes sense to run every 30 minutes instead.
- Use a different time for each example.

Emphasis

Use italics to emphasize words or parts of a sentence. Use emphasis sparingly for terminology or context that someone must be aware of to successfully complete the task that they're working on. Do not use italics to emphasize words that have other formatting applied such as all caps for placeholder text or bold for UI elements.

- **Use:** *fine-grained personal access tokens* have several security advantages over personal access tokens (classic) (classic).
- **Use:** *For types of packages other than containers*, to the right of the package version click **Delete**.
- **Avoid:** Next to ***Title***, add a descriptive label for your new key.

Footnotes

Avoid using footnotes where possible. Consider instead whether you could use a callout or present the information in another way. See some examples of alternatives to footnotes from NICE.org.uk.

If you must use footnotes, use Markdown-native footnotes (`[^1]`). Footnote markers will be hyperlinked to the footnote reference, which will be listed at the bottom of the page with a backlink to the marker.

Note that regardless of the identifier you use (letters, words), footnotes will render as sequential numbers.

	Mona	**Ursula**	**Paul**	**Davy Jones**[1]
Favorite pastime	Shipping code	Tricking mermaids[2]	Predicting sports	Haunting seafarers
Uses powers for good	Yes	No	Yes	No

```
| | Mona | Ursula | Paul | Davy Jones[^1] |
|---|---|---|---|---|
|Favorite pastime| Shipping code | Tricking mermaids[^2] | Predicting sports |
Haunting seafarers |
|Uses powers for good| Yes | No | Yes | No |
[^1]: Not to be confused with Davy Jones of The Monkees
[^2]: Also humans
```

Headers

Headers must adequately describe the content under them. Follow the same guidelines we use for writing titles. Use sentence casing for headers. Each header on a page must be unique.

Use H2 for headers, and H3 for subheaders. If the article has headers, the headers must start with an H2 level header and cannot skip header levels. There must be content between a header and subheader, such as an introduction. When referring to headers, surround the header name with quotation marks.

- **Use:** Under "User licenses," view your total licenses.

For more information, see the "Contents of a GitHub Docs article."

Images

We use static images including screenshots, diagrams, and graphs throughout the docs to complement textual information.

Do not use animated GIFs in the docs.

Alt text

Every image must include alt text providing a textual equivalent of the visual information.

- Express the core idea or meaning of the image, rather than describing it literally.
- Use 40–150 characters.
- End with a punctuation mark. This should generally be a period unless the alt text is describing an image of text that ends with other punctuation, such as a question mark or exclamation point.
- Don't start with "Image..." or "Graphic...". Screen readers say this automatically.
- Do begin with the *type* of graphic: "Screenshot of..." or "Diagram that shows..."
- Follow standard language used to describe UI elements in article text.
- Put multi-word titles, such as names of menu items, in double quotation marks (""). When using punctuation after multi-word titles, place it outside of the quotation marks so that the string in quotation marks exactly matches the title as it appears in context.
- If an area of the image is visually highlighted, describe how. This enables screen-reader users to understand and describe to a sighted friend/colleague what to look for from a visual language standpoint.

Alt text for screenshots

Alt text provides a short description of a screenshot's content to benefit people who cannot see it.

- Alt text only needs to include the most relevant elements of an image, not every detail.
- Alt text is not intended to provide instructions for using the GitHub interface. These should be included in accompanying article text.

Format

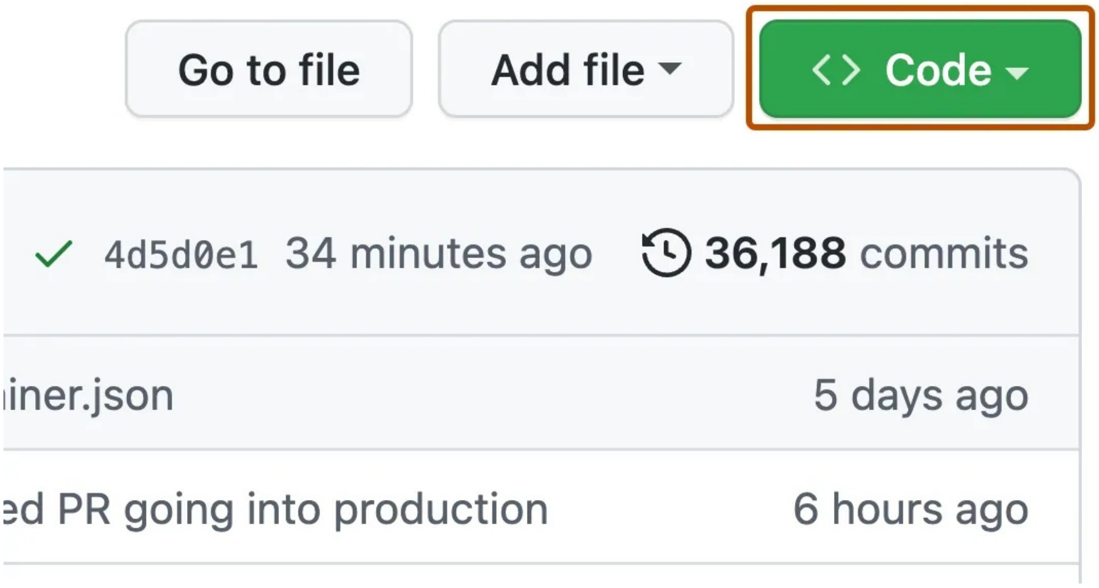
> Screenshot of the `Product name` + `UI element` shown. The `UI element` + `state of the element/controls`, and its `keyboard shortcut XYZ`, are outlined in dark orange.

- For `Product name`, use the GitHub product or feature name, such as "GitHub Actions" or "GitHub repository," rather than just "GitHub."
- Use a variable for the word `GitHub` as we do in running copy: `{% data variables.product.prodname_dotcom %}`
- Describe UI elements consistently with written documentation.
- Be flexible with word order when needed for clarity.
  - For example, write "Screenshot of the Debug menu in Visual Studio Code..." rather than "Screenshot of the Visual Studio Code Debug menu...," to avoid multiple nouns in a row.

Examples

> Screenshot of the GitHub committers by repository table. The horizontal kebab icon and "Download CSV report" button are outlined in dark orange.

> Screenshot of file options in a GitHub repository. A button with an arrow indicating a dropdown menu, labeled "Code," is outlined in dark orange.



Alt text for diagrams and graphs

Explain the information conveyed in the diagram or graph in text on the page.

Use alt text to express the core idea of the image, without duplicating the webpage text.

Example

> Diagram showing a five-step process by which a GitHub Actions runner can be automatically added to named classes of runners and then requested by specific jobs.

For example, see accompanying explanation of this diagram in the Actions documentation.

Alt text for images of command-line interfaces

Do not use screenshots of command-line interfaces to convey commands and their output. Instead, directly provide the commands a user should use. For more information, see the "Commands" section of the style guide.

When using a screenshot of a command-line interface to show user interface elements, follow standard alt text guidelines for screenshots.

File names for images

Be descriptive when naming image files: include the name, action, and UI element in the file name. Mirror product language. Use kebab case. Do not use Liquid conditionals in file names. If replacing an image, use the exact file name.

- **Use:** `data-pack-purchase-button.png`
- **Avoid:** `purchase_button.png`
- **Avoid:** `purchase-button-for-admins.png`

Screenshots

To learn about creating and versioning images, see "Creating and updating screenshots."

Diagrams

To learn about creating diagrams, see "Creating diagrams for GitHub Docs."

Inclusive language

As home to the largest developer community in the world, GitHub is committed to promoting diversity and inclusion in every aspect of what we do. All of our documentation is inclusive and respectful of our audience, which consists of people in widely varying circumstances from all over the planet. When we write our documentation, we use words that are inclusive, anti-racist, and accessible.

Individual words might be small, but together they can create community, belonging, and equity. Be empathetic in all word and style choices. Be accurate when referring to people and communities.

Use	Avoid
Allowlist	Whitelist
Denylist	Blacklist
Default/Main branch	Master branch

Resources about inclusive language

The Microsoft Style Guide offers resources on bias-free communication, accessibility terms, and writing for all abilities:

- Bias-free communication
- Writing for all abilities
- Accessibility terms

More resources for learning about inclusive and accessible language and style:

- 18F Content Guide on Inclusive Language
- MailChimp Content Style Guide:
  - Writing About People
  - Writing for Accessibility
- Readability Guidelines
- Conscious Style Guide

Keyboard shortcuts

For presenting keyboard shortcuts, follow the Microsoft Style Guide, **except for the following differences**:

- Use the HTML `<kbd>` tag for each individual key.
  - **Use:** `<kbd>Command</kbd>+<kbd>B</kbd>`
  - **Avoid:** `Command+B`

- Use full words instead of symbols for Apple modifier keys.
    - **Use:** `Command`
    - **Avoid:** `⌘`
- Use symbols for keys of special character, not full words.
    - **Use:** `.`, `,`, and `→`.
    - **Avoid:** `Period`, `Comma`, and `Right arrow`.

Usage highlights

Below are some usage highlights for how we present keyboard shortcuts in our documentation:

- The basic syntax is to show keys with `+` between key combinations, without any spaces.
    - **Use:** `<kbd>Command</kbd>+<kbd>B</kbd>`, which is rendered as `Command`+`B`.
    - **Avoid:** `<kbd>Command</kbd> + <kbd>B</kbd>` or `<kbd>Command + B</kbd>` which are rendered as `Command` + `B` or `Command + B`.
- Always capitalize letter keys for general references and keyboard shortcuts.
    - **Use:** `Command`+`B`
    - **Avoid:** `Command`+`b`.
- Use the correct modifier keys for the each operating system.

  Note: Windows and Linux have `Ctrl` abbreviated, whereas on Mac it is spelled in full: `Control`.

    - For Windows and Linux:
        - **Use:** `Ctrl`, `Alt`.
        - **Avoid:** `Control`
    - For Mac:
        - **Use:** `Command`, `Option`, `Control`.
        - **Avoid:** `Cmd`, `⌘`, `Opt`, `⌥`, `Ctrl`, `^`
- Don't confuse key combinations with keys in a sequence.
    - `Command`+`B` indicates that the user should hold down the `Command` key and press the `B` key.
    - `G` `I` indicates that the user should press the `G` key, then press the `I` key.
- When describing a keyboard shortcut for multiple operating systems, append the operating system in brackets after the shortcut. Describe the Mac shortcut first, then Windows/Linux.
    - **Use:** `<kbd>Command</kbd>+<kbd>B</kbd> (Mac) or <kbd>Ctrl</kbd>+<kbd>B</kbd> (Windows/Linux)`, presented as:

      `Command`+`B` (Mac) or `Ctrl`+`B` (Windows / Linux)

    - **Avoid:** `<kbd>Ctrl</kbd>+<kbd>B</kbd> or <kbd>Command</kbd>+<kbd>B</kbd>`, presented as:

      `Ctrl`+`B` or `Command`+`B`

Licensed content

GitHub Docs is licensed under a CC-BY license. If you reuse or modify licensed content in an article, you must make sure that the license is compatible and properly attributed.

Do not create reusables for license attributions. We must use the exact license a project is licensed under, so any attributions must be accurately written for the articles that they appear in.

If you are unsure of the legality of reusing any content, contact legal. If you are adding content with a license that is not listed below, you must receive a legal review before you can publish the content.

Attributing MIT-licensed content

If we reuse or modify content under an MIT license, we must attribute the MIT license where the content appears.

At the end of the article containing MIT-licensed content

- Create a header titled `Legal notice`
- Attribute where the content comes from and that it is licensed under the MIT license. Include a link to the project
- Paste the full text of the MIT license from the project that you are attributing in a codeblock

Example MIT license attribution

This text is only an example. Always use the license text from the project you are attributing.

~~~
## Legal notice

Portions have been adapted from [PROJECT](/LINK/TO/PROJECT) under the MIT
license:

```
MIT License

Copyright YEAR COPYRIGHT-HOLDER

Permission is hereby granted, free of charge, to any person obtaining a copy of
this software and associated documentation files (the "Software"), to deal in the
Software without restriction, including without limitation the rights to use,
copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the
Software, and to permit persons to whom the Software is furnished to do so,
subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS
FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN
AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION
WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
```
~~~

Linebreaks

For plain text, use linebreaks to separate paragraphs in the source (two consecutive linebreaks), rather than to create visual space in the source. Avoid unneeded linebreaks, especially in lists.

Links

Introduce links consistently using a standard format that clearly indicates where we're linking.

For any link that points to another GitHub Docs page, use the special keyword `AUTOTITLE`. See details in the content markup reference.

Usage examples:

- For links to other pages: `For more information, see "[AUTOTITLE](/PATH/TO/PAGE)."`
- For links to sections in other pages: `For more information, see "[AUTOTITLE](/PATH/TO/PAGE#SECTION-LINK)."`
- For links to a page with a tool selected: `For more information, see the TOOLNAME documentation in "[AUTOTITLE](/PATH/TO/PAGE?tool=TOOLNAME)."`

Same-page section links do **not** work with `AUTOTITLE`. Instead, type out the full header text: `For more information, see "[HEADER-TITLE](#SECTION-LINK)."`

For links to external documentation, type out the full article name: `For more information, see [PAGE-TITLE](https://some-docs.com/PATH/TO/PAGE) in the XYZ documentation.`

Do not include quotation marks within a hyperlink.

Some best practices for using links:

- Links should be meaningful and provide high value to the user's journey—link out carefully.
- Move links that are helpful but not necessary to an article's further reading section.
- Do not repeat the same link more than once in the same article or under the same H2 header.
- Do not include the `apiVersion` query parameter in REST links unless you need to link to a specific calendar version of the REST docs. (This should be a rare occurrence.)

For accessibility and readability, avoid inline or midsentence links.

- **Use:** `OAuth2 tokens can be acquired programmatically for applications that are not websites. For more information, see "[AUTOTITLE](/apps/creating-github-apps/authenticating-with-a-github-app/generating-a-user-access-token-for-a-github-app)" and "[AUTOTITLE](/apps/oauth-apps/building-oauth-apps/authorizing-oauth-apps)."`
- **Avoid:** `Read [more about OAuth2.](/apps/building-integrations/setting-up-and-registering-oauth-apps/) Note that OAuth2 tokens can be [acquired programmatically](/enterprise-server@2.22/rest/reference/oauth-authorizations/#create-a-new-authorization), for applications that are not websites.`

For more information on links and accessibility, see "Links" in the Readability Guidelines project.

Links between versions

Sometimes, you need to link from one version of GitHub Docs to another. When you want to link to a different version of the *same* page, you should use the `currentArticle` property.

For example, the Free, Pro, & Team version of "Managing the publication of GitHub Pages sites for your organization" might link to the GitHub Enterprise Cloud version of the same article like this:

```
You can choose to allow or disallow the publication of GitHub Pages sites.

Organizations that use GGitHub Enterprise Cloud can choose to allow publicly
published sites, privately published sites, both, or neither. For more
information, see [the GitHub Enterprise Cloud documentation](/enterprise-
cloud@latest//contributing/style-guide-and-content-model/style-guide).
```

To link to a different article in a different version, use this format:

```
For more information, see "[ARTICLE TITLE](/)" in the VERSION documentation.
```

To link to the same article in a different version, use this format:

```
For more information, see [the VERSION documentation]
(/VERSION//contributing/style-guide-and-content-model/style-guide).
```

To link to a specific version, you must include the version in the path (e.g., `/enterprise-cloud@latest//contributing/style-guide-and-content-model/style-guide`).

Links to specific sections of articles

When we link to specific sections of articles, we want to make sure the link is descriptive enough so that someone knows they are in the correct spot after following a link.

To link to a specific header in the same article, use this format:

```
For more information, see "[HEADER TITLE](#HEADER-TITLE)."
```

To link to a specific header in a different article, use this format:

```
For more information, see "[AUTOTITLE](PATH-TO-ARTICLE#HEADER-TITLE)."
```

To link to two or more specific headers in a different article, use this format:

```
For more information, see "[HEADER-TITLE-1](PATH-TO-ARTICLE#SECTION-LINK-1)" and
"[HEADER-TITLE-2](PATH-TO-ARTICLE#SECTION-LINK-2)" in "ARTICLE-TITLE."
```

Links to a specific tool

When we link to content with a specific tool selected, we want to make sure that someone knows that they will be looking at content relevant to a specific tool even if they do not view the tool switcher tabs in the article.

```
For more information, see the TOOLNAME documentation in "[ARTICLE TITLE]
(/PATH/TO/ARTICLE?tool=TOOLNAME)."
```

Links to learning paths

Use this format to link to a learning path.

```
For more information, follow the "[LEARNING PATH TITLE](/)" learning path.
```

Links to external resources

When linking to an external site, choose the most useful resource for the context of the link - you can link to a whole site if it's a general reference or to a specific page if that would be more helpful.

It's not necessary to link to an external product’s website when we mention an external product.

Lists

Capitalize the first letter in each line of a list. Use periods at the end of lines in a list only if the line contains a complete sentence.

When writing a list of items that consist of primary and secondary text, such as a `term` and its definition, use a colon delimiter. The secondary text should be capitalized as if it was the beginning of the line. For example:

- `foo`: Something that provides bar.
- `bar`: Something provided by foo.

Formatting unordered lists:

- If the order of items in the list is not important, alphabetize the list items.
- If the order is important, then order the list by the importance to the reader (for example, moving from broadest audience and applicability to a more specialized audience).

When introducing a list, avoid phrasing like “the following” or “these”, terms which are difficult to localize. Instead, be descriptive, yet general enough to allow a list to scale or change without having to update the description.

Placeholders

Style any placeholder text in all caps. If a placeholder is multiple words, connect the words with dashes (kebab-case). If you use a placeholder, explain what someone might replace it with. This helps people modify examples to fit their needs and helps identify placeholders for people who use assistive technology.

Use:

- In the following example, replace YOUR-REPOSITORY with the name of your repository. `git init YOUR-REPOSITORY`
- Click **Add USERNAME.** Where USERNAME is the username of the person you want to add.

Avoid:

- `git init your repository`
- `git init <your-repository>`
- Click **Add *username*.**

Procedural steps

Procedures give readers a set of sequential steps to follow to complete a task. Always use numbered lists for procedures. Give readers all of the prerequisites or conceptual information they’ll need to complete the task before the procedure, rather than including it within a specific step.

Each step must include an action. You can also choose to include whether a step is optional, explain the reason or result of the step, and orient the reader by describing the location of the action, before guiding them to complete the action.

Use a consistent order to present information within each step.

1. If the step is optional, indicate that first.
2. When needed for clarity, or to reinforce the severity of a destructive or confusing action, explain the reason for or result of the step.
3. Describe the location the user will find the action in.
4. Action.

Use: Optionally, to `REASON`, in `LOCATION`, take `ACTION`.

Examples:

- Click **Payment information**.
- Under your organization name, click **Settings**.
- To confirm your change, click **Remove credit card**.
- Optionally, to see your plan's details, click **Show details**.
- Under "GitHub Sponsors", to the right of the sponsored open source contributor, click [down arrow octicon] next to your sponsored amount, then click **Change tier**.

Product names

Use full product names. Do not abbreviate or shorten product names unless directly reproducing content from the product (e.g. UI copy or API responses).

Use product name variables to render product names - do not write product names in plain text. This makes product name changes easier to implement across the site and avoids typos in our product names. For more information about product name variables, see "Reusables and variables" in this document and the data directory of the github/docs repository.

Product names are always singular.

- **Use:** GitHub Actions helps you automate your software development workflows.
- **Avoid:** GitHub Actions help you automate your software development workflows.

Take care to distinguish between product names and product elements.

Product	Element
GitHub Actions	an action
GitHub Codespaces	a codespace
GitHub Packages	a package
GitHub Pages	a GitHub Pages site

Product-specific conventions

This section describes additional conventions that are specific to GitHub products.

GitHub Actions

Reusables for first-party actions

Code examples that use first-party actions must use the respective reusable for that action. This makes action version updates (e.g. from `v1` to `v2`) easier to manage for products like GitHub Enterprise Server, which might not have the same action version available until a future GitHub Enterprise Server release.

Actions reusables are located in `/data/reusables/actions/` and have a file name like `action-<action_name>.md`

For example, to use the `actions/checkout` action in an example, use its reusable:

```
steps:
  - name: Checkout
    uses: actions/checkout@v4
```

For GitHub Docs purposes, a first-party action is any action that has the `actions/`, `github/` or `octo-org/` prefix. For example, this is a first-party action:

```
steps:
  - uses: actions/checkout@main
```

Disclaimers for third-party actions

Code examples that use third-party actions must include the following disclaimer as part of the code block:

```
# This workflow uses actions that are not certified by GitHub.
# They are provided by a third-party and are governed by
# separate terms of service, privacy policy, and support
# documentation.
```

To insert this disclaimer, use the `{% data reusables.actions.actions-not-certified-by-github-comment %}{% endnote %}` reusable. If the code block is indented, you must use `indented_data_reference` along with the reusable. For example:

```
{% raw %}{% indented_data_reference reusables.actions.actions-not-certified-by-
github-comment spaces=4 %}
```

For GitHub Docs purposes, a third-party action is any action that doesn't have the `actions/`, `github/` or `octo-org/` prefix. For example, this is a first-party action:

```
steps:
  - uses: actions/checkout@main
```

This is an example of a third-party action:

```
steps:
    - uses: google-github-actions/setup-
gcloud@1bee7de035d65ec5da40a31f8589e240eba8fde5
```

Examples:

- See the code block in "[Publishing to package registries]"

Pinning version numbers to SHA

Code examples that use third-party actions must always pin to a full length commit SHA, instead of the version number or branch:

```
steps:
    - uses: google-github-actions/setup-
gcloud@1bee7de035d65ec5da40a31f8589e240eba8fde5
```

For GitHub Docs purposes, a third-party action is any action that doesn't have one of the following prefixes: `actions/`, `github/`, and `octo-org/`. For example, this is a first-party action:

```
steps:
  - uses: actions/javascript-action@main
```

For more information, see "Using SHAs"

Codespaces

When referring to the product Codespaces, always include "GitHub", except in these circumstances:

- In the `shortTitle` front matter.
- In subheadings within an article, if "Codespaces" has already been used anywhere in the article prior to the subheading.

Variables: `{% data variables.product.prodname_github_codespaces %}` ("GitHub Codespaces") and `{% data variables.product.prodname_codespaces %}` ("Codespaces").

When referring to instances of remote working environments created with this technology, refer to these as "codespaces" (lowercase c). For example, "to delete your codespace" or "to list your codespaces."

Always use "dev container" (or, where clarification is needed, its longer form "development container") and not "devcontainer" (one word), except in file/path names. The single word could form could be considered a brand, which we want to avoid, and we also want to be consistent with the two-word form used in the Visual Studio Code documentation.

Use "development container configuration files" to refer to all of the files in the `.devcontainer` directory (plus the `.devcontainer.json` if that's being used rather than `devcontainer.json` in the `.devcontainer` directory). Don't refer to these as "development container files" or "devcontainer files" to avoid this being taken as referring to `devcontainer.json` files. "Development container configuration files" refers to all of the files that can be used to configure a dev container, including `Dockerfile` and `docker-compose.yml` files. Don't use "the development container configuration file" (singular) when referring specifically to a `devcontainer.json` file. Instead refer to this file by its name.

GitHub Advanced Security (GHAS)

Use the terms `licenses` and `active committers` when you refer to GitHub Advanced Security billing.

We used to use the term `seats` to describe the number of accounts that can use GitHub Advanced Security in an enterprise. People can be confused by the term `seats`, so we removed this term from GitHub.com in autumn 2022 and versions from GHES/GHAE 3.7 onward do not use it.

Personal access tokens

GitHub has two types of personal access tokens:

- Fine-grained personal access tokens: Offer granular control over repository access and permissions
- Personal access token (classic): Use scopes and grant access to all repositories that the token owner can access

You should use variables to refer to these types of tokens, as well as to personal access tokens in general:

- Use `{% data variables.product.pat_generic %}` or `{% data variables.product.pat_generic_caps %}` to refer to personal access token in general. Use `{% data variables.product.pat_generic_title_case %}` if the phrase should be in title case ("Personal Access Token") in order to match UI text.
- Use `{% data variables.product.pat_v2 %}` or `{% data variables.product.pat_v2_caps %}` to refer to fine-grained personal access tokens.
- Use `{% data variables.product.pat_v1 %}`, `{% data variables.product.pat_v1_plural %}`, `{% data variables.product.pat_v1_caps %}`, or `{% data variables.product.pat_v1_caps_plural %}` to refer to personal access token (classic).

For more information about GitHub's personal access tokens, see "Managing your personal access tokens."

Punctuation

When referencing a UI element in quotation marks, place any subsequent punctuation outside of the quotation marks so that the label in quotation marks exactly matches the UI.

- **Use:** Under "User licenses", view your total licenses.

Otherwise, follow standard American English punctuation rules. For more guidance, see “Punctuation” in the Microsoft Style Guide.

Release notes

A set of release notes on GitHub Docs tell readers about administrator- or user-facing changes to a versioned release of a product like GitHub Enterprise Server (GHES). Release notes appear in the Release notes.

A good release note is a few sentences that sequentially answer the reader's questions about the change. For more information, see Release note content type.

Each release note in a set describes one of the following changes.

- Features: brand-new behavior or functionality
- Security fixes: fixes to flaws or unexpected behavior that have security implications
- Bug fixes: fixes to flaws or unexpected behavior
- Changes: notable changes to past behavior
- Known issues: issues that GitHub has identified, but cannot or has not yet prioritized
- Deprecations: removal of a feature or behavior
- Errata: correction to inaccurate release note or documentation

You can also review guidelines for updating release notes in Adding or updating a release note.

Features

A release note for a feature summarizes brand-new behavior. Generally, notes for features are only part of feature releases.

Writing release notes for features

A release note for a feature answers the following questions.

1. Does this new functionality apply to me, with my role or access?
2. What need does the functionality satisfy?
3. What is the functionality?
4. If applicable, where can I read more about the functionality?

> *AUDIENCE* (**1**) can *DESCRIPTION OF NEED* (**2**) by *DESCRIPTION OF FEATURE'S USE* (**3**). For more information, see "*ARTICLE TITLE*" (**4**).

- Categorize each feature in a section, under a feature heading.
- Write in the present tense.
- To reduce repetition and unnecessary words, "now" is usually implied.
- To clarify actors and impact, avoid passive language when possible.

Examples of feature release notes

- > Site administrators can increase the security of the Management Console by configuring the rate limit for sign-in attempts, as well as the lockout duration after exceeding the rate limit. For more information, see "Configuring rate limits."
- > Enterprise owners can control where users can fork repositories. Forking can be limited to preset combinations of organizations, the same organization as the parent repository, user accounts, or everywhere. For more information, see "Enforcing repository management policies in your enterprise."
- > Users can create files with geoJSON, topoJSON, and STL diagrams and render the diagrams in the web interface. For more information, see "Working with non-code files."

Security fixes

A release note for a security fix summarizes a change that mitigates or prevents exploitation of a security-related issue in the product. Generally, notes for security fixes are only part of patch releases.

Writing release notes for security fixes

A release note for a security fix answers the following questions.

1. If available, what is the NVD vulnerability severity rating for the vulnerability that's fixed?
2. What is the attack that an attacker could accomplish by exploiting the vulnerability?
3. What type of vulnerability is exploitable?
4. If available, what is the vulnerability's CVE identifier, pending or active?

5 Did someone report the vulnerability via the GitHub Bug Bounty program?

> *SEVERITY* (**1**): An attacker could *DESCRIPTION OF IMPACT* (**2**) by *DESCRIPTION OF EXPLOIT* (**3**). GitHub has requested CVE ID *CVE-####-#####* (**4**) for this vulnerability, which was reported via the GitHub Bug Bounty program (**5**).

Examples of release notes for security fixes

- > **MEDIUM**: An attacker could cause unbounded resource exhaustion on the instance by making parallel requests to the Markdown REST API. To mitigate this issue, GitHub has updated CommonMarker. GitHub has requested CVE ID CVE-2022-39209 for this vulnerability.
- > **MEDIUM**: An attacker could embed dangerous links in the instance's web UI because pull request preview links did not properly sanitize URLs. This vulnerability was reported via the GitHub Bug Bounty program.

Base image and package updates

We also include base image and dependent package updates in the "Security fixes" section, since these updates often address security issues. We consolidate all of these updates in the following note.

> Packages have been updated to the latest security versions.

Bug fixes

A release note for a bug fix describes a correction to an undesired or otherwise unexpected behavior. Generally, notes for bug fixes are only part of patch releases.

Writing release notes for bug fixes

A release note for a bug fix answers the following questions.

1 Did the behavior affect me, with my role or access?

2 What behavior would the reader experience prior to the fix?

> *AUDIENCE* (**1**) *DESCRIPTION OF BEHAVIOR* (**2**).

- Because the bug is now fixed, write in the past tense.
- Language like "fixed a bug..." or "fixed an issue..." is implied and unnecessary.
- To reduce repetition and unnecessary words, "now" is usually implied.
- To clarify actors and impact, avoid passive language when possible.

Examples of release notes for bug fixes

- > After a user imported a repository with push protection enabled, the repository was not immediately visible in the security overview's "Security Coverage" view.
- > On an instance with GitHub Actions enabled, a workflow job for GitHub Actions would not start if a matching runner group was unavailable when the job was initially queued, even if a matching runner group became available after the job entered the queue.
- > Commands that site administrators ran via SSH on any of the instances nodes were not logged in `/var/log/ssh-console-audit.log`.

Changes

A release note for a change describes a notable, but minor change to existing behavior. Notes for changes answer the following questions.

Writing release notes for changes

A release note for a change answers the following questions.

1. Did the behavior affect me, with my role or access?
2. If the change solves or avoids a problem, what's that problem?
3. What's the new behavior?
4. If relevant, what was the behavior before the change?

> *AUDIENCE* (**1**) / *DESCRIPTION OF PROBLEM CHANGE SOLVES* (**2**) *DESCRIPTION OF NEW BEHAVIOR* (**3**) *DESCRIPTION OF OLD BEHAVIOR* (**4**).

- Because the change applies to the release in question, write notes for changes in the present tense.
- To reduce repetition and unnecessary words, "now" is usually implied.
- To clarify actors and impact, avoid passive language when possible.
- Often, the audience is implied.
- If useful, include relevant links to GitHub Docs.

Examples of release notes for changes

- > On an instance with a GitHub Advanced Security license, users who author custom patterns for secret scanning can provide expressions that must or must not match that are up to 2,000 characters. This limit is an increase from 1,000 characters.

- > For administrators who need to review or modify SAML mappings, the default path for output from `ghe-saml-mapping-csv -d` is `/data/user/tmp` instead of `/tmp`. For more information, see "Command-line utilities."

- > To avoid intermittent issues with the success of Git operations on an instance with multiple nodes, GitHub Enterprise Server checks the status of the MySQL container before attempting a SQL query. The timeout duration has also been reduced.

Known issues

A release note for a known issue describes an issue that GitHub has identified, but cannot or has not yet prioritized.

Writing release notes for known issues

A release note for a known issue answers the following questions.

1. Does the behavior affect me, with my role or access?
2. If the change solves or avoids a problem, what's that problem?
3. What are any error messages or other recognizable UI elements that appear?

4 Do I need to act? If so, what should I do?

> *AUDIENCE* (**1**) *DESCRIPTION OF ISSUE* (**2**) *DETAILS OF BEHAVIOR* (**3**) *NEXT STEPS* (**4**).

- To clarify actors and impact, avoid passive language when possible.
- To reduce repetition and unnecessary words, "now" is usually implied.
- If useful, include relevant links to GitHub Docs.
- Known issues are also a type of content on GitHub Docs. For more information, see "Troubleshooting content type." If useful, write or link to more in-depth and contextually relevant content in the docs.

Examples of release notes for known issues

- > After a user enables the option for a repository to allow users with read access to create discussions, the feature is not enabled.
- > After an administrator begins a configuration run, a `No such object error` may occur during the validation phase for the Notebook and Viewscreen services. This error can be ignored as the services should still correctly start.

Deprecations

A deprecation release note summarizes a behavior or feature that GitHub has removed or plans to remove. Generally, notes for deprecations are only part of feature releases.

Writing release notes for deprecations

A release note for a deprecation answers the following questions.

1 Does this existing functionality apply to me, with my role or access?

2 What is the functionality that's being deprecated?

3 If applicable, what replaces the deprecated functionality?

4 If applicable, where can I read more?

> *AUDIENCE* (**1**) *DESCRIPTION OF DEPRECATED FUNCTIONALITY* (**2**) *REPLACEMENT FUNCTIONALITY* (**3**) For more information, see "*ARTICLE TITLE*" (**4**).

- Notes are in the present tense, or the future tense for upcoming changes. If applicable, specify the upcoming release where the deprecation will occur.
- To reduce repetition and unnecessary words, "now" is usually implied.
- To clarify actors and impact, avoid passive language when possible.
- Categorize each feature in a section, under a feature heading.

Examples of release notes for deprecations

- > **Upcoming deprecation**: In GitHub Enterprise Server 3.8 and later, to ensure instance security, unsecure algorithms will be disabled for SSH connections to the administrative shell.
- > Commit comments, which are comments that users add directly to a commit outside of a pull request, no longer appear in the pull request timeline. Users could not reply to or resolve these comments. The Timeline events REST API and the GraphQL API's `PullRequest` object also no longer return commit comments.

Errata

Errata corrects inaccurate information previously published in the release notes or documentation for a release.

Writing errata

Errata answers the following questions.

1. If applicable, which section of the release notes or content on GitHub Docs was affected?
2. Did the incorrect information apply to me, with my role or access?
3. What did the release note or documentation describe that was incorrect?
4. When was the errata published?

> *CONTENT* (**1**) incorrectly indicated that *AUDIENCE* (**2**) can *SUMMARY OF INACCURATE INFORMATION* (**3**). [Updated: *PUBLICATION DATE* **4**]

- Format the publication date according to the guidance in Adding or updating a release note.

Example of errata

- > "Features" incorrectly indicated that users of the GitHub Advisory Database can see advisories for Elixir, Erlang's Hex package manager, and more. This feature is unavailable in GitHub Enterprise Server 3.7, and will be available in a future release. [Updated 2023-06-01]

Adding or updating a release note

To signal to readers that you've added or changed a note, or to indicate the publication date of errata, append a datestamp in the format "[Updated: YYYY-MM-DD]".

Reusables and variables

Use reusable strings for individual nouns (e.g. product names) or for complete sentences or paragraphs. Sentence fragments and phrases should not be contained in reusable strings as they can cause problems when content is localized. For more information, see the data directory in the github/docs repository, "Creating reusable content", and the "Product names" section of this document.

Sectional TOCs

If a section of an article uses `H3` or `H4` headers to further divide the content and only some of the content is relevant to a reader, you can use a sectional table of contents (TOC) to help readers identify and navigate to the information that is most relevant to them. For example, in "Streaming the audit log for your enterprise" people will probably only set up audit log streaming for one provider, so the sectional TOC in "Setting up audit log streaming" allows people to select their provider and navigate to the relevant content without reading the entire section.

Do not add a sectional TOC if `H3` or `H4` headers are used only to group content and all information could be of relevance to a reader. For example, in "About authentication for your enterprise," people should read and consider each section as it relates to their enterprise. We do not include a sectional TOC in this article because people should be reading through each section, not picking and choosing between them. Adding a sectional TOC would also force people who use screenreaders or other adaptive technology to tab and scroll through more headers before finding what they need.

Format sectional TOCs as a list. Include all subsections in the order that they appear in the article and refer to them using the full header title.

Sectional TOCs must be introduced with a sentence or paragraph that helps people understand how the content is organized and select the section that is most relevant to them. Do not include a sectional TOC directly beneath a header.

Example of sectional TOCs

```
## Setting up the application

Set up your application according to your operating system.

- [Setting up for macOS](#setting-up-for-macOS)
- [Setting up for Windows](#setting-up-for-windows)
- [Setting up for Linux](#setting-up-for-linux)

### Setting up for macOS

TEXT

### Setting up for Windows

The application is supported for all versions of Windows, but the set up steps
differ.

- [Windows 98](#windows-98)
- [Windows Vista](#windows-vista)
- [Windows 11](#windows-11)

#### Windows 98

TEXT

#### Windows Vista

TEXT

#### Windows 11

TEXT

### Setting up for Linux

TEXT
```

Tables

Tables are added to the GitHub Docs using Markdown. Because tables can be challenging to read and maintain, make sure that the data in a table is best represented in a table and not another format, like a list, before creating a table. Every row in a table must begin and end with a pipe, `|`.

Use tables only for presenting tabular information

Tables work best for presenting tabular data, such as information that needs to be compared or values with multiple attributes. Do not use tables for simple lists - see the

"Lists" section of this document.

Avoid describing table data

A table’s data and why it is important should be clear from any preceding content, the column headers, and (if needed) the row headers. Avoid unneeded descriptions of the data in a table. If the data in a table is unclear without a lengthy description, consider if your table needs row headers or if the information would be better communicated in a different way.

For example, in "Autoscaling with self-hosted runners," a table comparing the features between two supported autoscaling solutions is introduced with the sentence `Each solution has certain specifics that may be important to consider.` The article does not describe any of the different features that are compared because that information is clearly communicated by the table.

- **Use:** "Different size limits per repository apply depending on your GHES version."
- **Avoid:** "The first row of the table shows the information for GitHub Enterprise Cloud. The second row shows the information for GitHub Enterprise Server."
- **Avoid:** “The table below shows what kind of migration data is exported.”

Use proper markup for row and column headers

Tables in which the first column describes the data values in the table (but is not data itself) need to be marked up with row headers. This is important for assistive technology to understand relationships between cells.

For example in the following table, in order to make sense of the "Yes" and "No" values in the table, you need to know both the column header (role) and row header (permission).

Organization permission	Owners	Members	Moderators	Billing managers	Security managers
Create repositories	Yes	Yes	Yes	No	Yes
View and edit billing information	Yes	No	No	Yes	No
Invite people to join the organization	Yes	No	No	No	No

To add row headers for a Markdown table, wrap the table in the Liquid tags `{% rowheaders %} {% endrowheaders %}`. For more information about using row headers, see "Using Markdown and Liquid in GitHub Docs."

Include a value for every cell

Every cell in a table must contain a value. If the table has row headers, the first cell (cell A1) can be empty.

If there is no data, use "None" or "Not applicable". Do not use "NA" or "N/A".

Use clear, consistent symbols and labels

For tables that use symbols:

- Populate all cells. For example in a permissions table, do not mark only the cells for things that require a permission.
- Use octicons or SVG. Do not use emoji. For more information about octicons, see "[AUTOTITLE]/contributing/syntax-and-versioning-for-github-docs/using-markdown-and-liquid-in-github-docs#octicons)."
- Use a check mark for affirmative values ("Yes", "Required", "Supported") and a cross for negative values ("No", "Optional", "Unsupported").
- Use `aria-label` to describe the meaning of the symbol, not its visual characteristics. For example, "Required", not "Check mark icon".

Where table data is not truly binary (every value is either "Yes" or "No", for example), text values may be needed in addition to, or instead of, symbols. For example on the page "About GitHub Support", some features are marked as "Available to purchase".

Use footnotes sparingly

See "Footnotes."

Align table content consistently

All columns in a table should be left-aligned, except for columns containing only octicons which should be center-aligned. If a column contains both text and octicons, use center alignment.

Table content is left-aligned by default. Use Markdown table formatting, colons (`:`) to either the right or left of the dashes in the header row, to specify the alignment of each column. Read "Organizing information with tables" for more information.

The following example shows part of a table from "Configuration options for the dependabot.yml file."

Option	Required	Security Updates	Version Updates	Description
`package-ecosystem`	✓	✕	✓	Package manager to use
`directory`	✓	✕	✓	Location of package manifests
`schedule.interval`	✓	✕	✓	How often to check for updates

The table is generated with the following alignment syntax.

```
| Option              | Required | Security Updates | Version Updates |
Description                  |
|--------------------|:--------:|:---------------:|:--------------:|---------
---------------------|
| `package-ecosystem` |{% octicon "check" aria-label="Supported" %}|{% octicon
"x" aria-label="Not supported" %}|{% octicon "check" aria-label="Supported" %}|
Package manager to use       |
| `directory`         |{% octicon "check" aria-label="Supported" %}|{% octicon
"x" aria-label="Not supported" %}|{% octicon "check" aria-label="Supported" %}|
Location of package manifests  |
| `schedule.interval` |{% octicon "check" aria-label="Supported" %}|{% octicon
"x" aria-label="Not supported" %}|{% octicon "check" aria-label="Supported" %}|
How often to check for updates |
```

Titles

Use quotation marks around article titles, whether the article is hosted on GitHub Docs or elsewhere. Do not include quotation marks around the names of external sites.

For further guidance, see “Formatting titles” in Microsoft’s Style Guide.

Short titles

We use short titles to populate the sidebar navigation. Since short titles appear in the sidebar navigation, they can use context to convey meaning and be slightly less precise than full titles. The goal of short titles is to help people find the content that they are looking for without having sidebar navigation items that are too long. Short titles give people contextual understanding of an article and align to the following standards.

- Short titles are 2-3 words long.
  - For categories, short titles must be less than 27 characters.
  - For map topics, short titles must be less than 30 characters.
  - For articles, short titles must be less than 31 characters and are ideally between 20 and 25 characters.
- Short titles use the base form of verbs instead of gerunds.
  - **Use:** "Configure notifications" instead of "Configuring notifications."
- Short titles for categories, map topics, and articles can omit product and feature names if it is clear what product or feature they relate to.
  - **Use:** "Configure notifications" as the short title for "Configuring notifications for Dependabot alerts" since the article is in the "Dependabot alerts" map topic.
- Short titles do not introduce new words that are not in the full title.
- Short titles should be parallel to short titles for similar content.
  - **Use:** "Organizations and teams" and "Enterprise accounts"
  - **Avoid:** "Organizations and teams" and "Managing enterprise accounts"

Writing short titles can be challenging. To help get short titles under the character count, consider the short title in context. Remove any repeated words if possible and any product or feature names that are in the map topic or category that the content belongs to.

User interface elements

Boldface

Use bold to describe UI elements that can be interacted with.

- In the left sidebar, click **Billing**.
- Look in the merge box at the bottom of the pull request's **Conversation** tab.
- Next to **Title**, add a descriptive label for your new key.

Branch names

Use code formatting for branch names.

- `main`

- `USERNAME.github.io`

Buttons

Format button names in bold and, whenever possible, omit the word “button.” To describe using a button, write “click”, not push or press.

- **Use:** Click **Pull request**.
- **Avoid:** Press the Pull request button.

Checkboxes

Format checkbox names in bold and omit the word “checkbox.” To describe choosing or clearing a checkbox, use “select” or “deselect.”

- **Use:** Select **Enable for all new repositories**.
- **Avoid:** Check the “Enable for all new repositories” checkbox.

Dynamic text

Use capital letters to indicate text that changes in the user interface or that the user needs to supply in a command or code snippet.

- **Use:** Click **Add USERNAME to REPONAME**.

Lists and list items

Format lists and clickable list items in bold. To describe interacting with a list, such as a dropdown menu or UI element that expands, regardless of whether the list name is a word or an octicon, write "select." To describe choosing a list item, write "click."

- **Use:** Select the **Backup email addresses** dropdown menu and click **Only allow primary email**.
- **Avoid:** Click the "Backup email addresses" dropdown menu and click **Only allow primary email**.

Location

Describe a user interface element’s location with standard terms.

- Under or above
- Next to
- Upper-left, upper-right, lower-left, lower-right
- Top of the page, bottom of the page, right side of the page, left side of the page

Panels

When possible, avoid referring to panels. Instead, describe what someone needs to do.

- **Use:** Click **View charts and graphs** for your repository, then select the time period you want to view from the dropdown menu.
- **Avoid:** Click the **View charts and graphs** to open the panel for your selected repository, then select the time period you want to view from the dropdown menu.

If you need to refer to a panel to describe a change to the UI or to explain how to interact with the UI, format the panel name as user interface text. Only include the word panel if it adds clarity or if the panel has no name in the UI.

- **Use:** In the "Security coverage" panel, select **Enable** or **Disable**.
- **Use:** In the panel, select **Enable** or **Disable**.

Radio buttons

Format radio button labels in bold and omit the words “radio button” or any other descriptor. To describe using a radio button, write "select."

Repository names

Use a standard format to refer to repositories. Link to repositories when helpful.

- **Use:** in the `[account-name/repository-name](URL)` repository

User interface text

When referencing text in the user interface, reproduce the text exactly. Use quotation marks to surround UI text that cannot be interacted with.

- **Use:** Under “IP allow list”, click **Edit**.

More resources

Microsoft Style Guide:

- Formatting text in instructions

Videos

You may add videos to reinforce text-based information but videos should never replace written content. Videos are inaccessible to some users and are also difficult to find by searching.

Videos on the GitHub Docs website must be well-produced and contain fewer barriers for people with disabilities, and conform to our content model for videos. For more information, see "About using videos in GitHub Docs."

Voice and tone

Use clear, simple language that’s approachable for a wide range of readers. Be authentic, empathetic, and confident with your writing.

Write for your audience: some jargon and technical terms are necessary, but don't rely on the assumption that every reader has the same level of technical expertise.

We are a global developer community. Avoid turns of phrase, idioms, and slang that are specific to a particular region or country.

To learn more about writing approachable content, see “Microsoft's brand voice: Above all, simple and human and “Top 10 tips for Microsoft style and voice.”

Word choice and terminology

For general guidance and GitHub-specific terms, see our "Glossary". For more detailed guidance, see the “A-Z word list” in Microsoft’s style guide.

Abbreviations

Spell out words except when referring to a word that’s explicitly shortened in the product itself.

- **Use:** Repository
- **Avoid:** Repo
- **Use:** Administrator, people with admin permissions
- **Avoid:** Admins

Do not use symbols or octicons that aren’t used in GitHub’s user interface.

- **Use:** Click **File**, then click **Edit**.
- **Avoid:** Click **File > Edit**.

Accounts

Product names and accounts

To avoid ambiguity and confusion, do not use product names as adjectives to describe accounts in any of our products. Instead, clarify the account type and choose clearer phrasing that avoids conflating accounts and products. When talking about accounts, only refer to the product name when needed to disambiguate between products. For more information about types of accounts available in GitHub's products, see "Types of GitHub accounts."

- **Use:** Your organization on GitHub Enterprise Cloud
- **Avoid:** Your GitHub Enterprise Cloud account
- **Avoid:** Your GitHub Enterprise Server organization
- **Use:** You can highlight your work on GitHub Enterprise Server by sending the contribution counts to your GitHub.com profile.

Individual people's accounts on GitHub

We refer to an account that an individual person signs into in various ways depending on the context.

Unless the content is about administering an enterprise product, describe an individual person's account on GitHub as a "personal account." This creates consistency with the UI and prevents readers from being confused by seeing two terms that mean the same thing.

- **Use:** Managing scheduled reminders for your personal account
- **Avoid:** Managing scheduled reminders for your user account

Accounts for enterprise products

With GitHub's enterprise products, administrators manage an enterprise account. An enterprise account can own multiple organizations, and people's user accounts can be members of the organizations. For more information, see the "Roles in an enterprise" article for each product.

- GitHub Enterprise Cloud
- GitHub Enterprise Server
- GitHub AE

If the reader manages an enterprise account, and you're describing the people's accounts that they manage, use "user account." This applies to the following products.

- GitHub Enterprise Cloud with Enterprise Managed Users
  - **Use:** With Enterprise Managed Users , you can create and manage user accounts for your enterprise members.
  - **Avoid:** With Enterprise Managed Users , you can create and manage the personal accounts for your enterprise members.
- GitHub Enterprise Server
  - **Use:** If you need to temporarily take over a user account...
  - **Avoid:** If you need to temporarily take over a personal account...
- GitHub AE
  - **Use:** Authorized users can access your enterprise from any IP address.
  - **Avoid:** Authorized personal accounts can access your enterprise from any IP address.

The following documentation should reference "user accounts."

- The "Enterprise administrator documentation" product
- Enterprise-specific billing documentation, like "About billing for your enterprise"
- Content within other products that's intended for an administrative audience, like "Best practices for securing accounts" in the "Code security" product or "Setting up a trial of GitHub Enterprise Cloud" in the "Get started" product
- Enterprise-specific API content, like the "GitHub Enterprise administration" REST API reference documentation

For enterprises on GitHub Enterprise Cloud that don't use Enterprise Managed Users, use "personal account" when describing members of organizations owned by the enterprise.

- **Use:** If you configure SAML SSO, members of your organization will continue to sign into their personal accounts on GitHub.com.
- **Avoid:** If you configure SAML SSO, members of your organization will continue to sign into their user accounts on GitHub.com.

Documentation that describes GitHub Enterprise Cloud without Enterprise Managed Users is generally in the "Managing SAML single sign-on for your organization" category.

People's accounts for other services

When you describe a person's account for a service other than GitHub, such as an integration or authentication provider, use "user account."

Acronyms

Spell out acronyms the first time they’re used in an article, except in titles or headers.

Apps

Use "app" or "application" in general content.

- **Use:** Publish and list your app in GitHub Marketplace

Use "app" when referring to OAuth apps since these are not a product.

- **Use:** Register an OAuth app
- **Avoid:** Register an OAuth App

Use "App" when referring to GitHub Apps since this is a product.

- **Use:** Register a GitHub App

GitHub Apps and OAuth apps consist of two parts: the app registration, and the code that makes the app do something.

- To refer to just the GitHub App settings/configuration in the GitHub UI, use terminology like "register" and "GitHub App registration".
  - **Use:** Register a GitHub App
  - **Use:** Update a GitHub App registration
  - **Avoid:** Create a GitHub App
  - **Avoid:** Modify a GitHub App
- To refer to just the code for the app, use terminology like "code for your app" or "your app's code".
  - **Use:** code for your app
  - **Use:** code for your GitHub App
  - **Use:** your app's code
  - **Avoid:** Your GitHub App
  - **Avoid:** Your OAuth app
- To refer to the whole app collectively (registration + code), refer to it as a GitHub App or OAuth app.

GitHub Apps can be installed on organization and user accounts. To refer to an installation of the app, use "GitHub App installation" instead of "GitHub App."

Currency

When referring to dollars, cents, amounts of currency or using the `$` sign, ensure the currency used is defined even if the amount is zero. Use the ISO standard currency name, and the ISO standard currency code where possible.

Use lowercase for currency names, but capitalize the reference to the country or region.

- **Use:** US dollar.
- **Avoid:** US Dollar, $USD dollar.

Use uppercase for currency codes.

- **Use:** USD.

Where there is only one reference in an article, use the currency name without a `$` sign preceding the amount.

- **Use:** `10 US dollars` for a single reference to currency.

Where an article contains several references to the same currency, ensure that the first reference uses the currency name without a `$` sign preceding the amount and includes the currency code in parentheses following the currency name.

For subsequent references to currency in an article or where appropriate (such as when space is a consideration, or when several amounts are presented in a table or list), include the `$` sign preceding the amount and use the ISO standard currency code following the amount.

- **Use:** `10 US dollars (USD)` for the first reference, and `$0.25 USD` for subsequent references.
- **Avoid:** `$10 US dollars (USD)`, `USD$0.25`.

Where the first reference concerns `cents` or a non-dollar amount, capitalize the reference to the country or region of the currency used in parentheses immediately after the first reference. Subsequent currency references are treated using the guidelines above.

- **Use:** `99 cents (US currency)` for the first reference, and `99 cents` for subsequent references.
- **Avoid:** `$0.99 (US currency)`, `$0.99 USD cents`, `USD$0.99 cents`.

Permissions

A **permission** is the ability to perform a specific action. For example, the ability to delete an issue is a permission.

A **role** is a set of permissions that can be assigned to a user. Roles exist at different levels.

- Accounts (e.g., organization owner, billing manager for an enterprise account)
- Resources (e.g., "Write" for a repository, "Admin" for a security advisory)
- Teams (e.g., "team maintainer")

A person's **access** refers generally to all the abilities the person has in a particular context, regardless of which roles or individual permissions those abilities come from.

Only use **permission** or **role** when the distinction between the two is important. Otherwise, use **access**.

- **Use:** `To create a custom repository role, you choose an inherited role and then add individual permissions.`
- **Use:** `Managing a team's access to your organization's repository`
- **Use:** `If your team membership gives you a different level of access than your role as organization owner...`
- **Use:** `People with write access can...`
- **Avoid:** `People with the write role can...`
- **Avoid:** `People with write permissions can...`
- **Avoid:** `People with write privileges can...`

When specifying the access required to take an action, refer only to the role at the same level as the action. For example, you need admin access to a repository, which is a repository-level role, to configure protected branches. You can get admin access to a repository by being an organization owner, an organization-level role, but the repository-level role is what actually governs your ability to take the action, so that is the only role that should be mentioned.

- **Use:** `People with write access to a repository can do X to the repository.`
- **Avoid:** `Organization owners and people with write access can do X to the repository.`

For more information about word choice for permissions statements, see "[Contents of a GitHub Docs article]" in the content model.

Prepositions

Avoid ending a sentence with a preposition unless the rewritten sentence would sound awkward or too formal.

Product names

See the “[Product names]” section of this guide.

Terms to use or avoid

Use	Avoid
person	user, customer
terminal	shell
username	login
sign in	log in, login
sign up	signup
recommended limit	soft limit
email	e-mail
frontmatter	front matter, front-matter
on GitHub	on a remote repository
press (a key)	hit, tap
type (in the user interface)	enter (in the user interface)
enter (in the command line)	type (in the command line)

Word order

Strings of nouns

Avoid stacked modifiers (strings of nouns), which can lead to incorrect translations because translations may not be able to tell which word is modifying the other. You can rephrase the string of nouns using a preposition. If using a stacked modifier is essential, make sure the background information and context are clear so that readers and the translator can understand what is being modified.

- **Use:** Default source settings for public repositories
- **Avoid:** Public repository default source settings

1. Not to be confused with Davy Jones of The Monkees ↩
2. Also humans ↩