Templates

In this article

This article contains starter templates for the different content types used in GitHub Docs.

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

Conceptual article template

Use the content model for full instructions and examples on how to write conceptual content. For more information, see "Conceptual content type."

```
---
title: 'About <subject>'
shortTitle: '<subject>'
intro: 'Article intro. See tips for a great intro below.'
product: "optional product callout"
type: overview
topics:
  - topic
versions:
  - version
---

{% comment %}
- Follow the guidelines in https://docs.github.com/contributing/writing-for-
github-docs/content-model#conceptual to write this article.
- Great intros give readers a quick understanding of what's in the article, so
they can tell whether it's relevant to them before moving ahead. For more tips,
see https://docs.github.com/contributing/writing-for-github-docs/content-model
- For product callout info, see
https://github.com/github/docs/tree/main/content#product
- For product version instructions, see
https://github.com/github/docs/tree/main/content#versioning
- Remove these comments from your article file when you're done writing.
{% endcomment %}

## A section here

{% comment %}
Write one or two paragraphs about the main idea of your topic, as a summary.
Make sure you don't have any content that isn't preceded by a header, or it won't
be linkable in our table of contents.
```

```
{% endcomment %}

## Another section here

{% comment %}
Write one or two paragraphs about another element of your topic.
Keep adding headers and sections until you've completed your article.
{% endcomment %}

## Further reading

{% comment %}
Optionally, include a bulleted list of related articles the user can reference to
extend the concepts covered in this article. Consider linking to procedural
articles or tutorials that help the user use the information in your article.
{% endcomment %}

- "[Article title](article-URL)"
```

Referential article template

Use the content model for full instructions and examples on how to write referential content. For more information, see "Referential content type."

```
---
title: Nouns describing your subject
shortTitle: <subject> # Max 31 characters
intro: 'Article intro. See tips for a great intro below.'
product: "{{ optional product callout }}"
type: reference
topics:
  - <topic> # One or more from list of allowed topics:
https://github.com/github/docs/blob/main/data/allowed-topics.js
versions:
  - <version>
---

{% comment %}
Follow the guidelines in https://docs.github.com/contributing/writing-for-github-
docs/content-model#referential to write this article.-- >
Great intros give readers a quick understanding of what's in the article, so they
can tell whether it's relevant to them before moving ahead. For more tips, see
https://docs.github.com/contributing/writing-for-github-docs/content-model
For product callout info, see
https://github.com/github/docs/tree/main/content#product
For product version instructions, see
https://github.com/github/docs/tree/main/content#versioning
Remove these comments from your article file when you're done writing
{% endcomment %}

## A section here

{% comment %}
Write one or two paragraphs about the main idea of your topic, as a summary.
Make sure you don't have any content that isn't preceded by a header, or it won't
be linkable in our table of contents.
{% endcomment %}

## Another section here

{% comment %}
Write one or two paragraphs about another element of your topic.
Keep adding headers and sections until you've completed your article.
{% endcomment %}

## Further reading
```

```
{% comment %}
Optionally, include a bulleted list of related articles the user can reference to
extend the concepts covered in this article. Consider linking to procedural
articles or tutorials that help the user use the information in your article.
{% endcomment %}

- "[Article title](article-URL)"
```

Procedural article template

Use the content model for full instructions and examples on how to write procedural content. For more information, see "Procedural content type."

```
---
title: Start with a present participle
shortTitle: <subject> # Max 31 characters
intro: 'Article intro. See tips for a great intro below.'
product: "{{ optional product callout }}"
type: how_to
topics:
  - <topic> # One or more from list of allowed topics:
https://github.com/github/docs/blob/main/data/allowed-topics.js
versions:
  - <version>
---

{% comment %}
Follow the guidelines in https://docs.github.com/contributing/writing-for-github-
docs/content-model#procedural to write this article.-- >
Great intros give readers a quick understanding of what's in the article, so they
can tell whether it's relevant to them before moving ahead. For more tips, see
https://docs.github.com/contributing/writing-for-github-docs/content-model
For product callout info, see
https://github.com/github/docs/tree/main/content#product
For product version instructions, see
https://github.com/github/docs/tree/main/content#versioning
Remove these comments from your article file when you're done writing
{% endcomment %}

## Procedural section header here

{% comment %}
Include prerequisite information or specific permissions information here.
Then write procedural steps following the instructions in
https://docs.github.com/contributing/style-guide-and-content-model/style-
guide#procedural-steps.
Check if there's already a reusable string for the step you want to write in
https://github.com/github/docs/tree/main/data/reusables. Look at the source file
for a procedure located in the same area of the user interface to find reusables.
{% endcomment %}

## Optionally, another procedural section here

{% comment %}
Keep adding procedures until you've finished writing your article.
{% endcomment %}

## Further reading

{% comment %}
Optionally, include a bulleted list of related articles the user can reference to
extend the concepts covered in this article. Consider linking to procedural
articles or tutorials that help the user use the information in your article.
{% endcomment %}

- "[Article title](article-URL)"
```

Quickstart article template

Use the content model for full instructions and examples on how to write quickstarts. For more information, see "Quickstart content type."

```
---
title: Quickstart title
shortTitle: <subject> # Max 31 characters
intro: 'Article intro. Highlight that the guide is quick and to the point.'
type: quick_start
topics:
  - <topic> # One or more from list of allowed topics:
https://github.com/github/docs/blob/main/data/allowed-topics.js
versions:
  - <version>
---

{% comment %}
Follow the guidelines in https://docs.github.com/contributing/writing-for-github-
docs/content-model#quickstart to write this article.
For product version instructions, see
https://github.com/github/docs/tree/main/content#versions.
The entire quickstart should be about 600 words long or take about five minutes
to read.
Remove these comments from your article file when you're done writing
{% endcomment %}

## Introduction

{% comment %}
Build on the quick phrasing above by
- Clarifying the audience
- Clearly stating prerequisites and prior knowledge needed
- Stating what the user will accomplish or build
{% endcomment %}

## Step one: Action the user will take

{% comment %}
In one sentence, describe what the user will do in this step
Steps should break down the tasks the user will complete in sequential order
Avoid replicating conceptual information that is covered elsewhere, provide
inline links instead. Only include conceptual information unique to this use
case.
{% endcomment %}

### Task chunk

{% comment %}
A step may require the user to perform several tasks - break those tasks down
into chunks, allowing the user to scan quickly to find their place if they
navigated away from this screen to perform the task.
An example might be creating a personal access token for the action to use and
then storing it in secrets
For UI based tasks, include the button or options the users should click
If the task adds code, include the code in context (don't just show `needs setup`
show the entire `setup` and `dependent` jobs)
{% endcomment %}

### Another task chunk

## Step 2: Do the next thing

{% comment %}
Rinse and repeat, adding steps and tasks until the tutorial is complete
{% endcomment %}

## Next steps
```

```
{% comment %}
Provide a quick recap of what has been accomplished in the quick start as a means
of transitioning to next steps. Include 2-3 actionable next steps that the user
take after completing the quickstart. Always link to conceptual content on the
feature or product. You can also link off to other related information on
docs.github.com or in GitHub Skills.
{% endcomment %}
```

Tutorial article template

Use the content model for full instructions and examples on how to write tutorials. For more information, see "Tutorial content type."

```
---
title: Tutorial title
shortTitle: <subject> # Max 31 characters
intro: 'Article intro. See tips for a great intro below'
product: "{{ optional product callout }}"
type: tutorial
topics:
  - <topic> # One or more from list of allowed topics:
https://github.com/github/docs/blob/main/data/allowed-topics.js
versions:
  - <version>
---

{% comment %}
Follow the instructions in https://docs.github.com/contributing/writing-for-
github-docs/content-model#quickstart to write this article.
Great intros clarify who the tutorial is intended for, state what the user will
accomplish, and state the technologies that will be used.
For product callout info, see
https://github.com/github/docs/tree/main/content#product
For product version instructions, see
https://github.com/github/docs/tree/main/content#versioning
Remove these comments from your article file when you're done writing
{% endcomment %}

## Introduction

{% comment %}
The tutorial introduction should include the following in a short paragraph -

- Clarify audience
- State prerequisites and prior knowledge needed
- State what the user will accomplish or build and the user problem it solves
- Link to an example of the project the user will complete
{% endcomment %}

## Step 1: Action the user will take

{% comment %}
In one sentence, describe what the user will do in this step
Steps should break down the tasks the user will complete in sequential order
Avoid replicating conceptual information that is covered elsewhere, provide
inline links instead. Only include conceptual information unique to this use
case.
{% endcomment %}

### Task chunk

{% comment %}
A step may require the user to perform several tasks - break those tasks down
into chunks, allowing the user to scan quickly to find their place if they
navigated away from this screen to perform the task.
An example might be creating a personal access token for the action to use and
```

```
then storing it in secrets
For UI based tasks, include the button or options the users should click
If the task adds code, include the code in context (don't just show `needs:
setup` show the entire `setup` and `dependent` jobs)
{% endcomment %}

### Another task chunk

## Step 2: Do the next thing

{% comment %}
Rinse and repeat, adding steps and tasks until the tutorial is complete
Remember to show code snippets in context
{% endcomment %}

## Further reading

{% comment %}
Include a bulleted list of tutorials or articles the user can reference to extend
the concepts taught in this tutorial
{% endcomment %}

- "[Article title](article-URL)"
```

Language guides for GitHub Actions

Use the content model for full instructions and examples on how to write for GitHub Docs. For more information, see "About the content model."

```
---
title: Guide title
shortTitle: <subject> # Max 31 characters
intro: 'Article intro. See tips for a great intro below'
product: "{{ optional product callout }}"
type: tutorial
topics:
  - <topic> # One or more from list of allowed topics:
https://github.com/github/docs/blob/main/data/allowed-topics.js
versions:
  - <version>
---

{% comment %}
- Great intros clarify who the guide is intended for, state what the user will
accomplish, and state the technologies that will be used.
- Intros are typically 1-3 sentence summaries, with a longer "Introduction"
section that follows.
- Remove these comments from your article file when you're done writing
{% endcomment %}

## Introduction

{% comment %}
The language guide introduction should include the following in a short paragraph
-
- Clarify audience.
- State prerequisites and prior knowledge needed.
- Should the user have read any other articles?
- State what the user will accomplish or build and the user problem it solves.
{% endcomment %}

## Starting with the <language> workflow template

{% comment %}
Language guides typically walk through and build upon a starter workflow
template. If that format doesn't work, you can include a boilerplate workflow.
- Link to the GitHub Actions CI starter workflow as the boilerplate reference
```

```
code and then walk through and build on that code in this guide -
https://github.com/actions/starter-workflows/tree/master/ci
- Provide instructions for adding the starter workflow template to a repository.
- Include the starter template workflow code.
{% endcomment %}

## Running on different operating systems

{% comment %}
Include a brief overview of how to choose the runner environment. These should be
alternatives to what operating system is presented in the starter
workflow/boilerplate template.
{% endcomment %}

## Configuring the <language> version

{% comment %}
- Describe when and how to use available setup actions that configure the version
of the language on the runner (ex. actions/setup-node).
- How does the setup action configure the version and what happens when the
version isn't supported in the environment. What is the default version, when no
version is configured.
- Include any additional features the setup action might provide that are useful
to CI.
- If applicable, provide examples of configuring exact versions or major/minor
versions.
- Include information about software already installed on GitHub-hosted runners
or software configuration necessary to build and test the project.
- Provide examples of configuring matrix strategies.
- Link out to any docs about available software on the GitHub-hosted runners.
(Ex. https://docs.github.com/en/actions/reference/software-installed-on-github-
hosted-runners).
- Include code samples.
{% endcomment %}

## Installing dependencies

{% comment %}
- Include example of installing dependencies to prepare for building and testing.
- Are there any dependencies or scenarios where people might need to install
packages globally?
- Include examples of common package managers.
- If the language is supported by GitHub Packages, include an example installing
dependencies from GitHub.
- Include code samples.
{% endcomment %}

## Caching dependencies

{% comment %}
Include an example of restoring cached dependencies. We'll want to link out to
the article about caching for more information
(https://docs.github.com/en/actions/configuring-and-managing-workflows/caching-
dependencies-to-speed-up-workflows).
{% endcomment %}

## Building your code

{% comment %}
- Include any compile steps.
- Include any test commands.
- Note that you can use the same commands that your repository needs to build and
test your code by simply replacing the commands in the `run` keyword.
- Include any basic examples or commands specific to test frameworks.
- Include any common databases or services that might be needed. If so, we can
link out to the services guides in the docs
(https://docs.github.com/en/actions/configuring-and-managing-workflows/using-
databases-and-service-containers).
{% endcomment %}

## Packaging workflow data as artifacts
```

```
{% comment %}
This section can simply link out to
https://docs.github.com/en/actions/configuring-and-managing-workflows/persisting-
workflow-data-using-artifacts or provide additional information about which
artifacts might be typical to upload for a CI workflow.
{% endcomment %}
```