Suggesting content improvements

If you notice an issue with Kubernetes documentation or have an idea for new content, then open an issue. All you need is a [GitHub account](#) and a web browser.

In most cases, new work on Kubernetes documentation begins with an issue in GitHub. Kubernetes contributors then review, categorize and tag issues as needed. Next, you or another member of the Kubernetes community open a pull request with changes to resolve the issue.

Opening an issue

If you want to suggest improvements to existing content or notice an error, then open an issue.

1. Click the **Create an issue** link on the right sidebar. This redirects you to a GitHub issue page pre-populated with some headers.
2. Describe the issue or suggestion for improvement. Provide as many details as you can.
3. Click **Submit new issue**.

After submitting, check in on your issue occasionally or turn on GitHub notifications. Reviewers and other community members might ask questions before they can take action on your issue.

Suggesting new content

If you have an idea for new content, but you aren't sure where it should go, you can still file an issue. Either:

- Choose an existing page in the section you think the content belongs in and click **Create an issue**.
- Go to [GitHub](#) and file the issue directly.

How to file great issues

Keep the following in mind when filing an issue:

- Provide a clear issue description. Describe what specifically is missing, out of date, wrong, or needs improvement.
- Explain the specific impact the issue has on users.
- Limit the scope of a given issue to a reasonable unit of work. For problems with a large scope, break them down into smaller issues. For example, "Fix the security docs" is too broad, but "Add details to the 'Restricting network access' topic" is specific enough to be actionable.
- Search the existing issues to see if there's anything related or similar to the new issue.
- If the new issue relates to another issue or pull request, refer to it either by its full URL or by the issue or pull request number prefixed with a `#` character. For example, `Introduced by #987654`.
- Follow the [Code of Conduct](#). Respect your fellow contributors. For example, "The docs are terrible" is not helpful or polite feedback.

Writing a new topic

This page shows how to create a new topic for the Kubernetes docs.

Before you begin

Create a fork of the Kubernetes documentation repository as described in [Open a PR](#).

Choosing a page type

As you prepare to write a new topic, think about the page type that would fit your content the best:

Type	Description
Concept	A concept page explains some aspect of Kubernetes. For example, a concept page might describe the Kubernetes Deployment object and explain the role it plays as an application while it is deployed, scaled, and updated. Typically, concept pages don't include sequences of steps, but instead provide links to tasks or tutorials. For an example of a concept topic, see [Nodes](#).
Task	A task page shows how to do a single thing. The idea is to give readers a sequence of steps that they can actually do as they read the page. A task page can be short or long, provided it stays focused on one area. In a task page, it is OK to blend brief explanations with the steps to be performed, but if you need to provide a lengthy explanation, you should do that in a concept topic. Related task and concept topics should link to each other. For an example of a short task page, see [Configure a Pod to Use a Volume for Storage](#). For an example of a longer task page, see [Configure Liveness and Readiness Probes](#)
Tutorial	A tutorial page shows how to accomplish a goal that ties together several Kubernetes features. A tutorial might provide several sequences of steps that readers can actually do as they read the page. Or it might provide explanations of related pieces of code. For example, a tutorial could provide a walkthrough of a code sample. A tutorial can include brief explanations of the Kubernetes features that are being tied together, but should link to related concept topics for deep explanations of individual features.

Creating a new page

Use a [content type](#) for each new page that you write. The docs site provides templates or [Hugo archetypes](#) to create new content pages. To create a new type of page, run `hugo new` with the path to the file you want to create. For example:

```
hugo new docs/concepts/my-first-concept.md
```

Choosing a title and filename

Choose a title that has the keywords you want search engines to find. Create a filename that uses the words in your title separated by hyphens. For example, the topic with title [Using an HTTP Proxy to Access the Kubernetes API](#) has filename `http-proxy-access-api.md`. You don't need to put "kubernetes" in the filename, because "kubernetes" is already in the URL for the topic, for example:

```
/docs/tasks/extend-kubernetes/http-proxy-access-api/
```

Adding the topic title to the front matter

In your topic, put a `title` field in the [front matter](#). The front matter is the YAML block that is between the triple-dashed lines at the top of the page. Here's an example:

```
---
title: Using an HTTP Proxy to Access the Kubernetes API
---
```

Choosing a directory

Depending on your page type, put your new file in a subdirectory of one of these:

- /content/en/docs/tasks/
- /content/en/docs/tutorials/
- /content/en/docs/concepts/

You can put your file in an existing subdirectory, or you can create a new subdirectory.

Placing your topic in the table of contents

The table of contents is built dynamically using the directory structure of the documentation source. The top-level directories under `/content/en/docs/` create top-level navigation, and subdirectories each have entries in the table of contents.

Each subdirectory has a file `_index.md`, which represents the "home" page for a given subdirectory's content. The `_index.md` does not need a template. It can contain overview content about the topics in the subdirectory.

Other files in a directory are sorted alphabetically by default. This is almost never the best order. To control the relative sorting of topics in a subdirectory, set the `weight:` front-matter key to an integer. Typically, we use multiples of 10, to account for adding topics later. For instance, a topic with weight `10` will come before one with weight `20`.

Embedding code in your topic

If you want to include some code in your topic, you can embed the code in your file directly using the markdown code block syntax. This is recommended for the following cases (not an exhaustive list):

- The code shows the output from a command such as `kubectl get deploy mydeployment -o json | jq '.status'`.
- The code is not generic enough for users to try out. As an example, you can embed the YAML file for creating a Pod which depends on a specific [FlexVolume](#) implementation.
- The code is an incomplete example because its purpose is to highlight a portion of a larger file. For example, when describing ways to customize a [RoleBinding](#), you can provide a short snippet directly in your topic file.
- The code is not meant for users to try out due to other reasons. For example, when describing how a new attribute should be added to a resource using the `kubectl edit` command, you can provide a short example that includes only the attribute to add.

Including code from another file

Another way to include code in your topic is to create a new, complete sample file (or group of sample files) and then reference the sample from your topic. Use this method to include sample YAML files when the sample is generic and reusable, and you want the reader to try it out themselves.

When adding a new standalone sample file, such as a YAML file, place the code in one of the `<LANG>/examples/` subdirectories where `<LANG>` is the language for the topic. In your topic file, use the `code_sample` shortcode:

```
{{% code_sample file="<RELPATH>/my-example-yaml>" %}}
```

where `<RELPATH>` is the path to the file to include, relative to the `examples` directory. The following Hugo shortcode references a YAML file located at /content/en/examples/pods/storage/gce-volume.yaml.

```
{{% code_sample file="pods/storage/gce-volume.yaml" %}}
```

Showing how to create an API object from a configuration file

If you need to demonstrate how to create an API object based on a configuration file, place the configuration file in one of the subdirectories under `<LANG>/examples`.

In your topic, show this command:

```
kubectl create -f https://k8s.io/examples/pods/storage/gce-volume.yaml
```

Note:

When adding new YAML files to the `<LANG>/examples` directory, make sure the file is also included into the `<LANG>/examples_test.go` file. The Travis CI for the Website automatically runs this test case when PRs are submitted to ensure all examples pass the tests.

For an example of a topic that uses this technique, see [Running a Single-Instance Stateful Application](#).

Adding images to a topic

Put image files in the `/images` directory. The preferred image format is SVG.

What's next

- Learn about [using page content types](#).
- Learn about [creating a pull request](#).

Diagram Guide

This guide shows you how to create, edit and share diagrams using the Mermaid JavaScript library. Mermaid.js allows you to generate diagrams using a simple markdown-like syntax inside Markdown files. You can also use Mermaid to generate `.svg` or `.png` image files that you can add to your documentation.

The target audience for this guide is anybody wishing to learn about Mermaid and/or how to create and add diagrams to Kubernetes documentation.

Figure 1 outlines the topics covered in this section.

```
flowchart LR subgraph m[Mermaid.js] direction TB S[ ]-.- C[build
diagrams
with markdown] --> D[on-line
live editor] end A[Why are diagrams
useful?] --> m m --> N[3 x methods
for creating
diagrams] N --> T[Examples] T --> X[Styling
and
captions] X --> V[Tips] classDef box fill:#fff,stroke:#000,stroke-width:1px,color:#000; classDef spacewhite fill:#ffffff,stroke:#fff,stroke-width:0px,color:#000 class A,C,D,N,X,m,T,V box class S spacewhite %% you can hyperlink Mermaid diagram nodes to a URL using click
statements click A "https://mermaid-js.github.io/mermaid-live-
editor/edit/#eyJjb2RlIjoiZmxvd2NoYXJ0IExSXG4gICAgc3ViZ3JhcGggbVtNZXJtYWlkLmpzXVxuICAgIGRpcmVjdGlvbiBUQlxuICAgICAgICBTW
ZGlhZ3JhbXM8YnI-d2l0aCBtYXJrZG93bl0gLS0-
XG4gICAgICAgIERbb24tbGluZTxicj5saXZlIGVkaXRvcl1cbiAgICBlbmRcbiAgICBBW1doeSBhcmUgZGlhZ3JhbXM8YnI-
dXNlZnVsP10gLS0-
IG1cbiAgICBtIC0tPiBOWzMgeCBtZXRob2RzPGJyPmZvciBjcmVhdGluZzxicj5kaWFncmFtc11cbiAgICBOIC0tPiBUW0V4YW1wbGVzXVxuICAg
IFhbU3R5bGluZzxicj5hbmQQ8YnI-
Y2FwdGlvbnNdXG4gICAgWCAtLT4gVltUaXBzXVxuICAgIFxuIFxuICAgIGNsYXNzRGVmIGJveCBmaWxsOiNmZmYsc3Ryb2tlOiMwMDAsc3R
_blank click C "https://mermaid-js.github.io/mermaid-live-
editor/edit/#eyJjb2RlIjoiZmxvd2NoYXJ0IExSXG4gICAgc3ViZ3JhcGggbVtNZXJtYWlkLmpzXVxuICAgIGRpcmVjdGlvbiBUQlxuICAgICAgICBTW
ZGlhZ3JhbXM8YnI-d2l0aCBtYXJrZG93bl0gLS0-
XG4gICAgICAgIERbb24tbGluZTxicj5saXZlIGVkaXRvcl1cbiAgICBlbmRcbiAgICBBW1doeSBhcmUgZGlhZ3JhbXM8YnI-
dXNlZnVsP10gLS0-
IG1cbiAgICBtIC0tPiBOWzMgeCBtZXRob2RzPGJyPmZvciBjcmVhdGluZzxicj5kaWFncmFtc11cbiAgICBOIC0tPiBUW0V4YW1wbGVzXVxuICAg
IFhbU3R5bGluZzxicj5hbmQQ8YnI-
Y2FwdGlvbnNdXG4gICAgWCAtLT4gVltUaXBzXVxuICAgIFxuIFxuICAgIGNsYXNzRGVmIGJveCBmaWxsOiNmZmYsc3Ryb2tlOiMwMDAsc3R
_blank click D "https://mermaid-js.github.io/mermaid-live-
editor/edit/#eyJjb2RlIjoiZmxvd2NoYXJ0IExSXG4gICAgc3ViZ3JhcGggbVtNZXJtYWlkLmpzXVxuICAgIGRpcmVjdGlvbiBUQlxuICAgICAgICBTW
ZGlhZ3JhbXM8YnI-d2l0aCBtYXJrZG93bl0gLS0-
XG4gICAgICAgIERbb24tbGluZTxicj5saXZlIGVkaXRvcl1cbiAgICBlbmRcbiAgICBBW1doeSBhcmUgZGlhZ3JhbXM8YnI-
dXNlZnVsP10gLS0-
IG1cbiAgICBtIC0tPiBOWzMgeCBtZXRob2RzPGJyPmZvciBjcmVhdGluZzxicj5kaWFncmFtc11cbiAgICBOIC0tPiBUW0V4YW1wbGVzXVxuICAg
IFhbU3R5bGluZzxicj5hbmQQ8YnI-
Y2FwdGlvbnNdXG4gICAgWCAtLT4gVltUaXBzXVxuICAgIFxuIFxuICAgIGNsYXNzRGVmIGJveCBmaWxsOiNmZmYsc3Ryb2tlOiMwMDAsc3R
_blank click N "https://mermaid-js.github.io/mermaid-live-
editor/edit/#eyJjb2RlIjoiZmxvd2NoYXJ0IExSXG4gICAgc3ViZ3JhcGggbVtNZXJtYWlkLmpzXVxuICAgIGRpcmVjdGlvbiBUQlxuICAgICAgICBTW
ZGlhZ3JhbXM8YnI-d2l0aCBtYXJrZG93bl0gLS0-
XG4gICAgICAgIERbb24tbGluZTxicj5saXZlIGVkaXRvcl1cbiAgICBlbmRcbiAgICBBW1doeSBhcmUgZGlhZ3JhbXM8YnI-
dXNlZnVsP10gLS0-
IG1cbiAgICBtIC0tPiBOWzMgeCBtZXRob2RzPGJyPmZvciBjcmVhdGluZzxicj5kaWFncmFtc11cbiAgICBOIC0tPiBUW0V4YW1wbGVzXVxuICAg
IFhbU3R5bGluZzxicj5hbmQQ8YnI-
Y2FwdGlvbnNdXG4gICAgWCAtLT4gVltUaXBzXVxuICAgIFxuIFxuICAgIGNsYXNzRGVmIGJveCBmaWxsOiNmZmYsc3Ryb2tlOiMwMDAsc3R
_blank click T "https://mermaid-js.github.io/mermaid-live-
editor/edit/#eyJjb2RlIjoiZmxvd2NoYXJ0IExSXG4gICAgc3ViZ3JhcGggbVtNZXJtYWlkLmpzXVxuICAgIGRpcmVjdGlvbiBUQlxuICAgICAgICBTW
ZGlhZ3JhbXM8YnI-d2l0aCBtYXJrZG93bl0gLS0-
XG4gICAgICAgIERbb24tbGluZTxicj5saXZlIGVkaXRvcl1cbiAgICBlbmRcbiAgICBBW1doeSBhcmUgZGlhZ3JhbXM8YnI-
dXNlZnVsP10gLS0-
IG1cbiAgICBtIC0tPiBOWzMgeCBtZXRob2RzPGJyPmZvciBjcmVhdGluZzxicj5kaWFncmFtc11cbiAgICBOIC0tPiBUW0V4YW1wbGVzXVxuICAg
IFhbU3R5bGluZzxicj5hbmQQ8YnI-
Y2FwdGlvbnNdXG4gICAgWCAtLT4gVltUaXBzXVxuICAgIFxuIFxuICAgIGNsYXNzRGVmIGJveCBmaWxsOiNmZmYsc3Ryb2tlOiMwMDAsc3R
_blank click X "https://mermaid-js.github.io/mermaid-live-
editor/edit/#eyJjb2RlIjoiZmxvd2NoYXJ0IExSXG4gICAgc3ViZ3JhcGggbVtNZXJtYWlkLmpzXVxuICAgIGRpcmVjdGlvbiBUQlxuICAgICAgICBTW
ZGlhZ3JhbXM8YnI-d2l0aCBtYXJrZG93bl0gLS0-
XG4gICAgICAgIERbb24tbGluZTxicj5saXZlIGVkaXRvcl1cbiAgICBlbmRcbiAgICBBW1doeSBhcmUgZGlhZ3JhbXM8YnI-
```

dXNlZnVsP10gLS0-
IG1cbiAgICBtIC0tPiBOWzMgeCBtZXRob2RzPGJyPmZvciBjcmVhdGluZzxicj5kaWFncmFtc11cbiAgICBOIC0tPiBUW0V4YW1wbGVzXVxuICAg
IFhbU3R5bGluZzxicj5hbmQ8YnI-
Y2FwdGlvbnNdXG4gICAgWCAtLT4gVltUaXBzXVxuICAgIFxuIFxuICAgIGNsYXNzRGVmIGJveCBmaWxsOiNmZmYsc3Ryb2tlOiMwMDAsc3R
_blank click V "https://mermaid-js.github.io/mermaid-live-
editor/#eyJjb2RlIjoiZmxvd2NoYXJ0IExSXG4gICAgc3ViZ3JhcGggbVtNZXJtYWlkLmpzXVxuICAgIGRpcmVjdGlvbiBUQlxuICAgICBTV
ZGlhZ3JhbXM8YnI-d2l0aCBtYXJrZG93bl0gLS0-
XG4gICAgICAgIERbb24tbGluZTxicj5saXZlIGVkaXRvcl1cbiAgICBlbmRcbiAgICBBW1doeSBhcmUgZGlhZ3JhbXM8YnI-
dXNlZnVsP10gLS0-
IG1cbiAgICBtIC0tPiBOWzMgeCBtZXRob2RzPGJyPmZvciBjcmVhdGluZzxicj5kaWFncmFtc11cbiAgICBOIC0tPiBUW0V4YW1wbGVzXVxuICAg
IFhbU3R5bGluZzxicj5hbmQ8YnI-
Y2FwdGlvbnNdXG4gICAgWCAtLT4gVltUaXBzXVxuICAgIFxuIFxuICAgIGNsYXNzRGVmIGJveCBmaWxsOiNmZmYsc3Ryb2tlOiMwMDAsc3R
_blank

Figure 1. Topics covered in this section.

All you need to begin working with Mermaid is the following:

- Basic understanding of markdown.
- Using the Mermaid live editor.
- Using Hugo shortcodes.
- Using the Hugo {{< figure >}} shortcode.
- Performing Hugo local previews.
- Familiar with the Contributing new content process.

Note:

You can click on each diagram in this section to view the code and rendered diagram in the Mermaid live editor.

Why you should use diagrams in documentation

Diagrams improve documentation clarity and comprehension. There are advantages for both the user and the contributor.

The user benefits include:

- **Friendly landing spot**. A detailed text-only greeting page could intimidate users, in particular, first-time Kubernetes users.
- **Faster grasp of concepts**. A diagram can help users understand the key points of a complex topic. Your diagram can serve as a visual learning guide to dive into the topic details.
- **Better retention**. For some, it is easier to recall pictures rather than text.

The contributor benefits include:

- **Assist in developing the structure and content** of your contribution. For example, you can start with a simple diagram covering the high-level points and then dive into details.
- **Expand and grow the user community**. Easily consumed documentation augmented with diagrams attracts new users who might previously have been reluctant to engage due to perceived complexities.

You should consider your target audience. In addition to experienced K8s users, you will have many who are new to Kubernetes. Even a simple diagram can assist new users in absorbing Kubernetes concepts. They become emboldened and more confident to further explore Kubernetes and the documentation.

Mermaid

Mermaid is an open source JavaScript library that allows you to create, edit and easily share diagrams using a simple, markdown-like syntax configured inline in Markdown files.

The following lists features of Mermaid:

- Simple code syntax.
- Includes a web-based tool allowing you to code and preview your diagrams.
- Supports multiple formats including flowchart, state and sequence.
- Easy collaboration with colleagues by sharing a per-diagram URL.
- Broad selection of shapes, lines, themes and styling.

The following lists advantages of using Mermaid:

- No need for separate, non-Mermaid diagram tools.
- Adheres to existing PR workflow. You can think of Mermaid code as just Markdown text included in your PR.
- Simple tool builds simple diagrams. You don't want to get bogged down (re)crafting an overly complex and detailed picture. Keep it simple!

Mermaid provides a simple, open and transparent method for the SIG communities to add, edit and collaborate on diagrams for new or existing documentation.

Note:

You can still use Mermaid to create/edit diagrams even if it's not supported in your environment. This method is called **Mermaid+SVG** and is explained below.

Live editor

The Mermaid live editor is a web-based tool that enables you to create, edit and review diagrams.

The following lists live editor functions:

- Displays Mermaid code and rendered diagram.
- Generates a URL for each saved diagram. The URL is displayed in the URL field of your browser. You can share the URL with colleagues who can access and modify the diagram.
- Option to download `.svg` or `.png` files.

Note:

The live editor is the easiest and fastest way to create and edit Mermaid diagrams.

Methods for creating diagrams

Figure 2 outlines the three methods to generate and add diagrams.

graph TB A[Contributor] B[Inline

Mermaid code
added to .md file] C[Mermaid+SVG

Add mermaid-generated
svg file to .md file] D[External tool

Add external-tool-
generated svg file
to .md file] A --> B A --> C A --> D classDef box fill:#fff,stroke:#000,stroke-width:1px,color:#000; class A,B,C,D box %% you can hyperlink Mermaid diagram nodes to a URL using click statements click A "https://mermaid-js.github.io/mermaid-live-editor/edit/#eyJjb2RlIjoiZ3JhcGggVEJcbiAgICBBBW0NvbnRyaWJ1dG9yXVxuICAgIEJbSW5saW5lPGJyPjxicj5NZXJtYWlkIGNvZGU8YnI-YWRkZWQgdG8gLm1kIGZpbGVdXG4gICAgQ1tNZXJtYWlkK1NWRzxicj48YnI-QWRkIG1lcm1haWQtZ2VuZXJhdGVkPGJyPnN2ZyBmaWxlIHRvIC5tZCBmaWxlXSAgICAgICAgXG4gICAgRGRbRXh0ZXJuYWwgdG9vbDxicj48YnI-QWRkIGV4dGVybmFsLXRvb2wtPGJyPmdlbmVyYXRlZCBzdmcgZmlsZTxicj50byAubWQgZmlsZV0gICAgICAgIFxuICAgIEEgLT4gQlxuICAgIEEgLT4gQ1xuICAgIEEgLT4gRFxuICAgIGNsYXNzRGVmIGJveCBmaWxsOiNmZmYsc3Ryb2tlOiMwMDAsc3Ryb2tlLXdpZHRoOjFweCxjb2xvcjojMDAwO1xuICAgIGNsYXNzIEEsQixDLEQgYm94Il0In0
_blank click B "https://mermaid-js.github.io/mermaid-live-editor/edit/#eyJjb2RlIjoiZ3JhcGggVEJcbiAgICBBBW0NvbnRyaWJ1dG9yXVxuICAgIEJbSW5saW5lPGJyPjxicj5NZXJtYWlkIGNvZGU8YnI-YWRkZWQgdG8gLm1kIGZpbGVdXG4gICAgQ1tNZXJtYWlkK1NWRzxicj48YnI-QWRkIG1lcm1haWQtZ2VuZXJhdGVkPGJyPnN2ZyBmaWxlIHRvIC5tZCBmaWxlXSAgICAgICAgXG4gICAgRGRbRXh0ZXJuYWwgdG9vbDxicj48YnI-QWRkIGV4dGVybmFsLXRvb2wtPGJyPmdlbmVyYXRlZCBzdmcgZmlsZTxicj50byAubWQgZmlsZV0gICAgICAgIFxuICAgIEEgLT4gQlxuICAgIEEgLT4gQ1xuICAgIEEgLT4gRFxuICAgIGNsYXNzRGVmIGJveCBmaWxsOiNmZmYsc3Ryb2tlOiMwMDAsc3Ryb2tlLXdpZHRoOjFweCxjb2xvcjojMDAwO1xuICAgIGNsYXNzIEEsQixDLEQgYm94Il0In0
_blank click C "https://mermaid-js.github.io/mermaid-live-editor/edit/#eyJjb2RlIjoiZ3JhcGggVEJcbiAgICBBBW0NvbnRyaWJ1dG9yXVxuICAgIEJbSW5saW5lPGJyPjxicj5NZXJtYWlkIGNvZGU8YnI-YWRkZWQgdG8gLm1kIGZpbGVdXG4gICAgQ1tNZXJtYWlkK1NWRzxicj48YnI-QWRkIG1lcm1haWQtZ2VuZXJhdGVkPGJyPnN2ZyBmaWxlIHRvIC5tZCBmaWxlXSAgICAgICAgXG4gICAgRGRbRXh0ZXJuYWwgdG9vbDxicj48YnI-QWRkIGV4dGVybmFsLXRvb2wtPGJyPmdlbmVyYXRlZCBzdmcgZmlsZTxicj50byAubWQgZmlsZV0gICAgICAgIFxuICAgIEEgLT4gQlxuICAgIEEgLT4gQ1xuICAgIEEgLT4gRFxuICAgIGNsYXNzRGVmIGJveCBmaWxsOiNmZmYsc3Ryb2tlOiMwMDAsc3Ryb2tlLXdpZHRoOjFweCxjb2xvcjojMDAwO1xuICAgIGNsYXNzIEEsQixDLEQgYm94Il0In0
_blank click D "https://mermaid-js.github.io/mermaid-live-editor/edit/#eyJjb2RlIjoiZ3JhcGggVEJcbiAgICBBBW0NvbnRyaWJ1dG9yXVxuICAgIEJbSW5saW5lPGJyPjxicj5NZXJtYWlkIGNvZGU8YnI-YWRkZWQgdG8gLm1kIGZpbGVdXG4gICAgQ1tNZXJtYWlkK1NWRzxicj48YnI-QWRkIG1lcm1haWQtZ2VuZXJhdGVkPGJyPnN2ZyBmaWxlIHRvIC5tZCBmaWxlXSAgICAgICAgXG4gICAgRGRbRXh0ZXJuYWwgdG9vbDxicj48YnI-QWRkIGV4dGVybmFsLXRvb2wtPGJyPmdlbmVyYXRlZCBzdmcgZmlsZTxicj50byAubWQgZmlsZV0gICAgICAgIFxuICAgIEEgLT4gQlxuICAgIEEgLT4gQ1xuICAgIEEgLT4gRFxuICAgIGNsYXNzRGVmIGJveCBmaWxsOiNmZmYsc3Ryb2tlOiMwMDAsc3Ryb2tlLXdpZHRoOjFweCxjb2xvcjojMDAwO1xuICAgIGNsYXNzIEEsQixDLEQgYm94Il0In0
_blank

Figure 2. Methods to create diagrams.

Inline

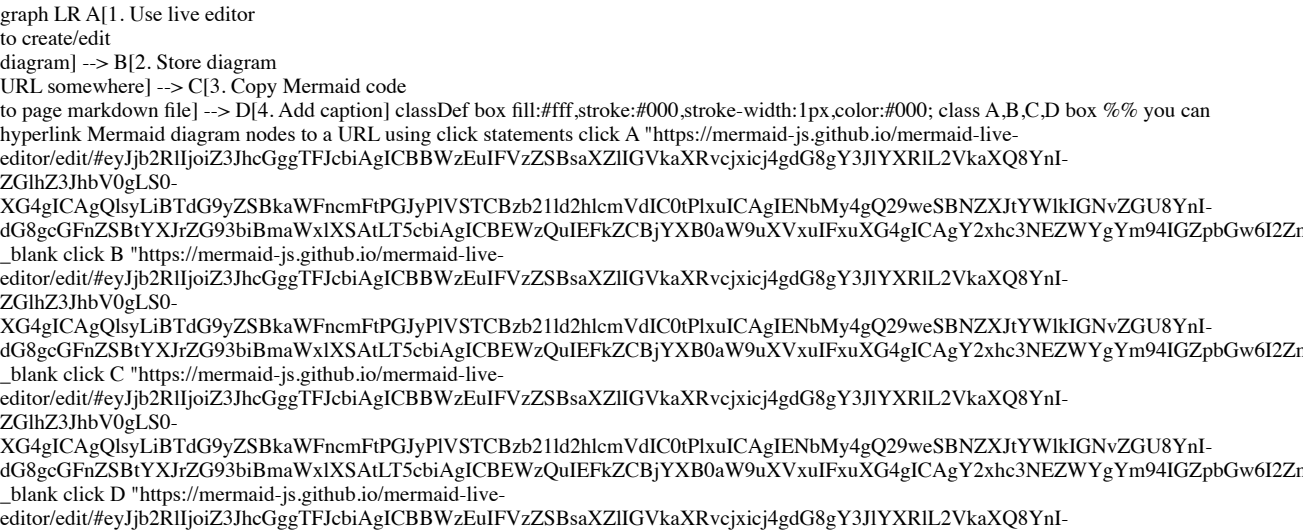Figure 3 outlines the steps to follow for adding a diagram using the Inline method.

graph LR A[1. Use live editor
to create/edit
diagram] --> B[2. Store diagram
URL somewhere] --> C[3. Copy Mermaid code
to page markdown file] --> D[4. Add caption] classDef box fill:#fff,stroke:#000,stroke-width:1px,color:#000; class A,B,C,D box %% you can hyperlink Mermaid diagram nodes to a URL using click statements click A "https://mermaid-js.github.io/mermaid-live-editor/edit/#eyJjb2RlIjoiZ3JhcGggTFJcbiAgICBBBW0NBBWzEuIFVzZSBsaXZlIGVkaXRvcjxicj4gdG8gY3JlYXRlL2VkaXQ8YnI-ZGlhZ3JhbV0gLS0-XG4gICAgQlsyLiBTdG9yZSBkaWFncmFtPGJyPlVSTCBzb21ld2hlcmVdIC0tPlxuICAgIENbMy4gQ29weSBNZXJtYWlkIGNvZGU8YnI-dG8gcGFnZSBtYXJrZG93biBmaWxlXSAtLT5cbiAgICBEWzQuIEFkZCBjYXB0aW9uXVxuICAgIGNsYXNzRGVmIGJveCBmaWxsOiNmZmYsc3Ryb2tlOiMwMDAsc3Ryb2tlLXdpZHRoOjFweCxjb2xvcjojMDAwO1xuICAgIGNsYXNzIEEsQixDLEQgYm94Il0In0
_blank click B "https://mermaid-js.github.io/mermaid-live-editor/edit/#eyJjb2RlIjoiZ3JhcGggTFJcbiAgICBBBW0NBBWzEuIFVzZSBsaXZlIGVkaXRvcjxicj4gdG8gY3JlYXRlL2VkaXQ8YnI-ZGlhZ3JhbV0gLS0-XG4gICAgQlsyLiBTdG9yZSBkaWFncmFtPGJyPlVSTCBzb21ld2hlcmVdIC0tPlxuICAgIENbMy4gQ29weSBNZXJtYWlkIGNvZGU8YnI-dG8gcGFnZSBtYXJrZG93biBmaWxlXSAtLT5cbiAgICBEWzQuIEFkZCBjYXB0aW9uXVxuICAgIGNsYXNzRGVmIGJveCBmaWxsOiNmZmYsc3Ryb2tlOiMwMDAsc3Ryb2tlLXdpZHRoOjFweCxjb2xvcjojMDAwO1xuICAgIGNsYXNzIEEsQixDLEQgYm94Il0In0
_blank click C "https://mermaid-js.github.io/mermaid-live-editor/edit/#eyJjb2RlIjoiZ3JhcGggTFJcbiAgICBBBW0NBBWzEuIFVzZSBsaXZlIGVkaXRvcjxicj4gdG8gY3JlYXRlL2VkaXQ8YnI-ZGlhZ3JhbV0gLS0-XG4gICAgQlsyLiBTdG9yZSBkaWFncmFtPGJyPlVSTCBzb21ld2hlcmVdIC0tPlxuICAgIENbMy4gQ29weSBNZXJtYWlkIGNvZGU8YnI-dG8gcGFnZSBtYXJrZG93biBmaWxlXSAtLT5cbiAgICBEWzQuIEFkZCBjYXB0aW9uXVxuICAgIGNsYXNzRGVmIGJveCBmaWxsOiNmZmYsc3Ryb2tlOiMwMDAsc3Ryb2tlLXdpZHRoOjFweCxjb2xvcjojMDAwO1xuICAgIGNsYXNzIEEsQixDLEQgYm94Il0In0
_blank click D "https://mermaid-js.github.io/mermaid-live-editor/edit/#eyJjb2RlIjoiZ3JhcGggTFJcbiAgICBBBW0NBBWzEuIFVzZSBsaXZlIGVkaXRvcjxicj4gdG8gY3JlYXRlL2VkaXQ8YnI-ZGlhZ3JhbV0gLS0-XG4gICAgQlsyLiBTdG9yZSBkaWFncmFtPGJyPlVSTCBzb21ld2hlcmVdIC0tPlxuICAgIENbMy4gQ29weSBNZXJtYWlkIGNvZGU8YnI-dG8gcGFnZSBtYXJrZG93biBmaWxlXSAtLT5cbiAgICBEWzQuIEFkZCBjYXB0aW9uXVxuICAgIGNsYXNzRGVmIGJveCBmaWxsOiNmZmYsc3Ryb2tlOiMwMDAsc3Ryb2tlLXdpZHRoOjFweCxjb2xvcjojMDAwO1xuICAgIGNsYXNzIEEsQixDLEQgYm94Il0In0
_blank

ZGlhZ3JhbV0gLS0-
XG4gICAgQlsyLiBTdG9yZSBkaWFncmFtPGJyPlVSTCBzb21ld2hlcmVdIC0tPlxuICAgIENbMy4gQ29weSBNZXJtYWlkIGNvZGU8YnI-
dG8gcGFnZSBtYXJrZG93biBmaWxlXSAtLT5cbiAgICBEWzQuIEFkZCBjYXB0aW9uXVxuXG4gICAgY2xhc3NEZWYgYm94IGZpbGw6I2Zr
_blank

Figure 3. Inline Method steps.

The following lists the steps you should follow for adding a diagram using the Inline method:

1. Create your diagram using the live editor.
2. Store the diagram URL somewhere for later access.
3. Copy the mermaid code to the location in your `.md` file where you want the diagram to appear.
4. Add a caption below the diagram using Markdown text.

A Hugo build runs the Mermaid code and turns it into a diagram.

Note:

You may find keeping track of diagram URLs is cumbersome. If so, make a note in the `.md` file that the Mermaid code is self-documenting. Contributors can copy the Mermaid code to and from the live editor for diagram edits.

Here is a sample code snippet contained in an `.md` file:

```
---
title: My PR
---
Figure 17 shows a simple A to B process.
some markdown text
...
{{< mermaid >}}
    graph TB
    A --> B
{{< /mermaid >}}

Figure 17. A to B
more text
```

Note:

You must include the Hugo Mermaid shortcode tags at the start and end of the Mermaid code block. You should add a diagram caption below the diagram.

For more details on diagram captions, see How to use captions.

The following lists advantages of the Inline method:

- Live editor tool.
- Easy to copy Mermaid code to and from the live editor and your `.md` file.
- No need for separate `.svg` image file handling.
- Content text, diagram code and diagram caption contained in the same `.md` file.

You should use the local and Netlify previews to verify the diagram is properly rendered.

Caution:

The Mermaid live editor feature set may not support the kubernetes/website Mermaid feature set.
And please, note that contributors can mention `kubernetes/website` as `k/website`. You might see a syntax error or a blank screen after the Hugo build. If that is the case, consider using the Mermaid+SVG method.

Mermaid+SVG

Figure 4 outlines the steps to follow for adding a diagram using the Mermaid+SVG method.

flowchart LR A[1. Use live editor
to create/edit
diagram] B[2. Store diagram
URL somewhere] C[3. Generate .svg file
and download to
images/ folder] subgraph w[ ] direction TB D[4. Use figure shortcode
to reference .svg
file in page
.md file] --> E[5. Add caption] end A --> B B --> C C --> w classDef box fill:#fff,stroke:#000,stroke-width:1px,color:#000; class A,B,C,D,E,w
box click A "https://mermaid-js.github.io/mermaid-live-
editor/edit/#eyJjb2RlIjoiZmxvd2NoYXJ0IExSXG4gICAgQVsxLiBVc2UgbGl2ZSBlZGl0b3I8YnI-
IHRvIGNyZWF0ZS9lZGl0PGJyPmRpYWdyYW1dXG4gICAgQlsyLiBTdG9yZSBkaWFncmFtPGJyPlVSTCBzb21ld2hlcmVdXG4gICAgQ1szLiBHZ
YW5kIGRvd25sb2FkIHRvPGJyPmltYWdlcy8gZm9sZGVyXVxuICAgIHN1YmdyYXBoIHdbIF1cbiAgICBkaXJlY3Rpb24gVEJcbiAgICBEWzQuIFV
XG4gICAgRVs1LiBBZGQgY2FwdGlvbl1cbiAgICBlbmRcbkEgLS0-IEJcbkIgLS0-IENcbkMgLS0-
IHdcbiAgICBjbGFzc0RlZiBib3ggZmlsbDojNmZmYysc3Ryb2tlOiMwMDAsc3Ryb2tlLXdpZHRoOjFweCxjb2xvcjojMDAwO1xuICAgIGNsYX
_blank click B "https://mermaid-js.github.io/mermaid-live-
editor/edit/#eyJjb2RlIjoiZmxvd2NoYXJ0IExSXG4gICAgQVsxLiBVc2UgbGl2ZSBlZGl0b3I8YnI-
IHRvIGNyZWF0ZS9lZGl0PGJyPmRpYWdyYW1dXG4gICAgQlsyLiBTdG9yZSBkaWFncmFtPGJyPlVSTCBzb21ld2hlcmVdXG4gICAgQ1szLiBHZ
YW5kIGRvd25sb2FkIHRvPGJyPmltYWdlcy8gZm9sZGVyXVxuICAgIHN1YmdyYXBoIHdbIF1cbiAgICBkaXJlY3Rpb24gVEJcbiAgICBEWzQuIFV
XG4gICAgRVs1LiBBZGQgY2FwdGlvbl1cbiAgICBlbmRcbkEgLS0-IEJcbkIgLS0-IENcbkMgLS0-
IHdcbiAgICBjbGFzc0RlZiBib3ggZmlsbDojNmZmYysc3Ryb2tlOiMwMDAsc3Ryb2tlLXdpZHRoOjFweCxjb2xvcjojMDAwO1xuICAgIGNsYX

```
_blank click C "https://mermaid-js.github.io/mermaid-live-
editor/edit/#eyJjb2RlIjoiZmxvd2NoYXJ0IExSXG4gICAgQVsxLiBVc2UgbGl2ZSBlZGl0b3I8YnI-
IHRvIGNyZWF0ZS9lZGl0PGJyPmRpYWdyYW1dXG4gICAgQlsyLiBTdG9yZSBkaWFncmFtPGJyPlVSTCBzb21ld2hlcmVcdXG4gICAgQ1szLiBHZ
YW5kIGRvd25sb2FkIHRvPGJyPmltYWdlcy8gZm9sZGVyXVxuICAgIHN1YmdyYXBoIHdbIF1cbiAgICBkaXJlY3Rpb24gVEJcbiAgICBEEWzQuIFV
XG4gICAgRVs1LiBBZGQgQgY2FwdGlvbl1cbiAgICBlbmRcbiAgICBBLS0-IEJcbkIgLS0-IENcbkMgLS0-
IHdcblxuICAgIGNsYXNzRGVmIGJveCBmaWxsOiNmZmYsc3Ryb2tlOiMwMDAsc3Ryb2tlLXdpZHRoOjFweeCxjb3xvcjojMDAwO1xuICAgIGNsY]
_blank click D "https://mermaid-js.github.io/mermaid-live-
editor/edit/#eyJjb2RlIjoiZmxvd2NoYXJ0IExSXG4gICAgQVsxLiBVc2UgbGl2ZSBlZGl0b3I8YnI-
IHRvIGNyZWF0ZS9lZGl0PGJyPmRpYWdyYW1dXG4gICAgQlsyLiBTdG9yZSBkaWFncmFtPGJyPlVSTCBzb21ld2hlcmVcdXG4gICAgQ1szLiBHZ
YW5kIGRvd25sb2FkIHRvPGJyPmltYWdlcy8gZm9sZGVyXVxuICAgIHN1YmdyYXBoIHdbIF1cbiAgICBkaXJlY3Rpb24gVEJcbiAgICBEEWzQuIFV
XG4gICAgRVs1LiBBZGQgQgY2FwdGlvbl1cbiAgICBlbmRcbiAgICBBLS0-IEJcbkIgLS0-IENcbkMgLS0-
IHdcblxuICAgIGNsYXNzRGVmIGJveCBmaWxsOiNmZmYsc3Ryb2tlOiMwMDAsc3Ryb2tlLXdpZHRoOjFweeCxjb3xvcjojMDAwO1xuICAgIGNsY]
_blank click E "https://mermaid-js.github.io/mermaid-live-
editor/edit/#eyJjb2RlIjoiZmxvd2NoYXJ0IExSXG4gICAgQVsxLiBVc2UgbGl2ZSBlZGl0b3I8YnI-
IHRvIGNyZWF0ZS9lZGl0PGJyPmRpYWdyYW1dXG4gICAgQlsyLiBTdG9yZSBkaWFncmFtPGJyPlVSTCBzb21ld2hlcmVcdXG4gICAgQ1szLiBHZ
YW5kIGRvd25sb2FkIHRvPGJyPmltYWdlcy8gZm9sZGVyXVxuICAgIHN1YmdyYXBoIHdbIF1cbiAgICBkaXJlY3Rpb24gVEJcbiAgICBEEWzQuIFV
XG4gICAgRVs1LiBBZGQgQgY2FwdGlvbl1cbiAgICBlbmRcbiAgICBBLS0-IEJcbkIgLS0-IENcbkMgLS0-
IHdcblxuICAgIGNsYXNzRGVmIGJveCBmaWxsOiNmZmYsc3Ryb2tlOiMwMDAsc3Ryb2tlLXdpZHRoOjFweeCxjb3xvcjojMDAwO1xuICAgIGNsY]
_blank
```

Figure 4. Mermaid+SVG method steps.

The following lists the steps you should follow for adding a diagram using the Mermaid+SVG method:

1. Create your diagram using the live editor.
2. Store the diagram URL somewhere for later access.
3. Generate an `.svg` image file for the diagram and download it to the appropriate `images/` folder.
4. Use the `{{< figure >}}` shortcode to reference the diagram in the `.md` file.
5. Add a caption using the `{{< figure >}}` shortcode's `caption` parameter.

For example, use the live editor to create a diagram called `boxnet`. Store the diagram URL somewhere for later access. Generate and download a `boxnet.svg` file to the appropriate `../images/` folder.

Use the `{{< figure >}}` shortcode in your PR's `.md` file to reference the `.svg` image file and add a caption.

```
{{< figure src="/static/images/boxnet.svg" alt="Boxnet figure" class="diagram-large" caption="Figure 14. Boxnet caption" >}}
```

For more details on diagram captions, see <u>How to use captions</u>.

Note:

The figure shortcode is the preferred method for adding `.svg` image files to your documentation. You can also use the standard markdown image syntax like so: `![my boxnet diagram](static/images/boxnet.svg)`. And you will need to add a caption below the diagram.

You should add the live editor URL as a comment block in the `.svg` image file using a text editor. For example, you would include the following at the beginning of the `.svg` image file:

```
<!-- To view or edit the mermaid code, use the following URL: -->
<!-- https://mermaid-js.github.io/mermaid-live-editor/edit/#eyJjb ... <remainder of the URL> -->
```

The following lists advantages of the Mermaid+SVG method:

- Live editor tool.
- Live editor tool supports the most current Mermaid feature set.
- Employ existing <u>kubernetes/website</u> methods for handling `.svg` image files.
- Environment doesn't require Mermaid support.

Be sure to check that your diagram renders properly using the <u>local</u> and Netlify previews.

External tool

Figure 5 outlines the steps to follow for adding a diagram using the External Tool method.

First, use your external tool to create the diagram and save it as an `.svg` or `.png` image file. After that, use the same steps as the **Mermaid+SVG** method for adding `.svg` image files.

```
flowchart LR A[1. Use external
tool to create/edit
diagram] B[2. If possible, save
diagram coordinates
for contributor
access] C[3. Generate .svg
or.png file
and download to
appropriate
images/ folder] subgraph w[ ] direction TB D[4. Use figure shortcode
to reference svg or
png file in
page .md file] --> E[5. Add caption] end A --> B B --> C C --> w classDef box fill:#fff,stroke:#000,stroke-width:1px,color:#000; class
A,B,C,D,E,w box click A "https://mermaid-js.github.io/mermaid-live-
editor/edit/#eyJjb2RlIjoiZmxvd2NoYXJ0IExSXG4gICAgQVsxLiBVc2UgZXh0ZXJuYWw8YnI-
dG9vbCB0byByBjcmVhdGUvZWRpdDxicj5kaWFncmFtXTXVxuICAgIEJbMi4gSWYgcG9zc2libGUsIHNhdmU8YnI-
```

ZGlhZ3JhbSBjb29yZGluYXRlczxicj5IgY29udHJpYnV0b3I8YnI-YWNjZXNzLVxuICAgIENbMy4gR2VuZXJhdGUgLnN2ZyA8YnI-
b3IucG5nIGZpbGU8YnI-
YW5kIGRvd25sb2FkIHRvPGJyPmFwcHJvcHJpYXRlPGJyPmltYWdlcy8gZm9sZGVyXVxuICAgIHN1YmdyYXBoIHdpdGhpbiAgICBkaXJlY3Rpb24
cG5nIGZpbGUgaW4-48YnI-cGFnZSAubWQgZmlsZV0LS0-
XG4gICAgRVVs1LiBBBZGQgY2FwdGlvbl1cbiAgICBClbmRcbiAgBBIC0tPiBCXG4gIAgQiAtLT4gQ1xuICAgIEMgLS0-
IHdcbiAgICBjbGFzc0RlZiBib3ggZmlsbsbDojZmZmLHN0cm9ZTojMDAwLHN0cm9rZS13aWR0aDpHgY29sb3I6IzAwMDtcbiAgICBjbGFzcyBBBL
click B "https://mermaid-js.github.io/mermaid-live-
editor/edit/#eyJjb2RlIjoiZmxvd2NoYXJ0IExSXG4gICAgQVsxLiBVc2UgZXh0ZXJuYWw8YnI-
dG9vbCByoyBjcmVhdGUvZWRpdDxicj5kaWFncmFtXVxuICAgIEJbMi4gSWYgcG9zc2libGUsIHNhdmU8YnI-
ZGlhZ3JhbSBjb29yZGluYXRlczxicj5IgY29udHJpYnV0b3I8YnI-YWNjZXNzLVxuICAgIENbMy4gR2VuZXJhdGUgLnN2ZyA8YnI-
b3IucG5nIGZpbGU8YnI-
YW5kIGRvd25sb2FkIHRvPGJyPmFwcHJvcHJpYXRlPGJyPmltYWdlcy8gZm9sZGVyXVxuICAgIHN1YmdyYXBoIHdpdGhpbiAgICBkaXJlY3Rpb24
cG5nIGZpbGUgaW4-48YnI-cGFnZSAubWQgZmlsZV0LS0-
XG4gICAgRVVs1LiBBBZGQgY2FwdGlvbl1cbiAgICBClbmRcbiAgBBIC0tPiBCXG4gIAgQiAtLT4gQ1xuICAgIEMgLS0-
IHdcbiAgICBjbGFzc0RlZiBib3ggZmlsbsbDojZmZmLHN0cm9ZTojMDAwLHN0cm9rZS13aWR0aDpHgY29sb3I6IzAwMDtcbiAgICBjbGFzcyBBBL
click C "https://mermaid-js.github.io/mermaid-live-
editor/edit/#eyJjb2RlIjoiZmxvd2NoYXJ0IExSXG4gICAgQVsxLiBVc2UgZXh0ZXJuYWw8YnI-
dG9vbCByoyBjcmVhdGUvZWRpdDxicj5kaWFncmFtXVxuICAgIEJbMi4gSWYgcG9zc2libGUsIHNhdmU8YnI-
ZGlhZ3JhbSBjb29yZGluYXRlczxicj5IgY29udHJpYnV0b3I8YnI-YWNjZXNzLVxuICAgIENbMy4gR2VuZXJhdGUgLnN2ZyA8YnI-
b3IucG5nIGZpbGU8YnI-
YW5kIGRvd25sb2FkIHRvPGJyPmFwcHJvcHJpYXRlPGJyPmltYWdlcy8gZm9sZGVyXVxuICAgIHN1YmdyYXBoIHdpdGhpbiAgICBkaXJlY3Rpb24
cG5nIGZpbGUgaW4-48YnI-cGFnZSAubWQgZmlsZV0LS0-
XG4gICAgRVVs1LiBBBZGQgY2FwdGlvbl1cbiAgICBClbmRcbiAgBBIC0tPiBCXG4gIAgQiAtLT4gQ1xuICAgIEMgLS0-
IHdcbiAgICBjbGFzc0RlZiBib3ggZmlsbsbDojZmZmLHN0cm9ZTojMDAwLHN0cm9rZS13aWR0aDpHgY29sb3I6IzAwMDtcbiAgICBjbGFzcyBBBL
click D "https://mermaid-js.github.io/mermaid-live-
editor/edit/#eyJjb2RlIjoiZmxvd2NoYXJ0IExSXG4gICAgQVsxLiBVc2UgZXh0ZXJuYWw8YnI-
dG9vbCByoyBjcmVhdGUvZWRpdDxicj5kaWFncmFtXVxuICAgIEJbMi4gSWYgcG9zc2libGUsIHNhdmU8YnI-
ZGlhZ3JhbSBjb29yZGluYXRlczxicj5IgY29udHJpYnV0b3I8YnI-YWNjZXNzLVxuICAgIENbMy4gR2VuZXJhdGUgLnN2ZyA8YnI-
b3IucG5nIGZpbGU8YnI-
YW5kIGRvd25sb2FkIHRvPGJyPmFwcHJvcHJpYXRlPGJyPmltYWdlcy8gZm9sZGVyXVxuICAgIHN1YmdyYXBoIHdpdGhpbiAgICBkaXJlY3Rpb24
cG5nIGZpbGUgaW4-48YnI-cGFnZSAubWQgZmlsZV0LS0-
XG4gICAgRVVs1LiBBBZGQgY2FwdGlvbl1cbiAgICBClbmRcbiAgBBIC0tPiBCXG4gIAgQiAtLT4gQ1xuICAgIEMgLS0-
IHdcbiAgICBjbGFzc0RlZiBib3ggZmlsbsbDojZmZmLHN0cm9ZTojMDAwLHN0cm9rZS13aWR0aDpHgY29sb3I6IzAwMDtcbiAgICBjbGFzcyBBBL
click E "https://mermaid-js.github.io/mermaid-live-
editor/edit/#eyJjb2RlIjoiZmxvd2NoYXJ0IExSXG4gICAgQVsxLiBVc2UgZXh0ZXJuYWw8YnI-
dG9vbCByoyBjcmVhdGUvZWRpdDxicj5kaWFncmFtXVxuICAgIEJbMi4gSWYgcG9zc2libGUsIHNhdmU8YnI-
ZGlhZ3JhbSBjb29yZGluYXRlczxicj5IgY29udHJpYnV0b3I8YnI-YWNjZXNzLVxuICAgIENbMy4gR2VuZXJhdGUgLnN2ZyA8YnI-
b3IucG5nIGZpbGU8YnI-
YW5kIGRvd25sb2FkIHRvPGJyPmFwcHJvcHJpYXRlPGJyPmltYWdlcy8gZm9sZGVyXVxuICAgIHN1YmdyYXBoIHdpdGhpbiAgICBkaXJlY3Rpb24
cG5nIGZpbGUgaW4-48YnI-cGFnZSAubWQgZmlsZV0LS0-
XG4gICAgRVVs1LiBBBZGQgY2FwdGlvbl1cbiAgICBClbmRcbiAgBBIC0tPiBCXG4gIAgQiAtLT4gQ1xuICAgIEMgLS0-
IHdcbiAgICBjbGFzc0RlZiBib3ggZmlsbsbDojZmZmLHN0cm9ZTojMDAwLHN0cm9rZS13aWR0aDpHgY29sb3I6IzAwMDtcbiAgICBjbGFzcyBBBL

Figure 5. External Tool method steps

The following lists the steps you should follow for adding a diagram using the External Tool method:

1. Use your external tool to create a diagram.
2. Save the diagram coordinates for contributor access. For example, your tool may offer a link to the diagram image, or you could place the source code file, such as an `.xml` file, in a public repository for later contributor access.
3. Generate and save the diagram as an `.svg` or `.png` image file. Download this file to the appropriate `../images/` folder.
4. Use the `{{< figure >}}` shortcode to reference the diagram in the `.md` file.
5. Add a caption using the `{{< figure >}}` shortcode's `caption` parameter.

Here is the `{{< figure >}}` shortcode for the `images/apple.svg` diagram:

```
{{< figure src="/static/images/apple.svg" alt="red-apple-figure" class="diagram-large" caption="Figure 9. A Big Red Apple" >}}
```

If your external drawing tool permits:

- You can incorporate multiple `.svg` or `.png` logos, icons and images into your diagram. However, make sure you observe copyright and follow the Kubernetes documentation guidelines on the use of third party content.
- You should save the diagram source coordinates for later contributor access. For example, your tool may offer a link to the diagram image, or you could place the source code file, such as an `.xml` file, somewhere for contributor access.

For more information on K8s and CNCF logos and images, check out CNCF Artwork.

The following lists advantages of the External Tool method:

- Contributor familiarity with external tool.
- Diagrams require more detail than what Mermaid can offer.

Don't forget to check that your diagram renders correctly using the local and Netlify previews.

Examples

This section shows several examples of Mermaid diagrams.

Note:

The code block examples omit the Hugo Mermaid shortcode tags. This allows you to copy the code block into the live editor to experiment on your own. Note that the live editor doesn't recognize Hugo shortcodes.

Example 1 - Pod topology spread constraints

Figure 6 shows the diagram appearing in the [Pod topology spread constraints](#) page.

> graph TB subgraph "zoneB" n3(Node3) n4(Node4) end subgraph "zoneA" n1(Node1) n2(Node2) end classDef plain fill:#ddd,stroke:#fff,stroke-width:4px,color:#000; classDef k8s fill:#326ce5,stroke:#fff,stroke-width:4px,color:#fff; classDef cluster fill:#fff,stroke:#bbb,stroke-width:2px,color:#326ce5; class n1,n2,n3,n4 k8s; class zoneA,zoneB cluster; click n3 "https://mermaid-js.github.io/mermaid-live-editor/edit/#eyJjb2RlIjoiZ3JhcGggVEJcbiAgICBzdWJncmFwaCBcInpvbmVCXCJcbiAgICAgICAgbjMoTm9kZTMpXG4gICAgICAgIG40KE5vZGU5vZGU _blank click n4 "https://mermaid-js.github.io/mermaid-live-editor/edit/#eyJjb2RlIjoiZ3JhcGggVEJcbiAgICBzdWJncmFwaCBcInpvbmVCXCJcbiAgICAgICAgbjMoTm9kZTMpXG4gICAgICAgIG40KE5vZGU5vZGU _blank click n1 "https://mermaid-js.github.io/mermaid-live-editor/edit/#eyJjb2RlIjoiZ3JhcGggVEJcbiAgICBzdWJncmFwaCBcInpvbmVCXCJcbiAgICAgICAgbjMoTm9kZTMpXG4gICAgICAgIG40KE5vZGU5vZGU _blank click n2 "https://mermaid-js.github.io/mermaid-live-editor/edit/#eyJjb2RlIjoiZ3JhcGggVEJcbiAgICBzdWJncmFwaCBcInpvbmVCXCJcbiAgICAgICAgbjMoTm9kZTMpXG4gICAgICAgIG40KE5vZGU5vZGU _blank

Figure 6. Pod Topology Spread Constraints.

Code block:

```
graph TB
    subgraph "zoneB"
        n3(Node3)
        n4(Node4)
    end
    subgraph "zoneA"
        n1(Node1)
        n2(Node2)
    end

    classDef plain fill:#ddd,stroke:#fff,stroke-width:4px,color:#000;
    classDef k8s fill:#326ce5,stroke:#fff,stroke-width:4px,color:#fff;
    classDef cluster fill:#fff,stroke:#bbb,stroke-width:2px,color:#326ce5;
    class n1,n2,n3,n4 k8s;
    class zoneA,zoneB cluster;
```

Example 2 - Ingress

Figure 7 shows the diagram appearing in the [What is Ingress](#) page.

> graph LR; client([client])-. Ingress-managed load balancer .->ingress[Ingress]; ingress-->|routing rule|service[Service]; subgraph cluster ingress; service-->pod1[Pod]; service-->pod2[Pod]; end classDef plain fill:#ddd,stroke:#fff,stroke-width:4px,color:#000; classDef k8s fill:#326ce5,stroke:#fff,stroke-width:4px,color:#fff; classDef cluster fill:#fff,stroke:#bbb,stroke-width:2px,color:#326ce5; class ingress,service,pod1,pod2 k8s; class client plain; class cluster cluster; click client "https://mermaid-js.github.io/mermaid-live-editor/edit/#eyJjb2RlIjoiZ3JhcGggIExSXG4gIGNsaWVudChbY2xpZW50XSktLiBJbmdyZXNzLW1hbmFnZWQgPGJyPiBsb2FkIGJhbGFuY2VyIC4t _blank click ingress "https://mermaid-js.github.io/mermaid-live-editor/edit/#eyJjb2RlIjoiZ3JhcGggIExSXG4gIGNsaWVudChbY2xpZW50XSktLiBJbmdyZXNzLW1hbmFnZWQgPGJyPiBsb2FkIGJhbGFuY2VyIC4t _blank click service "https://mermaid-js.github.io/mermaid-live-editor/edit/#eyJjb2RlIjoiZ3JhcGggIExSXG4gIGNsaWVudChbY2xpZW50XSktLiBJbmdyZXNzLW1hbmFnZWQgPGJyPiBsb2FkIGJhbGFuY2VyIC4t _blank click pod1 "https://mermaid-js.github.io/mermaid-live-editor/edit/#eyJjb2RlIjoiZ3JhcGggIExSXG4gIGNsaWVudChbY2xpZW50XSktLiBJbmdyZXNzLW1hbmFnZWQgPGJyPiBsb2FkIGJhbGFuY2VyIC4t _blank click pod2 "https://mermaid-js.github.io/mermaid-live-editor/edit/#eyJjb2RlIjoiZ3JhcGggIExSXG4gIGNsaWVudChbY2xpZW50XSktLiBJbmdyZXNzLW1hbmFnZWQgPGJyPiBsb2FkIGJhbGFuY2VyIC4t _blank

Figure 7. Ingress

Code block:

```
graph LR;
 client([client])-. Ingress-managed <br> load balancer .->ingress[Ingress];
 ingress-->|routing rule|service[Service];
 subgraph cluster
 ingress;
 service-->pod1[Pod];
 service-->pod2[Pod];
 end
 classDef plain fill:#ddd,stroke:#fff,stroke-width:4px,color:#000;
 classDef k8s fill:#326ce5,stroke:#fff,stroke-width:4px,color:#fff;
 classDef cluster fill:#fff,stroke:#bbb,stroke-width:2px,color:#326ce5;
 class ingress,service,pod1,pod2 k8s;
 class client plain;
 class cluster cluster;
```

Example 3 - K8s system flow

Figure 8 depicts a Mermaid sequence diagram showing the system flow between K8s components to start a container.

[K8s system flow diagram](#)

Figure 8. K8s system flow diagram

Code block:

```
%%{init:{"theme":"neutral"}}%%
sequenceDiagram
    actor me
    participant apiSrv as control plane<br><br>api-server
    participant etcd as control plane<br><br>etcd datastore
    participant cntrlMgr as control plane<br><br>controller<br>manager
    participant sched as control plane<br><br>scheduler
    participant kubelet as node<br><br>kubelet
    participant container as node<br><br>container<br>runtime
    me->>apiSrv: 1. kubectl create -f pod.yaml
    apiSrv-->>etcd: 2. save new state
    cntrlMgr->>apiSrv: 3. check for changes
    sched->>apiSrv: 4. watch for unassigned pods(s)
    apiSrv->>sched: 5. notify about pod w nodename=" "
    sched->>apiSrv: 6. assign pod to node
    apiSrv-->>etcd: 7. save new state
    kubelet->>apiSrv: 8. look for newly assigned pod(s)
    apiSrv->>kubelet: 9. bind pod to node
    kubelet->>container: 10. start container
    kubelet->>apiSrv: 11. update pod status
    apiSrv-->>etcd: 12. save new state
```

How to style diagrams

You can style one or more diagram elements using well-known CSS nomenclature. You accomplish this using two types of statements in the Mermaid code.

- `classDef` defines a class of style attributes.
- `class` defines one or more elements to apply the class to.

In the code for figure 7, you can see examples of both.

```
classDef k8s fill:#326ce5,stroke:#fff,stroke-width:4px,color:#fff; // defines style for the k8s class
class ingress,service,pod1,pod2 k8s; // k8s class is applied to elements ingress, service, pod1 and pod2.
```

You can include one or multiple `classDef` and `class` statements in your diagram. You can also use the official K8s `#326ce5` hex color code for K8s components in your diagram.

For more information on styling and classes, see Mermaid Styling and classes docs.

How to use captions

A caption is a brief description of a diagram. A title or a short description of the diagram are examples of captions. Captions aren't meant to replace explanatory text you have in your documentation. Rather, they serve as a "context link" between that text and your diagram.

The combination of some text and a diagram tied together with a caption help provide a concise representation of the information you wish to convey to the user.

Without captions, you are asking the user to scan the text above or below the diagram to figure out a meaning. This can be frustrating for the user.

Figure 9 lays out the three components for proper captioning: diagram, diagram caption and the diagram referral.

flowchart A[Diagram

Inline Mermaid or
SVG image files] B[Diagram Caption

Add Figure Number. and
Caption Text] C[Diagram Referral

Reference Figure Number
in text] classDef box fill:#fff,stroke:#000,stroke-width:1px,color:#000; class A,B,C box click A "https://mermaid-js.github.io/mermaid-live-editor/edit#eyJjb2RlIjoiZmxvd2NoYXJ0XG4gICAgQVtEaWFncmFmFtPGJyPjxicj5JbmxpbmUgTWVybWFpZCBvcjxicj5TVkcgaW1hZ2UgZmlsZXNdXPGJyPkFkZCBGaWd1cmUgTnVtYmVyLiBhbmQ8YnI-Q2FwdGlvbiBUZXh0XXuuICAgENbRGlhZ3JhbSBSZWZlcnJhbDxicj48YnI-UmVmZXJlbmVuY2UgRmlndXJlIE51bWJlcjxicj5pbiB0ZXh0XXuuXG4gICAgY2xhc3NEZWYgYm94IGZpbGw6I2ZmZixzdHJva2U6IzAwMCxzdHJva2H_blank click B "https://mermaid-js.github.io/mermaid-live-editor/edit#eyJjb2RlIjoiZmxvd2NoYXJ0XG4gICAgQVtEaWFncmFmFtPGJyPjxicj5JbmxpbmUgTWVybWFpZCBvcjxicj5TVkcgaW1hZ2UgZmlsZXNdXPGJyPkFkZCBGaWd1cmUgTnVtYmVyLiBhbmQ8YnI-Q2FwdGlvbiBUZXh0XXuuICAgENbRGlhZ3JhbSBSZWZlcnJhbDxicj48YnI-UmVmZXJlbmVuY2UgRmlndXJlIE51bWJlcjxicj5pbiB0ZXh0XXuuXG4gICAgY2xhc3NEZWYgYm94IGZpbGw6I2ZmZixzdHJva2U6IzAwMCxzdHJva2H_blank click C "https://mermaid-js.github.io/mermaid-live-editor/edit#eyJjb2RlIjoiZmxvd2NoYXJ0XG4gICAgQVtEaWFncmFmFtPGJyPjxicj5JbmxpbmUgTWVybWFpZCBvcjxicj5TVkcgaW1hZ2UgZmlsZXNdXPGJyPkFkZCBGaWd1cmUgTnVtYmVyLiBhbmQ8YnI-Q2FwdGlvbiBUZXh0XXuuICAgENbRGlhZ3JhbSBSZWZlcnJhbDxicj48YnI-UmVmZXJlbmVuY2UgRmlndXJlIE51bWJlcjxicj5pbiB0ZXh0XXuuXG4gICAgY2xhc3NEZWYgYm94IGZpbGw6I2ZmZixzdHJva2U6IzAwMCxzdHJva2H_blank

Figure 9. Caption Components.

Note:

You should always add a caption to each diagram in your documentation.

Diagram

The `Mermaid+SVG` and `External Tool` methods generate `.svg` image files.

Here is the `{{< figure >}}` shortcode for the diagram defined in an `.svg` image file saved to `/images/docs/components-of-kubernetes.svg`:

```
{{< figure src="/images/docs/components-of-kubernetes.svg" alt="Kubernetes pod running inside a cluster" class="diagram-large" capt
```

You should pass the `src`, `alt`, `class` and `caption` values into the `{{< figure >}}` shortcode. You can adjust the size of the diagram using `diagram-large`, `diagram-medium` and `diagram-small` classes.

Note:

Diagrams created using the `Inline` method don't use the figure shortcode. The Mermaid code defines how the diagram will render on your page.

See [Methods for creating diagrams](#) for more information on the different methods for creating diagrams.

Diagram Caption

Next, add a diagram caption.

If you define your diagram in an `.svg` image file, then you should use the `{{< figure >}}` shortcode's `caption` parameter.

```
{{< figure src="/images/docs/components-of-kubernetes.svg" alt="Kubernetes pod running inside a cluster" class="diagram-large" capt
```

If you define your diagram using inline Mermaid code, then you should use Markdown text.

```
Figure 4. Kubernetes Architecture Components
```

The following lists several items to consider when adding diagram captions:

- Use the `{{< figure >}}` shortcode to add a diagram caption for `Mermaid+SVG` and `External Tool` diagrams.
- Use simple Markdown text to add a diagram caption for the `Inline` method.
- Prepend your diagram caption with `Figure NUMBER.`. You must use `Figure` and the number must be unique for each diagram in your documentation page. Add a period after the number.
- Add your diagram caption text after the `Figure NUMBER.` on the same line. You must puncuate the caption with a period. Keep the caption text short.
- Position your diagram caption **BELOW** your diagram.

Diagram Referral

Finally, you can add a diagram referral. This is used inside your text and should precede the diagram itself. It allows a user to connect your text with the associated diagram. The `Figure NUMBER` in your referral and caption must match.

You should avoid using spatial references such as `..the image below..` or `..the following figure ..`

Here is an example of a diagram referral:

```
Figure 10 depicts the components of the Kubernetes architecture.
The control plane ...
```

Diagram referrals are optional and there are cases where they might not be suitable. If you are not sure, add a diagram referral to your text to see if it looks and sounds okay. When in doubt, use a diagram referral.

Complete picture

Figure 10 shows the Kubernetes Architecture diagram that includes the diagram, diagram caption and diagram referral. The `{{< figure >}}` shortcode renders the diagram, adds the caption and includes the optional `link` parameter so you can hyperlink the diagram. The diagram referral is contained in this paragraph.

Here is the `{{< figure >}}` shortcode for this diagram:

```
{{< figure src="/images/docs/components-of-kubernetes.svg" alt="Kubernetes pod running inside a cluster" class="diagram-large" capt
```

[Kubernetes pod running inside a cluster](#)

Figure 10. Kubernetes Architecture.

Tips

- Always use the live editor to create/edit your diagram.

- Always use Hugo local and Netlify previews to check out how the diagram appears in the documentation.

- Include diagram source pointers such as a URL, source code location, or indicate the code is self-documenting.

- Always use diagram captions.

- Very helpful to include the diagram `.svg` or `.png` image and/or Mermaid source code in issues and PRs.

- With the `Mermaid+SVG` and `External Tool` methods, use `.svg` image files because they stay sharp when you zoom in on the diagram.

- Best practice for `.svg` files is to load it into an SVG editing tool and use the "Convert text to paths" function. This ensures that the diagram renders the same on all systems, regardless of font availability and font rendering support.

- No Mermaid support for additional icons or artwork.

- Hugo Mermaid shortcodes don't work in the live editor.

- Any time you modify a diagram in the live editor, you **must** save it to generate a new URL for the diagram.

- Click on the diagrams in this section to view the code and diagram rendering in the live editor.

- Look over the source code of this page, `diagram-guide.md`, for more examples.

- Check out the [Mermaid docs](#) for explanations and examples.

Most important, **Keep Diagrams Simple**. This will save time for you and fellow contributors, and allow for easier reading by new and experienced users.

Documentation Style Guide

This page gives writing style guidelines for the Kubernetes documentation. These are guidelines, not rules. Use your best judgment, and feel free to propose changes to this document in a pull request.

For additional information on creating new content for the Kubernetes documentation, read the [Documentation Content Guide](#).

Changes to the style guide are made by SIG Docs as a group. To propose a change or addition, [add it to the agenda](#) for an upcoming SIG Docs meeting, and attend the meeting to participate in the discussion.

Note:

Kubernetes documentation uses [Goldmark Markdown Renderer](#) with some adjustments along with a few [Hugo Shortcodes](#) to support glossary entries, tabs, and representing feature state.

Language

Kubernetes documentation has been translated into multiple languages (see [Localization READMEs](#)).

The way of localizing the docs for a different language is described in [Localizing Kubernetes Documentation](#).

The English-language documentation uses U.S. English spelling and grammar.

Documentation formatting standards

Use upper camel case for API objects

When you refer specifically to interacting with an API object, use [UpperCamelCase](#), also known as Pascal case. You may see different capitalization, such as "configMap", in the [API Reference](#). When writing general documentation, it's better to use upper camel case, calling it "ConfigMap" instead.

When you are generally discussing an API object, use [sentence-style capitalization](#).

The following examples focus on capitalization. For more information about formatting API object names, review the related guidance on [Code Style](#).

Do	Don't
The HorizontalPodAutoscaler resource is responsible for ...	The Horizontal pod autoscaler is responsible for ...
A PodList object is a list of pods.	A Pod List object is a list of pods.
The Volume object contains a `hostPath` field.	The volume object contains a hostPath field.
Every ConfigMap object is part of a namespace.	Every configMap object is part of a namespace.
For managing confidential data, consider using the Secret API.	For managing confidential data, consider using the secret API.

Use angle brackets for placeholders

Use angle brackets for placeholders. Tell the reader what a placeholder represents, for example:

Display information about a pod:

```
kubectl describe pod <pod-name> -n <namespace>
```

If the namespace of the pod is `default`, you can omit the '-n' parameter.

Use bold for user interface elements

Do	Don't
Click **Fork**.	Click "Fork".
Select **Other**.	Select "Other".

Use italics to define or introduce new terms

Do	Don't
A *cluster* is a set of nodes ...	A "cluster" is a set of nodes ...
These components form the *control plane*.	These components form the **control plane**.

Use code style for filenames, directories, and paths

Do	Don't
Open the `envars.yaml` file.	Open the envars.yaml file.
Go to the `/docs/tutorials` directory.	Go to the /docs/tutorials directory.
Open the `/_data/concepts.yaml` file.	Open the /_data/concepts.yaml file.

Use the international standard for punctuation inside quotes

Do	Don't
events are recorded with an associated "stage".	events are recorded with an associated "stage."
The copy is called a "fork".	The copy is called a "fork."

Inline code formatting

Use code style for inline code, commands

For inline code in an HTML document, use the `<code>` tag. In a Markdown document, use the backtick (`` ` ``). However, API kinds such as StatefulSet or ConfigMap are written verbatim (no backticks); this allows using possessive apostrophes.

Do	Don't
The `kubectl run` command creates a Pod.	The "kubectl run" command creates a Pod.
The kubelet on each node acquires a Lease…	The kubelet on each node acquires a `Lease`…
A PersistentVolume represents durable storage…	A `PersistentVolume` represents durable storage…
The CustomResourceDefinition's `.spec.group` field…	The `CustomResourceDefinition.spec.group` field…
For declarative management, use `kubectl apply`.	For declarative management, use "kubectl apply".
Enclose code samples with triple backticks. (` ``` `)	Enclose code samples with any other syntax.
Use single backticks to enclose inline code. For example, `var example = true`.	Use two asterisks (`**`) or an underscore (`_`) to enclose inline code. For example, **var example = true**.
Use triple backticks before and after a multi-line block of code for fenced code blocks.	Use multi-line blocks of code to create diagrams, flowcharts, or other illustrations.
Use meaningful variable names that have a context.	Use variable names such as 'foo','bar', and 'baz' that are not meaningful and lack context.
Remove trailing spaces in the code.	Add trailing spaces in the code, where these are important, because the screen reader will read out the spaces as well.

Note:

The website supports syntax highlighting for code samples, but specifying a language is optional. Syntax highlighting in the code block should conform to the [contrast guidelines.](#)

Use code style for object field names and namespaces

Do	Don't
Set the value of the `replicas` field in the configuration file.	Set the value of the "replicas" field in the configuration file.
The value of the `exec` field is an ExecAction object.	The value of the "exec" field is an ExecAction object.
Run the process as a DaemonSet in the `kube-system` namespace.	Run the process as a DaemonSet in the kube-system namespace.

Use code style for Kubernetes command tool and component names

Do	Don't
The `kubelet` preserves node stability.	The kubelet preserves node stability.
The `kubectl` handles locating and authenticating to the API server.	The kubectl handles locating and authenticating to the apiserver.
Run the process with the certificate, `kube-apiserver --client-ca-file=FILENAME`.	Run the process with the certificate, kube-apiserver --client-ca-file=FILENAME.

Starting a sentence with a component tool or component name

Do	Don't
The `kubeadm` tool bootstraps and provisions machines in a cluster.	`kubeadm` tool bootstraps and provisions machines in a cluster.
The kube-scheduler is the default scheduler for Kubernetes.	kube-scheduler is the default scheduler for Kubernetes.

Use a general descriptor over a component name

Do	Don't
The Kubernetes API server offers an OpenAPI spec.	The apiserver offers an OpenAPI spec.
Aggregated APIs are subordinate API servers.	Aggregated APIs are subordinate APIServers.

Use normal style for string and integer field values

For field values of type string or integer, use normal style without quotation marks.

Do	Don't
Set the value of `imagePullPolicy` to Always.	Set the value of `imagePullPolicy` to "Always".
Set the value of `image` to nginx:1.16.	Set the value of `image` to `nginx:1.16`.
Set the value of the `replicas` field to 2.	Set the value of the `replicas` field to `2`.

However, consider quoting values where there is a risk that readers might confuse the value with an API kind.

Referring to Kubernetes API resources

This section talks about how we reference API resources in the documentation.

Clarification about "resource"

Kubernetes uses the word *resource* to refer to API resources. For example, the URL path `/apis/apps/v1/namespaces/default/deployments/my-app` represents a Deployment named "my-app" in the "default" [namespace](#). In HTTP jargon, [namespace](#) is a resource - the same way that all web URLs identify a resource.

Kubernetes documentation also uses "resource" to talk about CPU and memory requests and limits. It's very often a good idea to refer to API resources as "API resources"; that helps to avoid confusion with CPU and memory resources, or with other kinds of resource.

If you are using the lowercase plural form of a resource name, such as `deployments` or `configmaps`, provide extra written context to help readers understand what you mean. If you are using the term in a context where the UpperCamelCase name could work too, and there is a risk of ambiguity, consider using the API kind in UpperCamelCase.

When to use Kubernetes API terminologies

The different Kubernetes API terminologies are:

- *API kinds*: the name used in the API URL (such as `pods`, `namespaces`). API kinds are sometimes also called *resource types*.
- *API resource*: a single instance of an API kind (such as `pod`, `secret`).
- *Object*: a resource that serves as a "record of intent". An object is a desired state for a specific part of your cluster, which the Kubernetes control plane tries to maintain. All objects in the Kubernetes API are also resources.

For clarity, you can add "resource" or "object" when referring to an API resource in Kubernetes documentation. An example: write "a Secret object" instead of "a Secret". If it is clear just from the capitalization, you don't need to add the extra word.

Consider rephrasing when that change helps avoid misunderstandings. A common situation is when you want to start a sentence with an API kind, such as "Secret"; because English and other languages capitalize at the start of sentences, readers cannot tell whether you mean the API kind or the general concept. Rewording can help.

API resource names

Always format API resource names using [UpperCamelCase,](#) also known as PascalCase. Do not write API kinds with code formatting.

Don't split an API object name into separate words. For example, use PodTemplateList, not Pod Template List.

For more information about PascalCase and code formatting, review the related guidance on [Use upper camel case for API objects](#) and [Use code style for inline code, commands, and API objects.](#)

For more information about Kubernetes API terminologies, review the related guidance on [Kubernetes API terminology.](#)

Code snippet formatting

Don't include the command prompt

Do	Don't
`kubectl get pods`	`$ kubectl get pods`

Separate commands from output

Verify that the pod is running on your chosen node:

```
kubectl get pods --output=wide
```

The output is similar to this:

```
NAME      READY      STATUS      RESTARTS    AGE    IP          NODE
nginx     1/1        Running     0           13s    10.200.0.4  worker0
```

Versioning Kubernetes examples

Code examples and configuration examples that include version information should be consistent with the accompanying text.

If the information is version specific, the Kubernetes version needs to be defined in the `prerequisites` section of the [Task template](#) or the [Tutorial template](#). Once the page is saved, the `prerequisites` section is shown as **Before you begin**.

To specify the Kubernetes version for a task or tutorial page, include `min-kubernetes-server-version` in the front matter of the page.

If the example YAML is in a standalone file, find and review the topics that include it as a reference. Verify that any topics using the standalone YAML have the appropriate version information defined. If a stand-alone YAML file is not referenced from any topics, consider deleting it instead of updating it.

For example, if you are writing a tutorial that is relevant to Kubernetes version 1.8, the front-matter of your markdown file should look something like:

```
---
title: <your tutorial title here>min-kubernetes-server-version: v1.8---
```

In code and configuration examples, do not include comments about alternative versions. Be careful to not include incorrect statements in your examples as comments, such as:

```
apiVersion: v1 # earlier versions use...
kind: Pod...
```

Formulae and equations

You can use the Docsy support for [diagrams and formulae](#).

For example: \\(\frac{7}{9} \sqrt{K^8 s}\\), which renders as \(\frac{7}{9} \sqrt{K^8 s}\).

Prefer inline formulae where reasonable, but you can use a `math` block if that's likely to help readers.

Read the Docsy guide to find out what you need to change in your page to activate support; if you have problems, add `math: true` to the page [front matter](#) (you can do this even if you think the automatic activation should be enough).

Kubernetes.io word list

A list of Kubernetes-specific terms and words to be used consistently across the site.

Term	Usage
Kubernetes	Kubernetes should always be capitalized.
Docker	Docker should always be capitalized.
SIG Docs	SIG Docs rather than SIG-DOCS or other variations.
On-premises	On-premises or On-prem rather than On-premise or other variations.
cloud native	Cloud native or cloud native as appropriate for sentence structure rather than cloud-native or Cloud Native.
open source	Open source or open source as appropriate for sentence structure rather than open-source or Open Source.

Shortcodes

Hugo [Shortcodes](#) help create different rhetorical appeal levels. Our documentation supports three different shortcodes in this category: **Note** `{{< note >}}`, **Caution** `{{< caution >}}`, and **Warning** `{{< warning >}}`.

1. Surround the text with an opening and closing shortcode.

2. Use the following syntax to apply a style:

   ```
   {{< note >}}
   No need to include a prefix; the shortcode automatically provides one. (Note:, Caution:, etc.)
   {{< /note >}}
   ```

   The output is:

   Note:

   The prefix you choose is the same text for the tag.

Note

Use `{{< note >}}` to highlight a tip or a piece of information that may be helpful to know.

For example:

```
{{< note >}}
You can _still_ use Markdown inside these callouts.
{{< /note >}}
```

The output is:

Note:

You can *still* use Markdown inside these callouts.

You can use a `{{< note >}}` in a list:

```
1. Use the note shortcode in a list

1. A second item with an embedded note
```

```
{{< note >}}
Warning, Caution, and Note shortcodes, embedded in lists, need to be indented four spaces. See [Common Shortcode Issues](#commor
{{< /note >}}
```

```
1. A third item in a list
```

```
1. A fourth item in a list
```

The output is:

1. Use the note shortcode in a list

2. A second item with an embedded note

   Note:

   ```
   Warning, Caution, and Note shortcodes, embedded in lists, need to be indented four spaces. See [Common Shortcode Issues](#com
   ```

3. A third item in a list

4. A fourth item in a list

Caution

Use `{{< caution >}}` to call attention to an important piece of information to avoid pitfalls.

For example:

```
{{< caution >}}
The callout style only applies to the line directly above the tag.
{{< /caution >}}
```

The output is:

Caution:

The callout style only applies to the line directly above the tag.

Warning

Use `{{< warning >}}` to indicate danger or a piece of information that is crucial to follow.

For example:

```
{{< warning >}}
Beware.
{{< /warning >}}
```

The output is:

Warning:

Beware.

Common Shortcode Issues

Ordered Lists

Shortcodes will interrupt numbered lists unless you indent four spaces before the notice and the tag.

For example:

```
1. Preheat oven to 350˚F
```

```
1. Prepare the batter, and pour into springform pan.
   {{< note >}}Grease the pan for best results.{{< /note >}}
```

```
1. Bake for 20-25 minutes or until set.
```

The output is:

1. Preheat oven to 350˚F

2. Prepare the batter, and pour into springform pan.

   Note:

   Grease the pan for best results.

3. Bake for 20-25 minutes or until set.

Include Statements

Shortcodes inside include statements will break the build. You must insert them in the parent document, before and after you call the include. For example:

```
{{< note >}}
{{< include "task-tutorial-prereqs.md" >}}
{{< /note >}}
```

Markdown elements

Line breaks

Use a single newline to separate block-level content like headings, lists, images, code blocks, and others. The exception is second-level headings, where it should be two newlines. Second-level headings follow the first-level (or the title) without any preceding paragraphs or texts. A two line spacing helps visualize the overall structure of content in a code editor better.

Manually wrap paragraphs in the Markdown source when appropriate. Since the git tool and the GitHub website generate file diffs on a line-by-line basis, manually wrapping long lines helps the reviewers to easily find out the changes made in a PR and provide feedback. It also helps the downstream localization teams where people track the upstream changes on a per-line basis. Line wrapping can happen at the end of a sentence or a punctuation character, for example. One exception to this is that a Markdown link or a shortcode is expected to be in a single line.

Headings and titles

People accessing this documentation may use a screen reader or other assistive technology (AT). Screen readers are linear output devices, they output items on a page one at a time. If there is a lot of content on a page, you can use headings to give the page an internal structure. A good page structure helps all readers to easily navigate the page or filter topics of interest.

Do	Don't
Update the title in the front matter of the page or blog post.	Use first level heading, as Hugo automatically converts the title in the front matter of the page into a first-level heading.
Use ordered headings to provide a meaningful high-level outline of your content.	Use headings level 4 through 6, unless it is absolutely necessary. If your content is that detailed, it may need to be broken into separate articles.
Use pound or hash signs (#) for non-blog post content.	Use underlines (--- or ===) to designate first-level headings.
Use sentence case for headings in the page body. For example, **Extend kubectl with plugins**	Use title case for headings in the page body. For example, **Extend Kubectl With Plugins**
Use title case for the page title in the front matter. For example, `title: Kubernetes API Server Bypass Risks`	Use sentence case for page titles in the front matter. For example, don't use `title: Kubernetes API server bypass risks`
Place relevant links in the body copy.	Include hyperlinks (`<a href=""></a>`) in headings.
Use pound or hash signs (#) to indicate headings.	Use **bold** text or other indicators to split paragraphs.

Paragraphs

Do	Don't
Try to keep paragraphs under 6 sentences.	Indent the first paragraph with space characters. For example, ···Three spaces before a paragraph will indent it.
Use three hyphens (---) to create a horizontal rule. Use horizontal rules for breaks in paragraph content. For example, a change of scene in a story, or a shift of topic within a section.	Use horizontal rules for decoration.

Links

Do	Don't
Write hyperlinks that give you context for the content they link to. For example: Certain ports are open on your machines. See Check required ports for more details.	Use ambiguous terms such as "click here". For example: Certain ports are open on your machines. See here for more details.
Write Markdown-style links: `[link text](URL)`. For example: `[Hugo shortcodes](/docs/contribute/style/hugo-shortcodes/#table-captions)` and the output is Hugo shortcodes.	Write HTML-style links: `<a href="/media/examples/link-element-example.css" target="_blank">Visit our tutorial!</a>`, or create links that open in new tabs or windows. For example: `[example website](https://example.com){target="_blank"}`

Lists

Group items in a list that are related to each other and need to appear in a specific order or to indicate a correlation between multiple items. When a screen reader comes across a list—whether it is an ordered or unordered list—it will be announced to the user that there is a group of list items. The user can then use the arrow keys to move up and down between the various items in the list. Website navigation links can also be marked up as list items; after all they are nothing but a group of related links.

- End each item in a list with a period if one or more items in the list are complete sentences. For the sake of consistency, normally either all items or none should be complete sentences.

  Note:

  Ordered lists that are part of an incomplete introductory sentence can be in lowercase and punctuated as if each item was a part of the introductory sentence.

- Use the number one (`1.`) for ordered lists.

- Use (`+`), (`*`), or (`-`) for unordered lists.

- Leave a blank line after each list.

- Indent nested lists with four spaces (for example, $\cdots$).

- List items may consist of multiple paragraphs. Each subsequent paragraph in a list item must be indented by either four spaces or one tab.

Tables

The semantic purpose of a data table is to present tabular data. Sighted users can quickly scan the table but a screen reader goes through line by line. A table caption is used to create a descriptive title for a data table. Assistive technologies (AT) use the HTML table caption element to identify the table contents to the user within the page structure.

- Add table captions using [Hugo shortcodes](#) for tables.

Content best practices

This section contains suggested best practices for clear, concise, and consistent content.

Use present tense

Do	Don't
This command starts a proxy.	This command will start a proxy.

Exception: Use future or past tense if it is required to convey the correct meaning.

Use active voice

Do	Don't
You can explore the API using a browser.	The API can be explored using a browser.
The YAML file specifies the replica count.	The replica count is specified in the YAML file.

Exception: Use passive voice if active voice leads to an awkward construction.

Use simple and direct language

Use simple and direct language. Avoid using unnecessary phrases, such as saying "please."

Do	Don't
To create a ReplicaSet, ...	In order to create a ReplicaSet, ...
See the configuration file.	Please see the configuration file.
View the pods.	With this next command, we'll view the pods.

Address the reader as "you"

Do	Don't
You can create a Deployment by ...	We'll create a Deployment by ...
In the preceding output, you can see...	In the preceding output, we can see ...

Avoid Latin phrases

Prefer English terms over Latin abbreviations.

Do	Don't
For example, ...	e.g., ...
That is, ...	i.e., ...

Exception: Use "etc." for et cetera.

Patterns to avoid

Avoid using "we"

Using "we" in a sentence can be confusing, because the reader might not know whether they're part of the "we" you're describing.

Do	Don't
Version 1.4 includes ...	In version 1.4, we have added ...
Kubernetes provides a new feature for ...	We provide a new feature ...
This page teaches you how to use pods.	In this page, we are going to learn about pods.

Avoid jargon and idioms

Some readers speak English as a second language. Avoid jargon and idioms to help them understand better.

Do	Don't
Internally, ...	Under the hood, ...
Create a new cluster.	Turn up a new cluster.

Avoid statements about the future

Avoid making promises or giving hints about the future. If you need to talk about an alpha feature, put the text under a heading that identifies it as alpha information.

An exception to this rule is documentation about announced deprecations targeting removal in future versions. One example of documentation like this is the [Deprecated API migration guide](#).

Avoid statements that will soon be out of date

Avoid words like "currently" and "new." A feature that is new today might not be considered new in a few months.

Do	Don't
In version 1.4, ...	In the current version, ...
The Federation feature provides ...	The new Federation feature provides ...

Avoid words that assume a specific level of understanding

Avoid words such as "just", "simply", "easy", "easily", or "simple". These words do not add value.

Do	Don't
Include one command in ...	Include just one command in ...
Run the container ...	Simply run the container ...
You can remove ...	You can easily remove ...
These steps ...	These simple steps ...

EditorConfig file

The Kubernetes project maintains an EditorConfig file that sets common style preferences in text editors such as VS Code. You can use this file if you want to ensure that your contributions are consistent with the rest of the project. To view the file, refer to `.editorconfig` in the repository root.

What's next

- Learn about [writing a new topic](#).
- Learn about [using page templates](#).
- Learn about [custom hugo shortcodes](#).
- Learn about [creating a pull request](#).

Documentation style overview

The topics in this section provide guidance on writing style, content formatting and organization, and using Hugo customizations specific to Kubernetes documentation.

[Documentation Content Guide](#)

[Documentation Style Guide](#)

[Diagram Guide](#)

[Writing a new topic](#)

[Page content types](#)

[Content organization](#)

[Custom Hugo Shortcodes](#)

Documentation Content Guide

This page contains guidelines for Kubernetes documentation.

If you have questions about what's allowed, join the #sig-docs channel in [Kubernetes Slack](#) and ask!

You can register for Kubernetes Slack at https://slack.k8s.io/.

For information on creating new content for the Kubernetes docs, follow the [style guide](#).

Overview

Source for the Kubernetes website, including the docs, resides in the [kubernetes/website](#) repository.

Located in the `kubernetes/website/content/<language_code>/docs` folder, the majority of Kubernetes documentation is specific to the [Kubernetes project](#).

What's allowed

Kubernetes docs allow content for third-party projects only when:

- Content documents software in the Kubernetes project
- Content documents software that's out of project but necessary for Kubernetes to function
- Content is canonical on kubernetes.io, or links to canonical content elsewhere

Third party content

Kubernetes documentation includes applied examples of projects in the Kubernetes project—projects that live in the [kubernetes](#) and [kubernetes-sigs](#) GitHub organizations.

Links to active content in the Kubernetes project are always allowed.

Kubernetes requires some third party content to function. Examples include container runtimes (containerd, CRI-O, Docker), [networking policy](#) (CNI plugins), [Ingress controllers](#), and [logging](#).

Docs can link to third-party open source software (OSS) outside the Kubernetes project only if it's necessary for Kubernetes to function.

Dual sourced content

Wherever possible, Kubernetes docs link to canonical sources instead of hosting dual-sourced content.

Dual-sourced content requires double the effort (or more!) to maintain and grows stale more quickly.

Note:

If you're a maintainer for a Kubernetes project and need help hosting your own docs, ask for help in [#sig-docs on Kubernetes Slack](#).

More information

If you have questions about allowed content, join the [Kubernetes Slack](#) #sig-docs channel and ask!

What's next

- Read the [Style guide](#).

Content organization

This site uses Hugo. In Hugo, [content organization](#) is a core concept.

Note:

Hugo Tip: Start Hugo with `hugo server --navigateToChanged` for content edit-sessions.

Page Lists

Page Order

The documentation side menu, the documentation page browser etc. are listed using Hugo's default sort order, which sorts by weight (from 1), date (newest first), and finally by the link title.

Given that, if you want to move a page or a section up, set a weight in the page's front matter:

```
title: My Page
weight: 10
```

Note:

For page weights, it can be smart not to use 1, 2, 3 ..., but some other interval, say 10, 20, 30... This allows you to insert pages where you want later. Additionally, each weight within the same directory (section) should not be overlapped with the other weights. This makes sure that content is always organized correctly, especially in localized content.

Documentation Main Menu

The `Documentation` main menu is built from the sections below `docs/` with the `main_menu` flag set in front matter of the `_index.md` section content file:

```
main_menu: true
```

Note that the link title is fetched from the page's `linkTitle`, so if you want it to be something different than the title, change it in the content file:

```
main_menu: true
title: Page TitlelinkTitle: Title used in links
```

Note:

The above needs to be done per language. If you don't see your section in the menu, it is probably because it is not identified as a section by Hugo. Create a `_index.md` content file in the section folder.

Documentation Side Menu

The documentation side-bar menu is built from the *current section tree* starting below `docs/`.

It will show all sections and their pages.

If you don't want to list a section or page, set the `toc_hide` flag to `true` in front matter:

```
toc_hide: true
```

When you navigate to a section that has content, the specific section or page (e.g. `_index.md`) is shown. Else, the first page inside that section is shown.

Documentation Browser

The page browser on the documentation home page is built using all the sections and pages that are directly below the `docs section`.

If you don't want to list a section or page, set the `toc_hide` flag to `true` in front matter:

```
toc_hide: true
```

The Main Menu

The site links in the top-right menu -- and also in the footer -- are built by page-lookups. This is to make sure that the page actually exists. So, if the `case-studies` section does not exist in a site (language), it will not be linked to.

Page Bundles

In addition to standalone content pages (Markdown files), Hugo supports [Page Bundles](#).

One example is [Custom Hugo Shortcodes](#). It is considered a `leaf bundle`. Everything below the directory, including the `index.md`, will be part of the bundle. This also includes page-relative links, images that can be processed etc.:

```
en/docs/home/contribute/includes
├── example1.md
├── example2.md
├── index.md
└── podtemplate.json
```

Another widely used example is the `includes` bundle. It sets `headless: true` in front matter, which means that it does not get its own URL. It is only used in other pages.

```
en/includes
├── default-storage-class-prereqs.md
├── index.md
├── partner-script.js
├── partner-style.css
├── task-tutorial-prereqs.md
├── user-guide-content-moved.md
└── user-guide-migration-notice.md
```

Some important notes to the files in the bundles:

- For translated bundles, any missing non-content files will be inherited from languages above. This avoids duplication.
- All the files in a bundle are what Hugo calls `Resources` and you can provide metadata per language, such as parameters and title, even if it does not supports front matter (YAML files etc.). See [Page Resources Metadata](#).
- The value you get from `.RelPermalink` of a `Resource` is page-relative. See [Permalinks](#).

Styles

The [SASS](#) source of the stylesheets for this site is stored in `assets/sass` and is automatically built by Hugo.

What's next

- Learn about [custom Hugo shortcodes](#)
- Learn about the [Style guide](#)
- Learn about the [Content guide](#)

Page content types

The Kubernetes documentation follows several types of page content:

- Concept
- Task
- Tutorial
- Reference

Content sections

Each page content type contains a number of sections defined by Markdown comments and HTML headings. You can add content headings to your page with the `heading` shortcode. The comments and headings help maintain the structure of the page content types.

Examples of Markdown comments defining page content sections:

```
<!-- overview -->
```

```
<!-- body -->
```

To create common headings in your content pages, use the `heading` shortcode with a heading string.

Examples of heading strings:

- whatsnext
- prerequisites
- objectives
- cleanup
- synopsis
- seealso
- options

For example, to create a `whatsnext` heading, add the heading shortcode with the "whatsnext" string:

```
## {{% heading "whatsnext" %}}
```

You can declare a `prerequisites` heading as follows:

```
## {{% heading "prerequisites" %}}
```

The `heading` shortcode expects one string parameter. The heading string parameter matches the prefix of a variable in the `i18n/<lang>/<lang>.toml` files. For example:

```
i18n/en/en.toml:

[whatsnext_heading]
other = "What's next"
```

```
i18n/ko/ko.toml:

[whatsnext_heading]
other = "다음 내용"
```

Content types

Each content type informally defines its expected page structure. Create page content with the suggested page sections.

Concept

A concept page explains some aspect of Kubernetes. For example, a concept page might describe the Kubernetes Deployment object and explain the role it plays as an application once it is deployed, scaled, and updated. Typically, concept pages don't include sequences of steps, but instead provide links to tasks or tutorials.

To write a new concept page, create a Markdown file in a subdirectory of the `/content/en/docs/concepts` directory, with the following characteristics:

Concept pages are divided into three sections:

Page section
overview
body
whatsnext

The `overview` and `body` sections appear as comments in the concept page. You can add the `whatsnext` section to your page with the `heading` shortcode.

Fill each section with content. Follow these guidelines:

- Organize content with H2 and H3 headings.
- For `overview`, set the topic's context with a single paragraph.
- For `body`, explain the concept.
- For `whatsnext`, provide a bulleted list of topics (5 maximum) to learn more about the concept.

Annotations is a published example of a concept page.

Task

A task page shows how to do a single thing, typically by giving a short sequence of steps. Task pages have minimal explanation, but often provide links to conceptual topics that provide related background and knowledge.

To write a new task page, create a Markdown file in a subdirectory of the `/content/en/docs/tasks` directory, with the following characteristics:

Page section
overview
prerequisites
steps
discussion
whatsnext

The `overview`, `steps`, and `discussion` sections appear as comments in the task page. You can add the `prerequisites` and `whatsnext` sections to your page with the `heading` shortcode.

Within each section, write your content. Use the following guidelines:

- Use a minimum of H2 headings (with two leading # characters). The sections themselves are titled automatically by the template.
- For `overview`, use a paragraph to set context for the entire topic.
- For `prerequisites`, use bullet lists when possible. Start adding additional prerequisites below the `include`. The default prerequisites include a running Kubernetes cluster.
- For `steps`, use numbered lists.
- For discussion, use normal content to expand upon the information covered in `steps`.
- For `whatsnext`, give a bullet list of up to 5 topics the reader might be interested in reading next.

An example of a published task topic is [Using an HTTP proxy to access the Kubernetes API](#).

Tutorial

A tutorial page shows how to accomplish a goal that is larger than a single task. Typically a tutorial page has several sections, each of which has a sequence of steps. For example, a tutorial might provide a walkthrough of a code sample that illustrates a certain feature of Kubernetes. Tutorials can include surface-level explanations, but should link to related concept topics for deep explanations.

To write a new tutorial page, create a Markdown file in a subdirectory of the `/content/en/docs/tutorials` directory, with the following characteristics:

Page section
overview
prerequisites
objectives
lessoncontent
cleanup
whatsnext

The `overview`, `objectives`, and `lessoncontent` sections appear as comments in the tutorial page. You can add the `prerequisites`, `cleanup`, and `whatsnext` sections to your page with the `heading` shortcode.

Within each section, write your content. Use the following guidelines:

- Use a minimum of H2 headings (with two leading # characters). The sections themselves are titled automatically by the template.
- For `overview`, use a paragraph to set context for the entire topic.
- For `prerequisites`, use bullet lists when possible. Add additional prerequisites below the ones included by default.
- For `objectives`, use bullet lists.
- For `lessoncontent`, use a mix of numbered lists and narrative content as appropriate.
- For `cleanup`, use numbered lists to describe the steps to clean up the state of the cluster after finishing the task.
- For `whatsnext`, give a bullet list of up to 5 topics the reader might be interested in reading next.

An example of a published tutorial topic is [Running a Stateless Application Using a Deployment](#).

Reference

A component tool reference page shows the description and flag options output for a Kubernetes component tool. Each page generates from scripts using the component tool commands.

A tool reference page has several possible sections:

Page section
synopsis
options
options from parent commands
examples
seealso

Examples of published tool reference pages are:

- [kubeadm init](#)
- [kube-apiserver](#)
- [kubectl](#)

What's next

- Learn about the [Style guide](#)
- Learn about the [Content guide](#)
- Learn about [content organization](#)

Custom Hugo Shortcodes

This page explains the custom Hugo shortcodes that can be used in Kubernetes Markdown documentation.

Read more about shortcodes in the [Hugo documentation](#).

Feature state

In a Markdown page (`.md` file) on this site, you can add a shortcode to display version and state of the documented feature.

Feature state demo

Below is a demo of the feature state snippet, which displays the feature as stable in the latest Kubernetes version.

```
{{< feature-state state="stable" >}}
```

Renders to:

FEATURE STATE: `Kubernetes v1.34 [stable]`

The valid values for `state` are:

- alpha
- beta
- deprecated
- stable

Feature state code

The displayed Kubernetes version defaults to that of the page or the site. You can change the feature state version by passing the `for_k8s_version` shortcode parameter. For example:

```
{{< feature-state for_k8s_version="v1.10" state="beta" >}}
```

Renders to:

FEATURE STATE: `Kubernetes v1.10 [beta]`

Feature state retrieval from description file

To dynamically determine the state of the feature, make use of the `feature_gate_name` shortcode parameter. The feature state details will be extracted from the corresponding feature gate description file located in `content/en/docs/reference/command-line-tools-reference/feature-gates/`. For example:

```
{{< feature-state feature_gate_name="NodeSwap" >}}
```

Renders to:

FEATURE STATE: `Kubernetes v1.34 [stable]` (enabled by default: true)

Feature gate description

In a Markdown page (`.md` file) on this site, you can add a shortcode to display the description for a shortcode.

Feature gate description demo

Below is a demo of the feature state snippet, which displays the feature as stable in the latest Kubernetes version.

```
{{< feature-gate-description name="DryRun" >}}
```

Renders to:

`DryRun`: Enable server-side [dry run](#) requests so that validation, merging, and mutation can be tested without committing.

Glossary

There are two glossary shortcodes: `glossary_tooltip` and `glossary_definition`.

You can reference glossary terms with an inclusion that automatically updates and replaces content with the relevant links from [our glossary](#). When the glossary term is moused-over, the glossary entry displays a tooltip. The glossary term also displays as a link.

As well as inclusions with tooltips, you can reuse the definitions from the glossary in page content.

The raw data for glossary terms is stored at the glossary directory, with a content file for each glossary term.

Glossary demo

For example, the following include within the Markdown renders to cluster with a tooltip:

```
{{< glossary_tooltip text="cluster" term_id="cluster" >}}
```

Here's a short glossary definition:

```
{{< glossary_definition prepend="A cluster is" term_id="cluster" length="short" >}}
```

which renders as:

A cluster is a set of worker machines, called nodes, that run containerized applications. Every cluster has at least one worker node.

You can also include a full definition:

```
{{< glossary_definition term_id="cluster" length="all" >}}
```

which renders as:

A set of worker machines, called nodes, that run containerized applications. Every cluster has at least one worker node.

The worker node(s) host the Pods that are the components of the application workload. The control plane manages the worker nodes and the Pods in the cluster. In production environments, the control plane usually runs across multiple computers and a cluster usually runs multiple nodes, providing fault-tolerance and high availability.

Links to API Reference

You can link to a page of the Kubernetes API reference using the `api-reference` shortcode, for example to the Pod reference:

```
{{< api-reference page="workload-resources/pod-v1" >}}
```

The content of the `page` parameter is the suffix of the URL of the API reference page.

You can link to a specific place into a page by specifying an `anchor` parameter, for example to the PodSpec reference or the environment-variables section of the page:

```
{{< api-reference page="workload-resources/pod-v1" anchor="PodSpec" >}}
{{< api-reference page="workload-resources/pod-v1" anchor="environment-variables" >}}
```

You can change the text of the link by specifying a `text` parameter, for example by linking to the Environment Variables section of the page:

```
{{< api-reference page="workload-resources/pod-v1" anchor="environment-variables" text="Environment Variable" >}}
```

Table captions

You can make tables more accessible to screen readers by adding a table caption. To add a caption to a table, enclose the table with a `table` shortcode and specify the caption with the `caption` parameter.

Note:

Table captions are visible to screen readers but invisible when viewed in standard HTML.

Here's an example:

```
{{< table caption="Configuration parameters" >}}
Parameter | Description | Default
:---------|:------------|:-------
`timeout` | The timeout for requests | `30s`
`logLevel` | The log level for log output | `INFO`
{{< /table >}}
```

The rendered table looks like this:

Parameter	Description	Default
timeout	The timeout for requests	30s
logLevel	The log level for log output	INFO

If you inspect the HTML for the table, you should see this element immediately after the opening `<table>` element:

```
<caption style="display: none;">Configuration parameters</caption>
```

Tabs

In a markdown page (`.md` file) on this site, you can add a tab set to display multiple flavors of a given solution.

The `tabs` shortcode takes these parameters:

- `name`: The name as shown on the tab.
- `codelang`: If you provide inner content to the `tab` shortcode, you can tell Hugo what code language to use for highlighting.

- include: The file to include in the tab. If the tab lives in a Hugo [leaf bundle](), the file -- which can be any MIME type supported by Hugo -- is looked up in the bundle itself. If not, the content page that needs to be included is looked up relative to the current page. Note that with the `include`, you do not have any shortcode inner content and must use the self-closing syntax. For example, `{{< tab name="Content File #1" include="example1" />}}`. The language needs to be specified under `codelang` or the language is taken based on the file name. Non-content files are code-highlighted by default.
- If your inner content is markdown, you must use the `%`-delimiter to surround the tab. For example, `{{% tab name="Tab 1" %}}This is **markdown**{{% /tab %}}`
- You can combine the variations mentioned above inside a tab set.

Below is a demo of the tabs shortcode.

Note:

The tab **name** in a `tabs` definition must be unique within a content page.

Tabs demo: Code highlighting

```
{{< tabs name="tab_with_code" >}}
{{< tab name="Tab 1" codelang="bash" >}}
echo "This is tab 1."
{{< /tab >}}
{{< tab name="Tab 2" codelang="go" >}}
println "This is tab 2."
{{< /tab >}}
{{< /tabs >}}
```

Renders to:

- [Tab 1]()
- [Tab 2]()

```
echo "This is tab 1."
```

```
println "This is tab 2."
```

Tabs demo: Inline Markdown and HTML

```
{{< tabs name="tab_with_md" >}}
{{% tab name="Markdown" %}}
This is **some markdown.**
{{< note >}}
It can even contain shortcodes.
{{< /note >}}
{{% /tab %}}
{{< tab name="HTML" >}}
<div>
    <h3>Plain HTML</h3>
    <p>This is some <i>plain</i> HTML.</p>
</div>
{{< /tab >}}
{{< /tabs >}}
```

Renders to:

- [Markdown]()
- [HTML]()

This is **some markdown.**

Note:

It can even contain shortcodes.

Plain HTML

This is some *plain* HTML.

Tabs demo: File include

```
{{< tabs name="tab_with_file_include" >}}
{{< tab name="Content File #1" include="example1" />}}
{{< tab name="Content File #2" include="example2" />}}
{{< tab name="JSON File" include="podtemplate" />}}
{{< /tabs >}}
```

Renders to:

- [Content File #1]()
- [Content File #2]()
- [JSON File]()

This is an **example** content file inside the **includes** leaf bundle.

Note:

Included content files can also contain shortcodes.

This is another **example** content file inside the **includes** leaf bundle.

```json
{
  "apiVersion": "v1",
  "kind": "PodTemplate",
  "metadata": {
    "name": "nginx"
  },
  "template": {
    "metadata": {
      "labels": {
        "name": "nginx"
      },
      "generateName": "nginx-"
    },
    "spec": {
      "containers": [{
        "name": "nginx",
        "image": "dockerfile/nginx",
        "ports": [{"containerPort": 80}]
      }]
    }
  }
}
```

Source code files

You can use the `{{% code_sample %}}` shortcode to embed the contents of file in a code block to allow users to download or copy its content to their clipboard. This shortcode is used when the contents of the sample file is generic and reusable, and you want the users to try it out themselves.

This shortcode takes in two named parameters: `language` and `file`. The mandatory parameter `file` is used to specify the path to the file being displayed. The optional parameter `language` is used to specify the programming language of the file. If the `language` parameter is not provided, the shortcode will attempt to guess the language based on the file extension.

For example:

`{{% code_sample language="yaml" file="application/deployment-scale.yaml" %}}`

The output is:

application/deployment-scale.yaml Copy application/deployment-scale.yaml to clipboard

```yaml
apiVersion: apps/v1
kind: Deployment metadata:   name: nginx-deployment spec:   selector:     matchLabels:       app: nginx   replicas: 4 # Update the replic
```

When adding a new sample file, such as a YAML file, create the file in one of the `<LANG>/examples/` subdirectories where `<LANG>` is the language for the page. In the markdown of your page, use the `code` shortcode:

`{{% code_sample file="<RELATIVE-PATH>/example-yaml>" %}}`

where `<RELATIVE-PATH>` is the path to the sample file to include, relative to the `examples` directory. The following shortcode references a YAML file located at `/content/en/examples/configmap/configmaps.yaml`.

`{{% code_sample file="configmap/configmaps.yaml" %}}`

The legacy `{{% codenew %}}` shortcode is being replaced by `{{% code_sample %}}`. Use `{{% code_sample %}}` (not `{{% codenew %}}` or `{{% code %}}`) in new documentation.

Third party content marker

Running Kubernetes requires third-party software. For example: you usually need to add a [DNS server](#) to your cluster so that name resolution works.

When we link to third-party software, or otherwise mention it, we follow the [content guide](#) and we also mark those third party items.

Using these shortcodes adds a disclaimer to any documentation page that uses them.

Lists

For a list of several third-party items, add:

`{{% thirdparty-content %}}`

just below the heading for the section that includes all items.

Items

If you have a list where most of the items refer to in-project software (for example: Kubernetes itself, and the separate [Descheduler](#) component), then there is a different form to use.

Add the shortcode:

`{{% thirdparty-content single="true" %}}`

before the item, or just below the heading for the specific item.

Details

You can render a `<details>` HTML element using a shortcode:

```
{{< details summary="More about widgets" >}}
The frobnicator extension API implements _widgets_ using example running text.

Neque porro quisquam est, qui dolorem ipsum quia dolor sit amet, consectetur,
adipisci velit, sed quia non numquam eius modi tempora incidunt ut labore et
dolore magnam aliquam quaerat voluptatem.
{{< /details >}}
```

This renders as:

▶ More about widgets

Note:

Use this shortcode sparingly; it is usually best to have all of the text directly shown to readers.

Version strings

To generate a version string for inclusion in the documentation, you can choose from several version shortcodes. Each version shortcode displays a version string derived from the value of a version parameter found in the site configuration file, `hugo.toml`. The two most commonly used version parameters are `latest` and `version`.

{{< param "version" >}}

The `{{< param "version" >}}` shortcode generates the value of the current version of the Kubernetes documentation from the `version` site parameter. The `param` shortcode accepts the name of one site parameter, in this case: `version`.

Note:

In previously released documentation, `latest` and `version` parameter values are not equivalent. After a new version is released, `latest` is incremented and the value of `version` for the documentation set remains unchanged. For example, a previously released version of the documentation displays `version` as `v1.19` and `latest` as `v1.20`.

Renders to:

v1.34

{{< latest-version >}}

The `{{< latest-version >}}` shortcode returns the value of the `latest` site parameter. The `latest` site parameter is updated when a new version of the documentation is released. This parameter does not always match the value of `version` in a documentation set.

Renders to:

v1.34

{{< latest-semver >}}

The `{{< latest-semver >}}` shortcode generates the value of `latest` without the "v" prefix.

Renders to:

1.34

{{< version-check >}}

The `{{< version-check >}}` shortcode checks if the `min-kubernetes-server-version` page parameter is present and then uses this value to compare to `version`.

Renders to:

To check the version, enter `kubectl version`.

{{< latest-release-notes >}}

The `{{< latest-release-notes >}}` shortcode generates a version string from `latest` and removes the "v" prefix. The shortcode prints a new URL for the release note CHANGELOG page with the modified version string.

Renders to:

https://git.k8s.io/kubernetes/CHANGELOG/CHANGELOG-1.34.md

What's next

- Learn about [Hugo](#).
- Learn about [writing a new topic](#).
- Learn about [page content types](#).
- Learn about [opening a pull request](#).
- Learn about [advanced contributing](#).