Documentation Style Guide

- Write Clearly
  - Voice
  - Numbers
  - Words and Phrases to Avoid
  - Clarity Rules
- Terminology
- Page Types
  - Reference Pages
  - Guide Pages
- Titles and Headings
- Formatting
  - Frontmatter
  - Code Blocks and Tabs
  - Option Tables
  - Callouts
  - Links
  - Screenshots and Media
  - Diagrams
- Generated Files
- Before You Open a Pull Request

This guide covers how to write and edit pages under `docs/content/`. It is written for contributors and for the AI assistants they use. Follow it for new pages and for any lines you change. Existing pages may not follow every rule. Don't copy a pattern from a neighboring page when it contradicts this guide, and don't fix unrelated lines in the same pull request.

When following a rule would make the text less accurate or less clear, accuracy wins. Say why in the pull request.

Write Clearly

Voice

- Use active voice and simple present tense.
- Use the imperative in procedures: "Set `address` to `:80`", not "You should set `address` to `:80`".
- Address the reader as "you". Guides can use "we" when walking through a procedure together ("we'll create a cluster").
- Make the reader the subject, not the page or the product. Describe what the reader can do or must do, and don't describe what the page does.
  - Avoid: "This page allows you to attach a middleware to a router."
  - Prefer: "You can attach a middleware to a router to change requests before they reach a service."
- Omit "please". Replace "should" with a direct statement.
- Write "click **Save**", not "click on **Save**".
- Use simple words. Use technical terms only when they are standard for the technology (`TLS`, `mTLS`, `CIDR`) or appear in Traefik's configuration or reference docs. Explain any other shorthand on first use.
- Don't give human characteristics to software ("Traefik wants", "the router decides").
- Don't include customer names or customer-specific details. Generalize the use case or leave it out.
- Don't state how a third-party service behaves (what it accepts, rejects, or requires) unless that service's documentation or the source says so. The source shows what Traefik sends, not how an external service responds. Link to the third party's documentation instead.
- When the source supports a vendor-specific behavior, write the general condition first and the vendor as the example: "If your identity provider returns no groups, as ExampleIdP does, ...".

Numbers

- Spell out one through ten in prose. Use numerals for 11 and above.
- Always use numerals for values, versions, ports, and numbers with units: `maxBodySize: 5`, HTTP/2, port 80, 5 MB.
- Don't state how many items a list or table on the page contains ("the following three options"). The count goes stale when the list changes. Limits the product defines ("up to 100 routers") are fine.

Words and Phrases to Avoid

Swap for a plain word:

Avoid	Write
leverage, utilize, harness	use
facilitate	help, let, enable
In order to	To
Furthermore, Moreover, Additionally	Also
Could potentially	can, might
Prior to this release	Before v3.4

Delete, and keep the rest of the sentence. The sentence reads correctly without them: cutting-edge, innovative, In conclusion, To summarize, As mentioned above.

Replace with the specific fact. These words describe nothing, so rewrite the sentence with the concrete detail:

- Avoid: "A robust, seamless integration with Consul." Prefer: "Traefik reads services from Consul Catalog and updates routes when services change."
- Avoid: "Powerful middleware lets you streamline authentication." Prefer: "The `forwardAuth` middleware sends each request to your authentication service before it reaches a backend."
- Avoid: "This empowers you to control traffic." Prefer: "You can limit requests per client with the `rateLimit` middleware."

This applies to: streamline, robust, comprehensive, powerful, seamless, seamlessly, empower, delve into.

Also avoid these constructions:

- **Em dashes as connectors.** Split into two sentences, or join with "so", "and", or "but". Don't swap the dash for a semicolon or colon.
- **Rhetorical questions** ("But what if you need X?"). Write a statement.
- **Dramatic framing** ("This changes everything"). State the fact.
- **Contrast framing** ("It's not just a proxy, it's a platform"). State the positive claim. A negation is fine when it names a wrong value the reader might choose: "Set the timeout in seconds, not milliseconds."
- **Absolutes** ("never X, even when Y"). State the rule and keep the condition the source gives. This doesn't apply to security warnings and `!!! danger` content, which keep their explicit form: "Never expose the dashboard without authentication."

Clarity Rules

- **Name the concrete noun.** Replace "this", "it", "the same", "both", and "these" with the thing they point to when the noun isn't in the same sentence. Every table row must make sense on its own.
- **Name the exact option when you mean the option.** Write `entryPoints` when you refer to the option, and "entry point" when you explain the concept. Write "identifier", not "ID", in prose. Use `id` only for a literal field, and keep standard or UI names as they are ("Client ID", "Trace ID").
- **Give a concrete recommendation.** Replace "use a smaller timeout" with "Set `idleTimeout` to a lower value, for example `3s`." Use only options and values the source or reference docs show.
- **Make every sentence in a procedure change what the reader does.** Each sentence gives an action, a value to use, or the result of an action. This doesn't apply to option descriptions, warnings, concept and reference pages, introductions, and callouts.
- **Don't claim a capability without the way to use it.** "You can verify each entry" needs the steps or a link in the same section.
- **Document each case separately** when behavior differs by state, or by the value of another option. List the value for each sub-option by name.
- **Document a trade-off option by its trade-off.** Say what it does, when to use it, and the risk of leaving it on.
- **Put a caveat directly after the option, step, or example it qualifies.** Put a prerequisite before the action that needs it.
- **Lead exceptions with the action.** Write "**Set `x` for `y`.** Without it, ..." and then the explanation.
- **Don't refer to position.** Avoid "above", "below", and "as shown earlier". Name the section or option instead. Readers often land mid-page from a search result.
- **Don't narrate code.** Explain why a step matters, not what a readable block of code does.
- **Use a worked example as an example.** Say "using a payments API as the example", so readers don't take the scenario as the limit of the feature.
- **Keep setup details out of a concept introduction.** Option names and values belong in Prerequisites or the reference page.
- **State version-specific behavior only for versions you verified.** Omit "Starting in vX" when the whole page requires vX. A single option added in a later version keeps its note. This doesn't apply to `migrate` and `deprecation` pages or to security advisories, where the version is the content.

Terminology

- Write **install configuration** and **routing configuration**. Don't write "static configuration" or "dynamic configuration" in new text. Older pages use both pairs of terms, so leave a term unchanged when it is part of a quoted option name or a URL.
- Write **Traefik Proxy** on first mention on a page and **Traefik** afterward.

- Use one name per concept across pages. Search neighboring pages and the pages you link to for an existing term before you introduce a name, and define it on first use on each page.
- Use each provider's own name (Docker, Consul Catalog, Kubernetes Gateway API, IngressRoute). Don't abbreviate them in prose.

Page Types

Put new content in the existing sections. Don't add a top-level section.

Section	Contains
`getting-started`, `setup`	First-run and installation walkthroughs
`expose`	How-to guides for exposing services on a platform
`secure`, `observe`	Task guides and overviews for a feature area
`reference`	One page per option group, middleware, router, service, or provider
`migrate`, `deprecation`	Version migration and deprecation notices
`security`	Security advisories

Reference Pages

Reference pages have no body H1. They start with the description sentence. Use these sections, in this order, when they apply:

1. **Configuration Examples.** One example per supported format.
2. **Configuration Options.** A table of options.

Guide Pages

Guides start with a body H1 that names the task. Use this order:

1. A short introduction: what the reader accomplishes, and when to do it.
2. **Prerequisites.** For a prerequisite whose absence causes a visible failure, state what the reader sees.
3. One H2 per step or task, named with a verb ("Create the Cluster").
4. A closing paragraph that states the outcome, then links to related pages. Don't restate the steps.

State the expected result at the end of each major step. When a step is backed by an example with its own identifier (a Kubernetes `metadata.name`), that identifier carries the step's number, and so does every later reference to that step.

Titles and Headings

- **Use Title Case for every heading and every page title,** including a heading you add to a page whose other headings use another case. Capitalize the first and last word and every major word. Lowercase articles, conjunctions, and short prepositions (a, an, the, and, but, or, for, in, of, on, to, with) unless they are the first or last word.
- Keep code, option names, acronyms, and error messages exactly as written: `entryPoints`, HTTP, `connection refused`.
- Name a task with a verb: "Configure Sticky Sessions". Name a reference section after its topic using the product's own term: "Certificate Resolver".
- Don't phrase a heading as a question, or as the reader's discovery ("Find Out What Went Wrong"). Use "Troubleshoot Failed Requests".
- Title a troubleshooting entry with the symptom or the exact error message.
- Reference page titles in frontmatter follow the existing pattern: `Traefik <Name> Documentation`.
- Make each section understandable without the sections around it.

Formatting

Frontmatter

Use `title` and `description` only. Write the description as one sentence that says what the page covers, for search results.

Code Blocks and Tabs

- Show every format the feature supports for a configuration example, and make sure each format produces the same configuration.
- Label tabs with these names, in this order:
  - **Install configuration:** `File (YAML)`, `File (TOML)`, `CLI`, `Helm Chart Values`.
  - **Routing configuration:** `Structured (YAML)`, `Structured (TOML)`, `Labels`, `Tags`, `Kubernetes`. Include only the formats the provider supports.
- Use the tab syntax in the build guide: a `tab="Label"` attribute on the code fence.

- Put explanation in comments inside the example, once per example, in the formats that support comments. Don't narrate the example in the paragraph around it.
- Use `yaml`, `toml`, `bash`, or `text` as the fence language. Don't leave a fence without a language.

Option Tables

- Use the columns **Field**, **Description**, **Default**, and **Required**. Use **Value** instead of **Default** and **Required** for a table that lists fixed values.
- Link each field name to its own anchor so readers can link to a single option.
- Describe each option with what it does, its expected format, and an example value.
- State what happens when the option is unset. Write "Traefik doesn't compress responses", not "the default configuration".
- When a field is conditional, state when it is required or applies.
- Sort a long table alphabetically. Don't use table headers that address the reader ("Tells you"). Use neutral labels ("Contains", "Effect").
- When you extend a table, write out every row. Never abbreviate existing rows with "..." or "etc."
- Don't follow a table with a list that groups the same rows differently. Add a column instead.
- Avoid tables that need horizontal scrolling. Split them or use a list.

Callouts

- Prose is the default. Add a callout only when the information would be lost in the body text.
- Use `!!! note` for extra details or alternatives, `!!! warning` for restrictions that cause errors if missed, `!!! tip` for shortcuts, and `!!! info` for background. Give each callout a title when the type alone doesn't say enough.
- If a callout explains why an instruction matters, merge it into the instruction.
- Never stack two callouts. Put body text between them, or fold one into prose.
- `??? example` is a collapsible container for a long, optional example. It is not an attention callout, so the limits above don't apply to it.

Links

- Name the destination in the link text. Never write "here" or "this link".
- Use relative paths to the `.md` file: `[HTTP Routers](../routing/router.md)`.
- Link to a page only when it exists on the branch you target. If it doesn't exist yet, use plain text and note the missing link in your pull request.
- After you move or rename a page, add a redirect to `mkdocs.yml`.

Screenshots and Media

- Use text to instruct. A screenshot supports the text and never replaces it.
- Write alt text and a supporting sentence for each screenshot.
- Use as few screenshots as you can.
- Name the control ("click **Settings**"). Don't describe its position ("the button on the left").

Diagrams

Avoid adding diagrams. Add one only when a new feature is documented noticeably better with it. In that case:

- Save the image in `docs/content/assets/img/` as PNG, or as SVG if your tool exports it cleanly. Say in the pull request where the editable source file is, so maintainers can update it.
- Reference it with a relative path and Markdown image syntax: `![Request Flow Through Entry Points, Routers, and Services](../assets/img/request-flow.png)`.
- Write alt text that states what the diagram shows. Don't write "diagram of".
- Introduce the diagram with one sentence that uses the same labels as the diagram.
- Put every fact the diagram carries in the text as well, because a reader using a screen reader or search can't read an image.
- Don't redraw a flow that another page already diagrams. Link to that page, or verify every arrow against the source.

Generated Files

Don't edit these files by hand. A build step overwrites them:

- `reference/install-configuration/configuration-options.md`
- `reference/dynamic-configuration/traefik.io_*.yaml` and `kubernetes-crd-definition-v1.yml`

To change an option description in the generated reference, change the comment on the Go struct field and run `make generate`. The CI check fails if generation leaves uncommitted changes.

Before You Open a Pull Request

1. Verify every claim against the source code or the third party's documentation. Don't add an error message, default, or value you haven't confirmed, and don't publish behavior you haven't confirmed.
2. Check that every link and anchor resolves.

3. Run the docs build with `mkdocs build --strict` and the markdown lint, as described in the contributing guide.
4. Target the right branch: `master` for new features, and the maintenance branch for the release for fixes. Don't backport your own change.
5. Declare substantial AI assistance with an `Assisted-by:` trailer.

□□□ #2
□ □□□ □□□ 2026-10-31 13:57:54 UTC
□ □□□ □□□ 2026-10-31 13:57:55 UTC