How I write technical posts: a field guide with examples

Muhammad Gulzaib
https://orcid.org/0009-0000-8709-6989
SynergyFlow Labs · Islamabad, Pakistan

Published	2026-11-22
URL	https://gulzaib.vercel.app/blog/how-i-write-technical-posts-a-field-guide-with-examples
Licence	Creative Commons Attribution 4.0 International (CC BY 4.0), https://creativecommons.org/licenses/by/4.0/

ABSTRACT

A practical process for clear technical articles, from finding the one idea to editing for skimmers, plus a live reference of every block this blog supports.

Keywords: writing, documentation, style-guide, mdx

> **TL;DR**
>
> Write for one reader with one problem. Lead with the answer, then the reasoning, then the details. Edit for the skimmer first: headings, summaries and visuals should tell the story on their own.

I was the technical editor of my university's magazine, and I've written a lot of documentation since. The posts that work best all follow the same process. This post explains it, and it doubles as the reference for every component this blog supports, so you can see each one in context.

1. Find the one idea

Before writing, finish this sentence: *"After reading this, the reader will be able to …"*. If you need two sentences, you have two posts.

It also helps to know which kind of document you're writing. The Diátaxis framework [1] splits technical writing into four kinds:

Type	Reader's question	Example on this blog
Tutorial	"Teach me."	A first Laravel API
How-to guide	"Help me do this task."	Multi-tenant Laravel
Explanation	"Help me understand."	RAG for exam generation

Reference	"Give me the facts."	This component list

2. Lead with the answer

Readers decide in seconds whether to keep reading. Put the conclusion first, in a summary box:

> **TL;DR**
>
> This is a `TLDR` box. Use it at the top of a post for the answer in two or three sentences.

Research posts get an `Abstract` instead, which reads more formally:

Abstract

This is an `Abstract`. Problem, approach, result, in one paragraph.

3. Structure for skimmers

Most people skim first and read second. Good headings, short paragraphs and lists let them find what they need. Every heading on this blog gets an anchor link (hover it and click the #), and the table of contents follows you as you read.

Callouts

Use callouts sparingly, for things the reader must not miss.

> **Note**
>
> A `note` adds useful context.

> **Tip**
>
> A `tip` is a shortcut or best practice.

> **Warning**
>
> A `warning` flags something that can go wrong.

> **Careful**
>
> A `danger` callout is for data loss, security or money.

> **Insight**
>
> An `insight` is the non-obvious idea in the post.

> **Custom titles work too**
>
> A `success` callout reports a result.

Steps

1. **Use Steps for procedures**

   When order matters, number it. Each step gets a title and a body.

2. **One action per step**

   If a step has "and then" in it, split it.

Details

Use Details to hide optional depth

Edge cases, long logs and full config files go here, so the main path stays short.

4. Show, don't only tell

Code

Code blocks have syntax highlighting, a language label, a copy button, optional file titles, line numbers and highlighted lines:

`src/example.ts`

```
export function greet(name: string) {
  return `Hello, ${name}!` // this line is highlighted
}
```

Inline code looks like `npm run build`, and keyboard shortcuts look like Ctrl + K. On the blog index, press / to search.

npm

```
npm install @mdx-js/rollup
```

pnpm

```
pnpm add @mdx-js/rollup
```

- src/blog/
  - **posts/** one `.mdx` file per post
  - components/
  - blog.css

Diagrams and charts

1. **Idea**
2. **Outline**
3. **Draft**
4. **Edit**
5. **Publish**

A Flow shows a sequence or pipeline. It wraps on small screens.

BarChart: simple comparisons without a charting library

Skim headings80%

Read summary60%

Read everything20%

Illustrative numbers. Captions should say what the reader is looking at and where the data came from.

9Postsand counting

4Categories

68chLine lengthfor comfortable reading

1. Step 1

   A Timeline

   Good for stories, launches and post-mortems.
2. Step 2

   Each event

   Has a date, a title and an optional body.

Math

Inline math like `E = mc^2` uses single dollar signs. Display math uses double:

```
\operatorname{softmax}(z)_i = \frac{e^{z_i}}{\sum_j e^{z_j}}
```

Figures and video

A Figure with a caption. Use src and alt for images; always describe the image in alt text. Cover art is generated from the post's slug.

A `YouTube` component loads a privacy-friendly embed only when clicked, so it never slows the page down.

5. Help the reader decide

Founders and engineers read to make decisions. Make the trade-offs explicit.

ProsCons

Pros

- Makes trade-offs scannable
- Forces you to be honest about downsides

Cons

- Oversimplifies nuanced choices

Decision · Nov 2026 · Proposed

A Decision record (ADR)

Context. Why a decision was needed.

Decision. What we chose.

Consequences. What gets easier and what gets harder.

6. Write like a person

Plain words beat clever ones. Zinsser's advice still holds: strip every sentence to its cleanest components [2]. Use "I" and "you". Explain jargon the first time, like retrieval quality, with a `Term` that shows a definition on hover or tap.

Side comments that would interrupt the flow go in a sidenote. (On wide screens this sits in the margin. On phones, tap the mark to open it inline.) Classic footnotes work too.1

> Use a pull quote for the one line you want people to remember and share.

— A PullQuote

> A regular blockquote is for quoting someone else's words.

Link to related posts with a card:

*[Figure: image]***Code is not a business: what 20 paying customers taught me**: I built a POS SaaS, sold it to about 20 shops, then saw it solved the wrong problem. How I validate a SaaS idea now, and the checklist I use before coding.

Or to anything on the web:

Diátaxis documentation framework: A systematic approach to technical documentation authoring.

7. Edit, then edit for the skimmer

- ☐ Read only the headings. Do they tell the story?
- ☐ Read only the summary and takeaways. Is the answer there?
- ☐ Cut the first paragraph. Was it throat-clearing?
- ☐ Read it aloud. Fix every sentence you stumble on.

> **Key takeaways**
>
> - One reader, one problem, one idea per post.
> - Lead with the answer.
> - Structure for skimmers, then write for readers.
> - Make trade-offs explicit and cite your sources.

> Update · 22 Nov 2026
>
> Use an `Update` block when you change a published post, so returning readers know what's new.

Footnotes

1. Footnotes are collected at the end of the post, with a link back to where you were. ↩

References

1. Procida, D. Diátaxis: a systematic approach to technical documentation.
2. Zinsser, W. *On Writing Well*. Harper Perennial, 2006.

CITE AS

Gulzaib, M. (2026, November 22). How I write technical posts: a field guide with examples. Gul's Notebook. https://gulzaib.vercel.app/blog/how-i-write-technical-posts-a-field-guide-with-examples

You may share and adapt this text for any purpose, including commercially, as long as you credit the author and link to the original. Licence: https://creativecommons.org/licenses/by/4.0/