

Table of Contents

1. Numbered steps	2
1.1 Enable the extension	2
1.2 Write a procedure	2
1.2.1 Opening and closing the blocks	3
1.2.2 Content inside a step	3
1.3 Configure numbered steps	3
1.3.1 Continue numbering after a break	3
1.3.2 Add an id or other HTML attributes	4
1.4 Reference	5
1.5 Customise the appearance	5
1.5.1 Generated HTML	5

1. Numbered steps

`prodockit.steps` presents a procedure as numbered steps that a reader works through in order. Each step has room for a title, instructions, commands, and supporting content. A joining line makes the sequence clear on both the website and in the PDF.

Use an ordinary numbered list for a list of facts. Use `prodockit.steps` when the reader needs to complete one action before moving to the next.

1.1 Enable the extension

Add the extension to `zensical.toml`:

```
[project.markdown_extensions."prodockit.steps"]
```

1.2 Write a procedure

Copy this complete example:

```
/// steps

//// step | Load the key into the agent

```bash
ssh-add --apple-use-keychain ~/.ssh/id_ed25519_gitlab
```

////

//// step | Upload the public key

Paste it into your host's SSH keys page.

////

///
```

It renders as:

1. Load the key into the agent

```
ssh-add --apple-use-keychain ~/.ssh/id_ed25519_gitlab
```

2. Upload the public key

Paste it into your host's SSH keys page.

1.2.1 Opening and closing the blocks

This extension uses the fenced-container syntax from [PyMdown Blocks](#). An opening fence names the block; its closing fence uses the same number of slashes without the name. Nested blocks must use a different fence length. In a numbered procedure:

- `/// steps` opens the complete ordered procedure; `///` closes it.
- `//// step` opens one step; `////` closes it.
- Add a title after `|`, as in `//// step | Upload the public key`.
- Leave off `|` and the title when a step needs only body content.

The blank line after a step header is not required for a simple paragraph, but some block content needs it. Keeping the blank lines shown in the example is the safe, consistent form for paragraphs, fenced code, admonitions, and content tabs.

Fence lengths must differ when blocks are nested

The `step` fence uses four slashes because it is nested inside the three-slash `steps` fence. If a step contains another Blocks-based container, give that nested block another unused fence length.

Showing Markdown that contains a code fence

The copyable example uses four backticks around the complete Markdown sample because the sample itself contains a three-backtick `bash` fence. The outer fence must be longer than a fence inside it.

1.2.2 Content inside a step

A step body is ordinary block Markdown. It can contain multiple paragraphs, code fences, admonitions, or content tabs. For example, the [Build your first site](#) walkthrough places macOS, Windows, and Linux commands in tabs inside its first two steps.

1.3 Configure numbered steps

There are no additional `prodockit.steps` settings in `zensical.toml`. Configure each procedure inside its `/// steps` Markdown block.

1.3.1 Continue numbering after a break

A long procedure can be split across sections or pages. Set `start` on the later block so its first step continues at the required number:

Markdown

```
/// steps
  start: 9

//// step | Point the clone at your own project

Update the remote URL before pushing.

////

///
```

Result

1. Point the clone at your own project
- Update the remote URL before pushing.

Options use YAML syntax. Put them directly below `/// steps`, with no blank line between the header and its options, and indent each option by at least four spaces. The blank line after the final option separates the options from the procedure's content.

1.3.2 Add an id or other HTML attributes

Use the Blocks API's `attrs` option to add attributes to the generated ordered list. This complete example combines an id with continued numbering:

Markdown

```
/// steps
  start: 9
  attrs: {id: 'setup-continued'}

//// step | Verify the remote

Run git remote -v before pushing.

////
```

```
///
```

Result

- 1. Verify the remote
- Run `git remote -v` before pushing.

`attrs` is inherited from the underlying Blocks API rather than defined by `prodockit.steps` ; `start` is the extension's only steps-specific option.

1.4 Reference

The block markers and options accepted by a steps block are summarised in [Table 26.1](#).

Table 1.1. Reference

| Syntax | Purpose |
|---------------------------------|--|
| <code>/// steps</code> | Open or close the ordered procedure |
| <code>//// step</code> | Open or close one step without a title |
| <code>//// step \ Title</code> | Open one step with a title |
| <code>start : 9</code> | Start this procedure at step 9 |
| <code>attrs: {...}</code> | Add HTML attributes to the generated <code></code> |

Each title becomes a separate paragraph before the step body. A step may have no title, but every `step` must be inside a `steps` block for the resulting HTML to be a valid ordered list.

1.5 Customise the appearance

Prodockit's managed `docs/stylesheets/pdk.css` contains the numbered-step defaults used by the examples on this page. Put project-specific changes in `docs/stylesheets/extra.css` , which is loaded afterwards.

1.5.1 Generated HTML

The first example produces this structure:

```
<ol class="prodockit-steps">
  <li>
    <p class="prodockit-step-title">Load the key into the agent</p>
```

```
<pre><code>ssh-add --apple-use-keychain ~/.ssh/id_ed25519_gitlab</code></pre>
</li>
<li>
  <p class="prodockit-step-title">Upload the public key</p>
  <p>Paste it into your host's SSH keys page.</p>
</li>
</ol>
```

The two stable class names are:

[Table 26.2](#) identifies the stable HTML classes emitted for a procedure and its individual steps.

Table 1.2. Generated HTML

Selector	Element
<code>ol.prodockit-steps</code>	The complete procedure
<code>.prodockit-step-title</code>	A step's optional title paragraph

The title is an element rather than bold text, so it can be styled separately from emphasis used in the step body.

If you override the rules from `pdk.css` in `extra.css`, retain these details:

- the joining line is positioned from the number's own size, so changing `--step-size` does not move the line away from the number;
- the line's `::after` pseudo-element uses the same `font-size` as the number's `::before` pseudo-element, keeping `em`-based measurements aligned;
- the final step has no trailing joining line.

The [bootstrap quick start](#) and [Build your first site](#) are larger working examples.

Contributors changing the generated representation should read [Extension integration](#).