Table of Contents

1. Release notes

Entries are arranged newest first. Read the Unreleased section when it is present, then every version between the one a project uses and the one it will install. A change that needs author or maintainer action is described with the release that introduces it. Packaged artefacts and tags are available from GitHub Releases.

The oldest entries record the package's earlier name, `zendoc`. Versioning restarted at `prodockit` 0.1.0 after that rename, so the historical sequence near the bottom appears to move from prodockit 0.1.1 back to zendoc 0.10.0. Those are two package eras rather than duplicate releases.

1.1 0.41.0 (2026-08-20)

- **Changed:** back-of-book index generation is now configured with the extension it belongs to:

  ```
  [project.markdown_extensions."prodockit.index"]
  include = true
  title = "Index"
  ```

  This is a breaking configuration change. Replace the former `project.extra.pdf_include_index` and `project.extra.pdf_index_title` settings; they are no longer read. The extension now declares both options itself, so Zensical, the Markdown extension, and the PDF builder use one public configuration surface.

- **Fixed:** a nested `index.md`, such as the new About overview, is no longer treated as a second PDF cover. Only the first navigation page can be the compiled document's cover; later directory indexes retain their chapter number and PDF bookmark.

- **Fixed:** `prodockit bootstrap --help` now says that `.pdk-bootstrap.toml` in the current directory is the default configuration file. It previously described a user configuration directory even though the command did not use that as its normal default. Existing legacy user configuration remains readable.

- **Changed:** fenced and inline code in the PDF render at 11pt, one point below the 12pt body text. This compensates for the monospace face appearing optically larger at the same nominal size.

- **Fixed:** short content-tab panels can opt into page-safe PDF rendering by wrapping the tab group in `.pdf-keep-tab-pages`. Each reconstructed tab header now stays with its body, avoiding an empty panel fragment at the foot

of one page with all of its content on the next. Ordinary tabs remain splittable because a panel can legitimately be taller than a page.

- **Documentation:** reorganised the site's information architecture around five audiences and tasks: getting started, authoring, publishing a document, maintaining prodockit, and contributor internals. New overview pages route each reader before the detailed reference, while About now centralises maturity, platform coverage, dependencies, limitations, release history, and licence information.

- **Documentation:** rewrote all nine authoring references for a reader with limited Markdown experience. Each guide now follows the same enable, write, and configure progression and provides a rendered result for every syntax section. The numbered-steps and directory-tree guides also make their direct foundation on PyMdown Blocks explicit.

- **Documentation:** added a document-author publishing path from `prodockit-template` and machine setup through local PDF and strict website builds, built-output tests, GitHub or GitLab Pages deployment, and public verification. The guide distinguishes template-owned publishing files from an author's content and explains the Surrey `is_surrey` presentation choice.

- **Documentation:** reorganised project maintenance into a task-based path from a safe report-only check through reviewed updates, local builds, pull request gates, package publishing, and live-site verification. The release runbook follows this repository's five GitHub Actions workflows and explains where each stage resumes after a failure.

- **Documentation:** expanded `README.md` and `CONTRIBUTING.md` into current entry points for package users and contributors. They now distinguish Python packages from external PDF requirements, record the tested Ubuntu, Windows, and macOS workflows, inventory the public commands, and give the complete local verification order.

1.2 0.40.0 (2026-08-19)

- **Added:** `prodockit template-sync --push` finishes a sync where the pipeline can see it (#502).

  `--apply` stops at staged, on its own branch, and that publishes nothing: both hosts build only from the default branch, so a sync sitting on a `template-update-...` branch produces no pipeline and no rebuilt site even after you commit and push it. `--push` commits, merges into the branch your host builds from, and pushes - showing what it will do and waiting for a yes first.

  It assumes you merge your own work, with no merge request in the way. A project that gates its default branch should use `--apply` alone and raise a merge request from the update branch.

- **Fixed:** a stale checkout beside a project no longer stops `template-sync` (#500).

  A checkout beside the project wins over fetching, which is right for somebody editing the template and their project together - but it won unconditionally, so an old clone taken before the template carried a manifest stopped the command outright, with a usable copy one fetch away. A sibling now has to carry a manifest to be preferred, and the run says which checkout it passed over.

- **Fixed:** the installation page lists every extension (#504).

  It explained how to enable an extension and then named four of the nine. `prodockit.tables`, `.tree`, `.steps`, `.bibliography` and `.index` had each arrived with an entry point and a documentation page without reaching the one page a new reader goes to first. A test now treats the entry points as the authority, so a registered extension cannot go undocumented again.

- **Changed:** the `template-sync` manual covers a finished sync, not just `--apply` (#503).

  What to do with a `.new` sidecar and how to tell "you edited this" from "you never received this update"; that `--force` takes exact paths one flag at a time; that a second run branches again; and that committing alone changes nothing on the host.

- **Added:** `\ref{id}` resolves a captioned figure or table, not only a heading (#506).

  A reference to a figure renders its label - **Figure 3.1** - linked to the figure, and updates itself when figures are added or moved. Until now the number had to be typed by hand beside a `\ref` that could not resolve it, so a document said "Figure 3.1" in prose while the figure itself was numbered by the stylesheet, with nothing keeping the two in step.

  Figures and tables count separately, and both take the page's chapter number - the same numbering the caption shows, so a reference and its target always agree. Numbered in the same pass that numbers headings, which is where the chapter number already exists.

  A caption reference is its label alone, where a heading reference keeps its name: "the components in Figure 3.1" reads badly as "Figure 3.1 Component Model", while a bare "1.1" says nothing about where a reader is being sent.

1.3 0.39.0 (2026-08-19)

- **Added:** `prodockit template-sync`, which brings a project back into step with the template it came from (#495, #498).

  A project generated from a template is a copy, not a link. It starts ageing

immediately - the template gains a CI fix, a stylesheet rule, a newer pin - and nothing says so, because nothing breaks. The site still builds and the document simply looks slightly unlike everyone else's.

```
prodockit template-sync           # report; writes no project
file
prodockit template-sync --apply   # branch, write, stage, do not
commit
```

What is written is decided by a manifest in the template, not by the command. The report, its figures and its bibliography are never written and never even read, so a sync cannot lose your writing. A template-owned file you have edited is kept, with the template's version written beside it as `.new` to compare; `--force` takes the template's copy for a named file.

The template is fetched into a per-user cache, so no checkout of it is needed. A project on Surrey's GitLab tracks the Surrey mirror and everything else the GitHub copy, unless `--github`, `--surrey` or `--template-path` says otherwise. A host that cannot be reached is a third answer rather than a failure - the run continues on the cached copy and says that it may be behind.

Built for repeated use through a project rather than once at the start. A run with nothing to do says so and creates no branch; a `.new` sidecar already holding the template's bytes is not rewritten; and the recorded baseline moves forward even when a template release changes only how files are classified.

Every run appends its full account - always the `--verbose` form, and including runs that failed - to `.prodockit-template.log`, which the command adds to `.gitignore` itself.

1.4 0.38.0 (2026-08-18)

- **Added:** multi-row table headers, merged cells and rotated headings (#474).

  A Markdown table has one header row and no syntax for a second, so a heading needing two lines is written as a body row - and then stops repeating when the table breaks across pages, because only `<thead>` repeats. `{: .header }` moves the row where it belongs:

  ```
  | Target {: rowspan=2 } | Measured {: colspan=2 } | | Note {:
  rowspan=2 } |
  |---|---|---|---|
  | | Before {: .header } | After | |
  ```

  Both lines repeat on every page now, verified on a five-page table.

WeasyPrint always could repeat a multi-row header, spans and all; nothing had ever put the second row in the header for it to repeat.

`colspan` / `rowspan` are `attr_list`'s own attributes - what is new is that the empty placeholder cells they need are removed, so the row is not left wider than the header. One with text in it is kept.

Headings turn with `{: rotate=270 width="1.8em" height="105pt" }`, at 90 or 270 and nothing else. `rotate` without `width` is refused: `transform` does not affect layout, so the width is what buys the space and the rotation is what keeps the heading readable once the column is narrow. Rotating alone renders a heading in a full-width column and looks like it worked.

`transform` in both outputs rather than `writing-mode`, which WeasyPrint ignores silently - measured, the text stays horizontal while the column still narrows.

- **Added:** `{: .compact }` for a table with many short columns (#489).

  A wide table is held wider still by the theme itself: every header cell carries `min-width: 5rem`, and every cell 1.25em of padding either side. A column holding `H` is then as wide as one holding a sentence, so the table overflows whatever it contains. Measured on a real 14-column table against 1009px of A4 landscape:

  ```
  as written                    1586.7px   (57% over)
  minimum dropped               1190.7px
  and the padding tightened      993.1px   (fits)
  ```

  Neither is enough alone, so one marker turns off both:

  ```
  | Threat {: .compact } | Likelihood | Impact | Risk |
  ```

  It applies to the PDF as well, where there is no minimum and only the padding is at stake - a marker that changed one output and not the other would be the silent half-failure this project keeps meeting.

  Written on a header cell because that is the only thing `attr_list` can reach in a Markdown table, then moved onto the table and removed from the cell, so it styles the table rather than that one column. Any header cell will do, and it combines with `width`.

  Opt-in on purpose: a table that reads well at its default keeps it, and a table changing shape because a column was added is the kind of surprise worth avoiding.

- **Fixed:** setting a column width no longer changes how the whole table looks (#490).

Two tables on a page looked like different components, and the only difference in the markup was that one asked for a column width:

```
| **RAID** {: width="10%" } | **Description** | **Action** |
```

`prodockit.tables` adds `prodockit-table-sized` to a table with a width, and Zensical scopes its entire table style to `table:not([class])` - so the class took the table out of all of it, not just the width control. The stylesheet rebuilt the gap with the *PDF's* look: a full grey grid at body text size, against the website's outer border, row rules and `.64rem`.

It rebuilds the theme's own appearance now, so a sized table is indistinguishable from an ordinary one - confirmed by comparing the computed style of both on the same page. The border colour was the part that would not have fixed itself: `--md-typeset-table-color` follows the colour scheme and the hard-coded `#555555` did not, so a sized table was wrong in dark mode by construction.

The PDF keeps its full grid, which suits print.

- **Fixed:** a directory tree is indented by one number, and set tighter (#486).

  Each level stepped in 98px on the published page, against the 30px `--tree-indent` asks for. Two thirds of that was accidental: the theme's own list margins were never reset, and a row's inset - the stub, and the hanging indent that keeps a wrapped description clear of its icon - accumulated into every level below it, because a child listing lives inside its parent's `<li>` and so starts from that `<li>`'s text rather than from its name.

  Subtracting a row's inset also settles where a rail hangs from. It now drops from under the icon above it at every depth; before, the top level stepped differently from the rest, because a top-level row carries only the hanging indent while every row below it also carries the stub, so a rule written for one was wrong for the other.

  `--tree-row` goes from 1.9em to 1.45em in the same pass: a tree is a dense index rather than body copy, and 1.9em left it looking airier than the prose beside it. The same 44-entry listing is a fifth shorter, and a wrapped description still separates clearly from the next entry.

  Both are now measured rather than reviewed - one test that every level steps by exactly one indent, another that a rail lands under the icon above it. The old stylesheet fails both.

- **Fixed:** a captioned figure is no longer narrower than the text around it (#485).

  An image asking for `width="100%"` came out 410pt in a 470pt column - short of the margin on both sides, while the paragraph directly above it ran the full

width. On the page it read as a picture floating inside a frame that never reaches the edges.

HTML's own default for `<figure>` is `margin: 1em 40px`. No website shows it, because every theme resets it; nothing here did, so the same markdown filled the column on the site and was inset in the PDF. 40px a side is 30pt a side, and 470 − 60 = 410. Measured in one document, captioned against uncaptioned:

```
with a caption      409.9pt
without a caption   469.9pt
```

The horizontal margin goes and the vertical stays, since the space above and below a figure is wanted. This reaches captioned tables too - a table caption is a `<figure>` - which is the same bug wearing different content, so both are tested.

- **Fixed:** the PDF build no longer announces a stage it does not run (#482).

  Every build ended with `[4/4] Rotating landscape pages` and rotated nothing. #469 replaced that post-processing step with a real landscape page box and deleted its module, but the stage title and its announcement stayed - and the announcement was the last statement before cleanup, so a three-stage build called itself four:

  ```
  [1/3] Preparing pages
  [2/3] Assembling the document
  [3/3] Building the PDF
  ```

  The count was self-consistent, which is why nothing caught it: the list of titles and the number announced agreed with each other, and only disagreed with what the build did. What separates a real stage from a leftover is whether anything happens afterwards, so that is what the new test measures - at the moment each stage is announced, the finished PDF must not already be on disk.

- **Fixed:** an image is no longer drawn off the edge of the paper (#480).

  The website holds an image to its column with the theme's own `img { max-width: 100% }`. The PDF stylesheet had no equivalent and nothing else clamped one, so a 1600px screenshot came out 1200pt wide on a 595pt page - most of it past the trim edge. Anything wider than about 627px overflowed, which is most screenshots taken on a modern display.

  The reported case, `{ width="50%" }` on an image, turned out to work: measured end to end through `prodockit pdf`, it renders at 235pt - half the 470pt text column - at every intrinsic size tried, with and without the `.screenshot`/`.pdf-only` classes. Both that and the overflow are now held

by tests, since the two look alike from a reader's chair: an image that ignores a width and one that runs off the page are both "the picture is the wrong size".

- **Fixed:** a hung CI job now fails in minutes rather than hours (#478).

  Three runs sat 35 minutes on `apt-get`, a step that takes nine to thirty-eight seconds. It was not the command: three other jobs in the same run ran the same step and finished, and GitHub reported all systems operational. Cancelling and re-dispatching cleared it.

  No job set `timeout-minutes`, so each inherited the six-hour default. A stall that costs seconds in the good case was free to hold a runner, a required check and a pull request for a working day - and to stay silent doing it, because a job that is running has not failed. Every job now carries a limit sized above its measured slowest run, and the `apt-get` steps carry one of their own so the failure names the stall rather than the job.

  The stall itself has a cause worth naming. The runner image lists `azure.archive.ubuntu.com` first and `https://archive.ubuntu.com` as a fallback; when the first is dead the fallback works, but apt waits out a 120-second connect timeout per source before reaching it. The steps now set ten seconds, well above a healthy mirror's response. Raising the retry count instead - tried first - made it worse: it re-asks the dead mirror rather than moving on.

  The scheduled drift check gets the widest margin relative to its work: it runs unattended, and the run nobody is watching is the one that must not hang.

- **Fixed:** the test suite no longer asks the real internet (#476).

  CI failed twice on a documentation-only branch and passed on a re-run with nothing changed. The bootstrap fixture replaces the command runner so stages answer from a table, and its docstring says that is so a test never goes to the network. That stopped being true when the site and Pages probes moved from launching `curl` to asking a URL through `fetch`: the fixture does not replace `fetch`, so two stages per test went to `github.io`, `api.github.com` or `pages.surrey.ac.uk` for real. Whether they answered decided whether "all 23 stages are set up" held.

  The fixture describes both seams now, `fetch` defaulting to a host that cannot be reached - the same choice `FakeRunner` makes in answering `127 not found`, so a test that has not said what a host does is not quietly given a working one.

  Three tests were passing for the wrong reason and are rewritten: they set `curl` runner keys nothing has read since the probes changed, so they no longer measured what their names claim. A fourth helper was dead code.

  The `sync-repo` tests made a live request each to decide badge visibility.

Nothing turned on the answer - stubbing it two ways gives the same passes - but it is a request per test that fails where there is no egress, so it is stubbed too.

Verified by running the suite with off-machine sockets refused: 1200 pass, the same as with them open. - **Docs:** `prodockit.steps` has a page of its own (#471).

It was the only user-facing extension without one - used once, in the bootstrap quick start, and otherwise documented only in its own docstring. That docstring also pointed at `docs/devcons/steps.md`, which has never existed, so the one signpost to the stylesheet a project is meant to copy led nowhere.

The new page covers the syntax, the `start` option, `attrs`, and the two traps the stylesheet exists to avoid - WeasyPrint ignoring `<ol start>`, and `em` resolving differently inside the number than in the connector.

Heading ids on both extension pages are now qualified (`#steps-writing-one`, `#tree-writing-one`), so two pages can share a heading name without one silently losing its anchor.

The tree page now ends with prodockit's own layout as a worked example - 44 entries, each checked to exist - so the extension is demonstrated on something real rather than on a fixture.

1.5 0.37.0 (2026-08-18)

- **Added:** `prodockit.tree` - a directory listing that looks like one (#379).

```
/// tree
docs/ - the documentation source tree
  index.md - the cover page
  stylesheets/ - CSS for both outputs
zensical.toml - project configuration
///
```

Indentation is the structure, a trailing `/` marks a directory, and `-` starts an optional description. Nothing else is typed, so the icon and the emphasis cannot disagree with what an entry actually is - which the hand-written list this replaces had no way to prevent.

Icons come from the project's own set: the block emits a shortcode and whatever icon extension the project already uses renders it, Lucide's folder and file by default. `directory_icon` / `file_icon` name others. Emitting SVG directly would have baked one project's icons into every document.

Ragged indentation is refused rather than guessed at. A listing is read for its

shape, so an entry attached to the wrong parent is a diagram that is wrong and looks right.

- **Fixed:** an inline SVG icon is no longer clipped in the PDF.

  `prodockit.pdf.html` encodes every inline `<svg>` to a data URI for Pandoc, and serialised it straight from BeautifulSoup - which parsed the page as HTML and lowercased `viewBox` to `viewbox`. SVG attribute names are case-sensitive, so the icon lost the coordinate system it scales into and WeasyPrint drew it at native size, clipped by its own box: a folder icon with its right-hand side sliced off.

  This affects every icon in every prodockit PDF, not only trees.

- **Changed:** `prodockit-table-rotated` is now `landscape-page`, and it works for any content (#469).

  The old name described a table and a rotation, and it is neither. The block gets its own landscape *page*, and what goes in it can be a table, a diagram, an image or anything else too wide for portrait.

  ```
  <div class="landscape-page" markdown="1">
  ```

- **Fixed:** that page is displayed landscape (#469).

  The block has always been laid out on a landscape *page box* - that is what makes its pagination and repeating header rows work, where a CSS transform did not. The finished page was then given the PDF's own per-page rotation flag, and a reader honours that by displaying a landscape page as **portrait**, with the content sideways. So the page a reader saw was portrait.

  Measured on the finished file, before and after:

  ```
  before   page 2: displayed 595x842 (portrait)   /Rotate=270
  box=842x595
  after    page 2: displayed 842x595 (landscape)  /Rotate=0
  box=842x595
  ```

  The box was landscape the whole time; only the display flag disagreed. The flag is gone, along with `prodockit.pdf.rotate` and the alternating recto/verso rotation `pdf_double_sided` applied to these pages - so the class carries the behaviour on its own, with nothing to configure. A document mixing orientations prints without special handling: a reader rotates each page to fit the paper by itself.

  Content longer than one page carries on across further landscape pages. Confirmed by building one: a 90-row table produced five landscape pages, each repeating its header row, with a diagram taking a landscape page of its own and the portrait document either side untouched.

1.6 0.36.4 (2026-08-18)

- **Changed:** the project stage says which address it is using, and where the real one is shown (#441).

  A GitLab group keeps its Name and its URL as separate fields, and renaming it changes only the Name. A group reading `assessment-commtest-2026` in the breadcrumb went on serving git at `comm058-2026`, so every derived URL missed - while the host said only `could not be found or you don't have permission to view it`, the same sentence it uses for a project that does not exist and for one you cannot see.

  Detected and reported rather than worked around. The group's real path cannot be read from here without credentials, so guessing a second address would replace a visible failure with a silent wrong answer. Instead the stage names the address it is using, says to read the browser's address bar rather than the breadcrumb, and shows a worked example of the two disagreeing.

  Per host, because both have the split under different names: GitLab's Name against its URL, GitHub's organisation display name against the part after `github.com/` - and a personal account, where there is no split at all.

- **Changed:** the site and Pages probes ask the URL directly instead of running `curl` (#449).

  Both stages made an HTTP request by starting another program, and that cost three fixes which were never really about curl: it arrived four stages after the first check that wanted it (#374), it was told to write to `/dev/null` on a platform without one (#443), and a `curl` typed at a PowerShell prompt resolved to `Invoke-WebRequest` and looked like an absent program, which sent a diagnosis the wrong way for an afternoon.

  `urllib` needs nothing installed, on any platform, at any point in the run - which also dissolves the ordering problem #374 documented, since there is no longer anything to install first. The `/dev/null` class of bug is gone by construction: nothing writes a response body anywhere.

  Two behaviours were kept deliberately and are covered against a real server rather than a mock. Redirects are **not** followed, because a `302` is how a login-walled site is recognised as published - a Surrey Pages URL answers `302` to GitLab's own OAuth consent page, and following it would report the consent page's `200` and call that site publicly readable. And "could not establish" stays distinguishable from "not there": a refused connection, a DNS failure or a timeout returns `None`, which is not a status.

  Tests describe a host through a `fetch` seam on `Context`, the same way they already describe a filesystem through `exists`.

- **Fixed:** the Ubuntu VS Code install no longer stops for a dialog (#428).

The `.deb` asks a debconf question part-way through installing:

```
Add Microsoft apt repository for Visual Studio Code?  <Yes>
<No>
```

`apt install -y` does not answer it - `-y` agrees to apt's own questions, not to a package's debconf ones. And bootstrap captures its subprocesses, so the dialog was invisible: the run simply stopped, looking no different from slow work.

The answer is now preseeded *before* the install, so it is part of what the reader approves rather than something they meet alone. Answered "yes" deliberately: that repository is how VS Code then updates through apt, and declining would pin them to whichever build the `.deb` happened to be.

The answer is written to a file by Python and read by `debconf-set-selections`, rather than piped into it. The pipe is the problem, not the echo: `echo ... | sudo debconf-set-selections` puts `sudo` inside a shell, and a sudo timestamp expiring there prompts where nobody is watching - the same invisible wait this fix exists to end, reintroduced by the fix itself.

- **Fixed:** a `.pdf-only` image is centred in the PDF (#462).

  Sizing an image differently for each medium - `.web-only` at one width, `.pdf-only` at another - left the PDF copy hard against the left edge, under its own centred caption.

  `.pdf-only` sets `display: block` so that a website's `display: none` is undone. That also took the image out of the one mechanism that positions it: `figure { text-align: center }` moves inline content, and a block box is placed by its margins instead. Measured inside the figure at 0.00pt left against 163.96pt right, where the same image without the class sits at 80.98/82.98.

  Images are exempted from that `display: block` rather than centred with `margin: auto`, which reads like the obvious answer and does nothing: WeasyPrint leaves a block-level replaced element at the left edge even with auto margins on both sides. Inline also keeps one centring mechanism for every image instead of two that have to agree.

- **Fixed:** the installation requirements say what the project actually declares (#372).

  The table had drifted from `pyproject.toml`: `Markdown` was recorded at `⩾ 3.4` long after the real floor moved to **3.10.3**, `zensical` carried no version at all, and `pymdown-extensions` was missing entirely - a dependency `prodockit.pdf` matches class shapes from. None of that breaks a build; all of it sends a reader to install the wrong thing.

`weasyprint` and `pandoc` were also filed together as "external binaries". They are not the same kind of thing: weasyprint is a `pip install` away and simply is not a dependency of prodockit, while pandoc genuinely has no Python package. A reader treating them alike goes looking for one that does not exist, or misses one that does. They now sit in a table of their own that says which is which, alongside the Node major version, the browser Mermaid renders through, and pandoc's floor and pinned release.

The citation style is documented too, as a download rather than an install: pandoc resolves it from the working directory and every CI script fetches it before building. It is deliberately not vendored - third-party content with its own licence and release cadence, which a committed copy would let go stale while every build kept succeeding - and `.gitignore` now keeps a local copy out of commits.

A test now holds the table to `pyproject.toml` - every declared dependency documented, every floor matching, and the versions `prodockit bootstrap` enforces stated - because this drift was invisible for months and nothing else would have caught it.

- **Fixed:** unassessed work is no longer asked for a course code (#458).

  The Surrey questions asked for the course code one question *before* learning whether the work was assessed at all - so an unassessed run answered it and the answer was thrown away. Neither `namespace_for()` nor `project_name_for()` runs on that path; it asks for the namespace and the repository name directly instead.

  "Is this an assessed assignment?" now comes first, and the course code is asked only where it is used.

  The two paths necessarily total different numbers once they diverge: assessed asks three more (course, stage, year) and stays at seven, unassessed asks two more (namespace, repository) and drops to six. The change of total is announced where it happens, the same way the host question already announces the eight-to-seven switch when Surrey's GitLab is chosen - a reader who has just seen `4/7` is owed the reason the next line reads `5/6`.

1.7 0.36.3 (2026-08-17)

- **Changed:** `prodockit pins` watches `prodockit` itself (prodockit-template#173).

  Its absence from the managed set had exactly the consequence the set exists to prevent: `prodockit-template` pinned `prodockit==0.35.0` and drifted two releases behind with nothing noticing, because the one command that looks at pins was not looking at this one. Moving it needed `-p prodockit` typed by hand, which is the step nobody remembers to take.

Adding it found a second case immediately: `prodockit-userguide` pins `prodockit[index]==0.21.0`, fifteen releases behind.

Safe in prodockit's own repository, where the name is the project's identity rather than a dependency: the pattern requires a version operator after the name, so `name = "prodockit"` and the adjacent `version` are not declarations. Without that, the command would offer to rewrite the release number of the package being built - which is why there is now a test saying so.

- **Changed:** pandoc is pinned on Windows, and a local pandoc that differs from the builds' is named (#454).

  Only Ubuntu pinned it. macOS took `brew install pandoc` and Windows took whatever winget was serving - so a machine bootstrap had just set up ran 3.10.2 while every build in this family pins 3.10.1, and nothing said so.

  That matters because pandoc is a *rendering* input. #207 was code blocks coming out as justified prose on an older major, and `limitations.md` records 3.1.3 accepting markup 3.10 does not. A student writes on one pandoc, CI publishes on another, and the PDF they checked is not the one that gets marked - silently, because both builds succeed.

  Windows now installs `--version {PANDOC_VERSION}`, matching what Ubuntu has always done. macOS cannot: Homebrew has no way to install an old pandoc. So the check *says* when the local version differs rather than failing - a failing status there would be one no reader could ever clear, which is the stuck-stage failure #443 and #451 were both about. A pandoc too old to render correctly still fails, because that one is fixable.

1.8 0.36.2 (2026-08-17)

- **Fixed:** Pandoc and Node are no longer reported missing when they are installed (#450).

  Both checks ran a bare `pandoc` / `node`, so they saw only this process's PATH. A `winget install` sets the *machine's* PATH and a running process never receives it - so the check failed on precisely the machine that had just installed the software, while git and VS Code, two stages above in the same listing, reported themselves found by full path.

  Not merely cosmetic: the stage then offers to install what is already there, winget answers "already up to date", and the check fails again, so the run cannot move on.

- **Fixed:** `sync-repo` finds git the way the stages do (#451).

  It invoked git by bare name and died on the same stale PATH:

```
Error: could not run git: [WinError 2] The system cannot find
the file specified
```

The same applied to the Jinja macros and the source bundle. All of them now resolve through one list of install locations, shared with the bootstrap stages - two lists would be two answers about one machine, which is how these drifted apart in the first place.

- **Fixed:** a sync check that could not run is no longer reported as a difference (#451).

  `sync-repo --check` exits non-zero both for "there is a difference" and for "I could not look", and the stage said *"origin is right, but the project config still needs syncing"* to both - naming a cause nobody had established and sending the reader to run something that would not have fixed it. It now says what actually stopped the check.

- **Fixed:** the site stage speaks about the host the reader is actually on (#444).

  Every instruction on that stage was a literal describing GitHub, so a GitLab reader was sent looking for a gear beside 'About', a 'Use your GitHub Pages website' tick-box and a Settings > Pages - none of which exist there.

  One of them was worse than merely wrong. It said the site "will be public" and that "a private repository does not make a private site", which is true of GitHub and false of GitLab: a private project's site stays behind the instance's own sign-in. An anonymous request to a published Surrey site answers `302` to GitLab's OAuth consent page. So a student was told their drafts were readable by anyone with the link, contradicting what the project stage had told them about the same project.

  Carried as per-host values rather than a branch in the stage, which is how `ssh_keys_steps` and `project_visibility` already work - the model's own note is that a difference by host belongs as "a value rather than a branch in the stage".

1.9 0.36.1 (2026-08-17)

- Zensical pinned to **0.0.55** (from 0.0.53), after building the site and the PDF under both and diffing the results.

  The PDF is **byte-identical** - 1,469,185 bytes either side. The site changes 23 HTML files, each by exactly two lines: the `generator` meta tag and the JavaScript bundle's hashed filename. No page differs by more than that, and the stylesheets, workers and licence files are byte-identical - so unlike 0.0.52, which redrew the GitHub brand icon across every page, nothing rendered changes here at all.

The bundle grew 624 bytes (165,840 → 166,464), which is the `ui` v0.0.25 bump in 0.0.54.

Upstream between the two: peak memory cut 8-10x, a `webbrowser` CVE dependency bump, `mike` defaults corrected, an autorefs false-positive fixed, code-span detection fixed in the reference extractor, and surrounding whitespace now ignored in inline link targets. Three of those touch Markdown and reference handling and so could have changed the output; measurably they did not.

The coupling pass in Zensical coupling was run rather than assumed: full suite green (1157), the integration tests that exercise the per-page render context green (70), the `zensical/__init__.py` export surface unchanged, and the count of unresolved `??` cross-references identical either side.

- **Fixed:** the first push no longer forces, so a protected branch accepts it (#442).

  A repository created with "initialize with a README" holds a commit this project's history does not, so an ordinary push is rejected. That was answered with `--force-with-lease`, which GitLab refuses outright on a protected branch - lease or no lease, the rule is about the operation:

  ```
  remote: GitLab: You are not allowed to force push code to a
  protected
  branch on this project.
  ```

  The host's commit is now merged in with `-s ours`, which takes this project's tree entire and leaves the README's behind - the same result the force push had, reached by a fast-forward. On an assessed repository the protection is the point and is rarely the student's to switch off.

  It is also the safer half of the trade: a lease fails only if the remote moved past the commit it names, while an ordinary push fails if the remote moved at all, and cannot destroy anything even in principle.

- **Fixed:** the site probe can run on Windows (#443).

  It discarded the response body into `/dev/null`, which Windows has not got. curl does not read that as a device - it tries to create the file, fails, and exits 23 - so the probe reported "could not check ... the probe did not run" about a site that was serving, on every retry, for as long as anyone kept answering yes. `NUL` is the equivalent there.

  Reporting could-not-establish rather than "missing" is what kept this from being a wrong answer. It is still a state nothing could leave.

- **Changed:** the project stage quotes what the host said (#439).

```
11  MISS  Your own project on the host - nothing visible at
          git@gitlab.surrey.ac.uk:assessment-commtest-2026/
report-... -
          gitlab.surrey.ac.uk said: The project you were looking
for could
          not be found or you don't have permission to view it.
```

It used to say "nothing visible at ..." and stop there, so a reader who had just created the project saw the same eleven words on every retry, with nothing new to work from.

The host's own reply distinguishes what the stage cannot: a project at another path, a group you cannot see into, and a key the host will not accept need three different things from you. Git's own wrapper lines are left out - "Could not read from remote repository" is not something anyone can act on - and a host that says nothing gets no words put in its mouth.

1.10 0.36.0 (2026-08-17)

- **Fixed:** the ssh-agent step ended the run without looking (#435).

  0.34.0 stopped that step looping, but overshot: the moment you said the service was started, the run ended and asked you to start again.

  A started service usually *is* visible to the next command that looks - `ssh-add` opens the agent's pipe afresh each time it runs - so the check is given its chance, and a run that can carry on carries on. Only where it still cannot be seen does the run end, because asking again would put the same question to the same unchanged answer.

  The message no longer says to open a new window either. There was never anything a new window would do that running it again would not.

- **Changed:** the assessment question comes before the year, and unassessed work is asked for its namespace (#437).

  The year question explains itself in terms of SRA and LSA, and nothing before it had said what those are - the question that introduces them, and lists them, came afterwards. They are the other way round now, so the year question refers back to something already on screen.

  The stage question is numbered too. It used to arrive after "is this assessed?" without a number of its own, so a reader counting down from six met a question that was not in the count. Seven either way now, and every one of them numbered.

  Unassessed work is no longer asked for a year it has no use for, nor for a stage it has not got. It is asked for its namespace and its repository name instead, offered as `<your login>` and `report-<your login>` - a project

group and a name of your own are perfectly ordinary answers, and a default is only an offer.

- **Changed:** the Surrey questions are worded as they were asked for (#420).

  The login question had lost its example, which was the part that made it answerable - a reader logs in with `ab1234@surrey.ac.uk`, so being asked for "the six-character ID" is a question about something they may never have typed. The year question had lost the rule for a resit, and gained a different one of mine.

  Both now read as specified: the login question shows the address and the six characters to take from it, and the year question says an SRA or LSA should be the year *prior* to the year the retake is being assessed.

- **Fixed:** a first run printed the stage list over the details it had just told you to note down (#433).

  The namespace and repository name are the only two values a reader has to carry from this command to a browser, and twenty-three stage lines printed after them scrolled both away. `--configure` already stopped after saving; a first run answered the same questions by a different route and carried on.

  It stops there too, and says what to run to see the stages. Filling a single missing value still reports, because nothing has been announced that a stage list would bury.

1.11 0.35.1 (2026-08-17)

- **Fixed:** a first run did not take Surrey's shorter configure path (#430).

  ```
  Some details are not set yet: host, full_name, email, ...
  1/8 The git host your project lives on [gitlab.surrey.ac.uk]:
  3/8 The email address used for your gitlab.surrey.ac.uk login
  []:
  ```

  Eight questions, asking for the email 0.35.0 exists to derive. The shorter path was chosen only when `--configure` asked for everything, and a first run does not arrive that way: nothing is set, so bootstrap offers to fill the gaps and asks for the fields by name.

  Nothing set at all is the configure arriving by a different door, not a repair, and is treated as one. Coming back later to correct a single value still asks for that value.

1.12 0.35.0 (2026-08-17)

- **Changed:** VS Code is driven from inside the application when `code` is not on PATH (#424).

  On macOS the application and the command are two different installs. The app is dragged to Applications; `code` arrives only when somebody runs "Shell Command: Install `code` command in PATH" from inside it - which readers routinely have not, while the binary sat in the app bundle the whole time.

  It is found there now, so the stages that drive VS Code work without it. The stage still says `code` is not on PATH, and how to add it, because that is a convenience worth having in your own terminal - it is no longer a thing that stops the run.

  Windows keeps its own wording, which is true there and would be false here: an app bundle is not on `PATH`, and a new terminal will not change that.

- **Fixed:** a repository created with a README was reported as pushed when nothing had been pushed (#423).

  Ticking "initialize this repository with a README" makes a commit your project's history does not contain. The stage asked only whether the remote had *anything* on it, so it answered `ok` about a project that had never been pushed - and the site stage then found nothing published, with nothing on screen to connect the two.

  It now compares the commit here with the commit there. Where the only thing on the host is that README, the push replaces it, pinned to exactly the commit that was inspected - so anything arriving in between fails the push rather than being overwritten. Where the host has anything else at all, nothing is pushed over it and the stage says to go and look.

- **Fixed:** the bootstrap page described eighteen stages when there were twenty-three (#413).

  Five stages missing from the table, the numbering shifted under the ones that remained, and a sentence about which stages are platform-independent naming numbers that had moved. A reader on a finished setup was told `All 23 stages are set up.` by a page listing eighteen, with no way to tell which was wrong.

  The table now names every stage, and a test holds it against the stage list itself - count, order and wording - so it cannot drift again. The count is no longer written into the prose: a number in a sentence goes stale in silence, which is how this lasted five releases.

- **Added:** a shorter `--configure` for Surrey's GitLab (#420).

```
1/8 The git host your project lives on [gitlab.surrey.ac.uk]:

  gitlab.surrey.ac.uk fills in the rest from your login ID and
course
  code, so this is 6 questions rather than 8.

2/6 Your full name, as it should appear on commits:
3/6 The six-character ID you log in to Surrey with, e.g.
`ab1234`:
4/6 Your course code, e.g. `comm058`:
5/6 What year does the module start in? [2026]:
6/6 Is this an assessed assignment? [Y/n]:
```

Your email, GitLab username, group and repository name all follow from those, so they are no longer asked for. Assessed work goes to `assessment-<course>-<year>`, with `-sra` / `-lsa` after the year for a resit; unassessed work stays in your own namespace, where no year applies.

The year is the one the module *starts* in, offered as this one. Both the awkward cases are in the question rather than in a handbook: a semester 2 module should be the year after the Christmas break, and for SRA and LSA the year should be the year prior to the year the retake is being assessed.

The repository, and the folder it lands in, are named `report-<course>-<year>-<login>`, with `-sra` / `-lsa` on the end for a resit. The name carries the year even for unassessed work, where the namespace does not: those repositories sit side by side in one personal namespace, and two years of the same module - or a first attempt and its resit - would otherwise be two repositories with one name between them. The repository is `report-<course>-<login>`, and the run ends by telling you both, since you have to find them on a website afterwards.

Every free-text answer removed is one fewer chance to type a namespace that does not exist and hear about it six stages later, from a host that says only "not found".

Nothing here applies to github.com or gitlab.com, where these rules are simply untrue - those keep the eight questions.

1.13 0.34.0 (2026-08-16)

- **Added:** `prodockit.steps`, numbered steps a reader works through in order (#378).

```
/// steps
    start: 9
```

```
//// step | Load the key into the agent
```bash
ssh-add --apple-use-keychain ~/.ssh/id_ed25519_gitlab
```

////

/// ```

A procedure is not a list of facts. Each step is a thing to stop and do, so it gets a number to find your place by, room for a command and its explanation, and a line joining one step to the next.

Built on pymdownx's Blocks API - the machinery Material's own admonitions and tabs use - so `attrs` works as it does everywhere else, and a step's body holds paragraphs, code or a table without indentation arithmetic.

`start` exists because a long procedure is often split across sections. It is written into the HTML twice, deliberately: a browser reads `start`, and WeasyPrint ignores it entirely and numbers from 1. Emitting both from one setting is the reason this is an extension rather than a documented HTML snippet - a pair maintained by hand drifts, and the failure is silent and PDF-only.

The bootstrap page now opens with a six-step quick start written in it, which is what exercises it.

`pymdown-extensions` is now a dependency. It was deliberately not one before, on the grounds that prodockit never imported it; this imports it, and python-markdown constructs an extension by importing its module, so there is no lazy route.

- **Fixed:** a push the host refused left the stage unable to try again (#414).

```
[main (root-commit) 5e98636] Initial commit
...
remote: You are not allowed to push code to this project.
failed: exit status 128 - see the output above
```

The commit was made and the push declined, which leaves the project committed here and empty there. On the next run `git status` is clean, so `git commit` exited 1 with nothing to commit and the run stopped *before* the push - failing for a reason that was neither true nor the obstacle. The commit is now run only when there is something to commit.

The refusal is named, too. Bootstrap runs those commands with the terminal attached, so their output goes to you and not to it, and `exit status 128` was all it could say - after sixty-eight `create mode` lines had scrolled the real sentence off the screen. From that one state, and no other, it asks the host
```

whether a push would be accepted, and says so: the key is fine, the account is not allowed to write there.

1.14 0.33.0 (2026-08-16)

- **Fixed:** the VS Code extensions stage said VS Code might not be installed, having just installed four extensions through it (#410).

```
Extension 'ms-python.python' v2026.4.0 was successfully
installed.
...
ran, but still not right: could not list extensions - is VS Code
installed?
```

The plan found `code.cmd` where the installer puts it; the check asked for a bare `code`, which on Windows cannot run at all - `CreateProcess` appends `.exe` and nothing else. So the stage could never pass there, whatever was installed, and the run stopped on it because later stages depend on it.

Third instance of one shape - after git (#390) and npm (#405) - so there is now a test that no check asks for a tool by a name it knows how to resolve.

- **Added:** `prodockit pdf` says which stage it is on, and how long the build took (#375).

```
Building PDF from zensical.toml...
  [1/6] Preparing pages
  [2/6] Assembling the document
  [3/6] Building the PDF
  [4/6] Collecting index entries
  [5/6] Rebuilding with page numbers
  [6/6] Rotating landscape pages
Wrote docs/site_documentation.pdf in 1m 35s
```

One line at the start and one at the end left minutes of silence in between, and a silent terminal is indistinguishable from a hung one.

A build with an index has two more stages than one without: the page numbers an index needs do not exist until the document has been laid out, so the whole thing is built, read, and built again. That is worth seeing rather than wondering about.

1.15 0.32.1 (2026-08-16)

- **Fixed:** the Node stage failed with `npm: not found` having just installed Node (#405).

Two things were true at once. A plan is written before any of it runs, so npm was resolved while Node was still absent and fell back to the bare name - and a bare `npm` can never run on Windows, because `CreateProcess` appends `.exe` and npm is a `.cmd`.

Refreshing `PATH` could not help, because the problem was the name rather than the path. Names are now resolved as each command is about to run, which is the only point at which the answer can be right.

- **Changed:** your answers live beside the project, as `.pdk-bootstrap.toml` (#373).

  There was one config per user, so setting up a second project overwrote the answers for the first - its namespace, its name, the directory it lives in - and the original could not be re-checked without answering everything again.

  One file per directory means one per project. The old `~/.config/prodockit/bootstrap.toml` is still read when a directory has no file of its own, so nothing has to be moved and a setup already answered keeps working.

  The file holds your name, email and username, and the first push commits with `git add -A` - so where the directory is a repository, it is added to `.gitignore`.

- **Changed:** the closing line says how to apply the setup, not only how to look at it (#376).

  ```
  14 of 23 stages need work.
  Run with --dry-run to see the exact commands that would fix
  them.
  ```

  `--apply` is the one a reader has come for, and it was the one option not mentioned. Both ends of a run had the same gap: after a dry run, nothing said how to run what had just been printed either.

- **Changed:** the run says which prodockit is doing the work (#399).

  ```
    Running:  pdk from C:\users\buckwem\GitHub\.venv\Lib\site-
  packages\prodockit
  ```

  A report arrived headed with one version, showing commands that version had not contained since the release before - an older install doing the work, in a window that had never been reopened. The version alone could not show that. The path can, and it is the first thing to check when a run does something the source says it cannot.

- **Fixed:** git installed a moment ago was reported as not installed (#390).

```
[2/22] Git, installed and configured
        git is not installed
...
Found an existing package already installed.
```

The installer puts git on the *machine's* `PATH`, and `PATH` is read when a process starts - so a window that has not been reopened since cannot see it. VS Code has had this answer since 0.27: find the executable where the installer puts it and use it by its full path.

Git now shares it, and every stage that runs git uses the same answer. Fixing only the check would have moved the failure a dozen stages down to the clone.

- **Fixed:** the ssh-agent step asked again after you had answered it (#397).

```
Have you started the ssh-agent service? (yes/no): yes
not there yet - no ssh agent is running
Try again? [Y/n]:
```

On Windows the service is started from a separate Administrator window, and the run in your own window cannot see that happen. So it re-checked something that could not have changed, reported it missing, and asked again - which reads as the tool ignoring the answer it was just given.

The run now ends there, saying what to type next, and exits zero. Nothing failed: the step was done, and a new run is what it takes to see it.

1.16 0.32.0 (2026-08-16)

- **Fixed:** Windows on ARM stopped at the Pandoc stage, having just installed MSYS2 successfully (#393).

```
Successfully installed
MSYS2 is not at C:\msys64 - install it there, or run ...
failed: exit status 1 - see the output above
```

Two assumptions, both about the machine rather than about this project. `C:\msys64` is the installer's default, not a promise; and an arm64 Windows gets the arm64 build of MSYS2, which has no MINGW64 environment at all - its native one is CLANGARM64, with different package names and a different DLL directory.

Neither can be known from here, so both are settled on the machine when the step runs: several locations are searched and the one found is used, the

environment follows the processor architecture, and the directory added to `PATH` follows the environment. When nothing is found it says where it looked.

- **Added:** Surrey's GitLab Pages address is derived rather than asked for (#392).

```
Note: cannot derive a published URL for GitLab; site_url left
unchanged
```

printed for an instance whose layout was perfectly well known. `https://gitlab.surrey.ac.uk/mb0105/report` publishes at `https://mb0105.pages.surrey.ac.uk/report/`, so `sync-repo` writes that, and the site stage has an address to check.

Only instances somebody has actually run against are derived, and they are listed. GitLab's default layout is `<namespace>.pages.<instance domain>/<project>`, but the instance domain is an administrator's setting that nothing in a remote URL reveals - a confidently wrong canonical URL is worse than none, so every other instance still declines and uses `pages_base`.

The site stage now reads a login wall as proof rather than absence. A university instance publishes behind its own sign-in, and "is not answering yet" of a site that is plainly up would leave every Surrey run one stage short for ever.

```
Note: could not tell whether GitLab is public; badges left as
they are
```

is gone too, on hosts where it could never be answered. A self-hosted instance shows a stranger a login page, and nothing turned on the answer: the badges that do come from shields.io, which cannot read that host either. Assumed private, and no longer reported.

- **Added:** the first stage checks that prodockit is running from a virtual environment of its own (#381).

```
 1  MISS  prodockit runs in an environment of its own - running
from
          /usr/bin/python3, which is not a virtual environment
```

There are two environments in a finished setup, and they are easy to confuse: **prodockit's own**, which `pdk bootstrap` runs from, and **the project's** `.venv`, which holds Zensical and everything else in `requirements.txt`. The second is built by the first.

So the first is a prerequisite for the run rather than a step within it, and it is asked first. On a system Python the run used to fail fifteen stages later -

Debian and Ubuntu refuse `pip install` outside a virtual environment and ship `venv` without `ensurepip` besides - in words about the project rather than about the interpreter building it.

Where the missing piece is a package, it is installed. Either way the exact commands for your platform are printed, including the three that put prodockit in an environment of its own. Nothing can repair this in place: a new environment needs a new process, so the steps are shown and the next run confirms them.

The header said "with the Python virtual environment active" without saying which. It now names it. Twenty-three stages.

1.17 0.31.1 (2026-08-16)

- **Fixed:** the GitHub Pages stage reported itself done without having looked (#374).

```
11  ok    Pages switched on - cannot be seen from outside a
private repository
```

Pages had never been switched on. A private repository answers `404` to every anonymous caller, and "cannot be seen" was being printed as though it meant "is set up" - so the one stage on the one host that has to be done by hand was skipped in silence.

It is a finding now, and shows the steps. It still does not claim to have looked: a check that cannot settle itself says so, takes your word once, and leaves the proof to the site check at the end of the run. A project that has already published is recognised from its own site, so a private repository no longer carries a finding it could never clear.

- **Fixed:** three checks reported a look they had not taken.

curl is installed by the Pandoc stage, several stages below the first check that wants it, so on a machine part-way through a setup the probe can simply be missing - and `curl: not found` was reaching you as "cannot be seen from outside a private repository" for Pages, and "is not answering yet" for the site. Both now say the probe did not run.

The stage that looks for your repository no longer reports "is not reachable" when the host has answered. github.com says `Repository not found.` for a repository that is missing *and* for one your key cannot see, so the steps now tell you to look before creating anything: an issued repository carries the permissions that decide who can read your work, and a second one will not have them.

- **Changed:** the manual steps ask you to type `yes` (#374).

`[Y/n]` is answered by pressing Enter, and a reader twelve stages into twenty-two presses it in rhythm - which at a browser step means claiming to have done something they have not. All three now want the word. `no` is a real answer: it leaves the stage outstanding rather than pretending it was done.

- **Fixed:** the configure prompt named github.com whatever host you had answered (#370).

```
5/8 The group, organisation or user the project lives under
    (e.g. comm058-2026, or your own username on github.com) []
```

asked of a setup against gitlab.surrey.ac.uk. The host is the first question so that the rest can be phrased in terms of the answer, and naming a different service reads as a question about a different account somewhere else. It now says your own gitlab.surrey.ac.uk username, or whichever host you gave.

- **Changed:** the stage that looks for your project now says which address it asked about (#377).

```
 7  ok    Where the project comes from - nothing visible at
          git@github.com:mb0105/report-ubuntu-v1.git - the
template
          will be cloned
```

It used to report "no existing repository found", which gave a reader whose project plainly exists nothing to work with. The namespace is one setting shared by every host, so a setup that changes host carries the previous host's namespace to the new one - and the address on screen is what makes that visible.

It no longer claims the repository is absent, either. Both hosts answer a private repository your key cannot see with the same words they use for one that does not exist - github.com says `Repository not found.` either way - so "nothing visible at" is as much as the question can establish.

- **Fixed:** setup stopped on Windows at "Clone pointed at your project", saying prodockit was not found (#371).

```
Commands finished, checking the result...
failed: prodockit: not found
Stopping - later stages depend on this one.
```

Said of a machine where prodockit was installed and running the bootstrap that reported it. Two stages run prodockit commands of their own - `sync-repo` when the clone is repointed, and the MathJax install - and both named `prodockit` bare, leaving the machine to find it a second time. A virtual

environment's scripts are reachable when it launches one; they are not necessarily on the `PATH` a child process inherits.

Both now run through the interpreter already running them, which also settles *which* prodockit: the one doing the work, rather than a different install earlier on `PATH`. `python -m prodockit` works as a command in its own right.

- **Fixed:** a finished setup was told its project needed a decision (#368).

  Every stage done, `prodockit boot` run once more to confirm, and the stage that asks where the project comes from reported `MISS` and put three choices to somebody whose project was already cloned, pushed and published. None of the three could change where the contents had come from: they were on disk.

  It now says where they came from. A clone still on the template keeps the question, because there the decision really is ahead - and an `origin` that cannot be read stays unknown rather than being taken for an answer.

  A check pass on a finished machine now costs three connections to the host rather than four.

- **Added:** `source`, a short name for `source-bundle` (#366).

  The longest name on the list, typed at a prompt while a submission is being checked. `bootstrap` already answered to `boot`; both now come from one table, so a third alias is one line rather than three places to keep in step.

  The long names all stay - they are what the User Guide, the changelog and anything scripted use.

- **Fixed:** the brand logo stayed the template's until something cleared the cache (#364).

  `prodockit sync-repo` rewrites the icon in `zensical.toml`, and a build served from `.cache/` kept showing the old one - readers were clearing it by running `zensical serve` and wondering why it changed.

  The commit-and-push stage builds once with `--clean` before committing. It also means the first push is the first time the project is *proved* to build, rather than that being discovered from a red pipeline minutes later.

1.18 0.31.0 (2026-08-13)

- **Added: gitlab.com is a supported host** (#361).

  Declared since the beginning and refused, because nothing had been run against it. It is covered by tests rather than by a machine - Surrey's own instance has been unreachable, so no GitLab path has been run end to end - and that is worth knowing before relying on it.

A self-hosted instance of either family is still refused. What cannot be guessed for one is where it publishes its Pages and where its API lives, and inventing either would send a reader somewhere that was never going to answer.

- **Changed:** a green stage says why, and the history reset defaults to yes (#356).

  "Where the project comes from" reporting `ok` read the same whether the host had been searched and found empty or never reached at all - the ambiguity behind #344 and #351. It now says `no existing repository found - the template will be cloned`, or `could not reach <host> to look`, and the apply loop prints the detail rather than discarding it.

  The history reset defaults to yes. It deletes only ever the *template's* history - a clone carrying the reader's own is never offered it, and the stage is blocked while the decision is unmade - so defaulting to No left anyone who pressed Enter with the template's commits behind their project. - **Removed: the `gh` / `glab` requirement** (#357).

  Verifying Pages through a host CLI meant installing a tool, authenticating it in a browser, from the right directory, on every machine - four ways to go wrong, each of them hit in testing.

  A public repository reports `has_pages` to any anonymous caller, and the published site answers anonymously even when the repository behind it is private. So Pages is read without a token where that is possible, and where it is not the stage says so and leaves the proof to the site check at the end of the run - which needed no token in the first place.

  The About > Website link is an instruction now rather than an authenticated call: open the repository, click the gear beside 'About', tick 'Use your GitHub Pages website'.

  The Pages stage is GitHub's alone now, too. GitLab configures its own Pages from the CI job, so a GitLab reader was being shown GitHub's steps - an instruction to do nothing - and the metadata URL those steps were checked against was `api.github.com` written into the stage, which a gitlab.com project would have been asked about. Both are fields on the host.

  Nothing that was proven stops being proven; it moves one push later. Bootstrap is 22 stages.

- **Fixed:** bootstrap asked you to sign in with a tool it had not installed yet (#354).

  The stage reported `gh is not installed` and the next line said "Run `gh auth login`". Both halves were in `instructions`, which are printed before a plan's commands - so the install came after the request to use it.

The sign-in is `follow_up` now, which exists for work that only makes sense once the commands have run.

1.19 0.30.1 (2026-08-12)

- **Fixed:** `--apply` trusted checks taken before any stage had run (#351).

  A project with work on the host was cloned over with the template, and the reader was never asked: "Where the project comes from" reported `ok` because it had been checked *before* the SSH stages ran, when the host could not be reached.

  Each stage is checked again when the loop reaches it now. Earlier stages change the machine the later ones are about - that is what a setup tool is - so a single snapshot taken up front was never going to hold. The memo keeps repeats free within a pass and is dropped between stages, which is what makes the second look real.

1.20 0.30.0 (2026-08-12)

- **Fixed: the clone decision is now actually offered**, as a stage between the SSH stages and the clone (#348).

  `--configure` runs before there is an SSH key, so it could never see whether the project existed and had to say it could not look - which left the three-option question unreachable on every first run, the one where it matters most.

  It is asked at the first point the answer is knowable, and the last at which it still matters. Nothing to decide stays `ok` rather than becoming a question: a project that does not exist, or exists and is empty, gets the template and nobody is asked to choose between one thing. A recorded answer is read, not asked again.

  A plan can put a numbered choice now rather than a yes/no. Three paths as three consecutive yes/no questions invites pressing Enter through them, and one of these deletes commits that cannot be recovered - so it has no default.

- **Fixed:** whether a repository holds a *project* was being judged by "has any commits" again. The check for `zensical.toml` and `README.md` was written for #332 and lost when `stages.py` was reset during #339, so it never reached a release. A pass costs four host connections now rather than three, which the test that tracks that number records.

- **Added:** `boot`, a short name for `bootstrap`.

  `pdk boot --apply` is the form this is typed in most, and it is typed repeatedly while a machine is being brought up. Registered rather than

wrapped - one command object under two names - so the two can never take different options or drift in their help. `bootstrap` stays.

1.21 0.29.0 (2026-08-12)

- **Changed:** CI caches the pandoc download, so an outage at the release CDN cannot stop a build.

  It returned `503` for hours one evening and failed every job in the matrix before a single test ran, three times across three branches. The retry added earlier rides out a blip and did not help with that: it fired five times and still gave up.

  Keyed on the pinned version, so a bump fetches afresh and an unchanged pin never touches the network again. The retry stays for the first run on a new key, and for the day the cache is evicted - a cache is a saving, not a guarantee.

  Only affects building this project; nothing in the package changes.

- **Fixed:** a repository bootstrap could not reach was reported as not existing, and a partial run never asked where the project comes from (#344).

  On a fresh machine there is no SSH key until stage 3, so `git ls-remote` fails on authentication - and that was read as the repository being absent. A reader was told their work was not on the host while it was sitting there.

  Only the host saying so counts as absent now. A refused key, an unreachable network, a name that will not resolve: none of them is evidence about the repository, and "cannot tell" says so plainly.

  Separately, `missing_keys` omits `source_url` - blank is a valid answer - so a run filling in a few gaps skipped the question about an existing project and ended without its summary. Both paths ask it now.

- **Added: the host's own command line** is installed and signed in as a stage (#342).

  `gh` for GitHub, `glab` for GitLab - including a self-hosted one, which is still GitLab. It is the only thing that can answer questions no anonymous caller can: whether Pages is switched on, and what the repository's About panel says. It holds a token so bootstrap does not.

  Signed in counts as much as installed - an unauthenticated tool is installed and useless, and calling that done would leave the stages depending on it failing for a reason two stages away. `auth login` opens a browser, so it stays the reader's to run.

- **Added: Pages is a stage of its own**, straight after creating the project (#341).

It was a trailing item on the "create your project" list and was missed twice, at a cost of a red first build whose error names the site rather than the setting. Now it is asked while the reader is still in the browser, and before the push, where missing it breaks the build.

The site stage also fills in the repository's About > Website field, using `gh` for the same reason. GitHub does not set it from Pages, so a project with a perfectly good published site showed no link at all on the page anybody actually lands on. `sync-repo` cannot do it - it reads and writes local files, and this lives on the host.

Confirmed with `gh` where that is installed, since it already holds a token and bootstrap does not have to. There is no tokenless way to ask: the Pages API answers `404` to an anonymous caller even for a *public* repository with Pages enabled, and the published site cannot be fetched until a push has built it. Without `gh` the stage says it could not look, rather than guessing - the site check at the end of the run is the honest test either way.

- **Added: a stage** that makes the first commit and pushes it (#339).

  Everything before it left a working project on one machine and an empty repository on the host - and a reader trying to finish the job from VS Code's "Publish Branch" could not, because that creates a repository and theirs already existed.

  It runs after the local setup stages, so the first commit carries the CSL style, the MathJax bundle and the VS Code settings rather than needing a second commit for them, and before the site check, because the push is what builds the site.

  It waits on the clone and on `origin`: committing into a directory that is not a repository, or pushing to a remote that was never set, cannot be a plan worth running. - **Added:** `pdk`, the same tool under a shorter name.

  Every command here is typed at a prompt, often several times over while a setup is being repaired - `pdk bootstrap --apply` rather than `prodockit bootstrap --apply`. Both names stay, so anything written against `prodockit` keeps working, and the help text says whichever one was typed.

1.22 0.28.1 (2026-08-12)

- **Fixed:** the "your own project" stage could loop for ever, and still asked for a Pages step the workflow now does itself (#336).

  The repository existed, the reader said so, and was told the clone still points at the template - a fact about their machine that creating a repository cannot change. Blocking that stage on the history reset was wrong: the repository lives on the host, and `rm -rf .git` is local, so nothing about creating it is undone by the reset. Only the repoint is, and only that stage is blocked now.

A blocked stage also ends the confirm loop with `waiting` rather than asking again, since no answer can satisfy a stage waiting on another one.

The Settings > Pages instruction is gone. The template's workflow enables Pages itself, so it described work already done. Stage 19 still checks the published site answers - what has gone is the instruction, not the verification.

- **Changed:** the configure questions are numbered, and their text wraps to the terminal instead of at line breaks typed in by hand - which only lined up for one length of project name.

1.23 0.28.0 (2026-08-12)

- **Added: a nineteenth stage** that checks the documentation site is actually published (#333).

  A test rather than a step. The template's workflow enables Pages itself now, so there is nothing for a reader to switch on - and an instruction telling them to anyway is one more thing to rush past, which is how two projects reached a red first build.

  Last of all, because it can only be true once a push has built the site. Fetched anonymously: a Pages site answers to anyone even when the repository behind it is private, which is what makes it checkable without a token at all.

  The `ok` message says the site is **public**, because that is the part readers get wrong: a private repository does not make a private site - only a GitHub Enterprise plan can restrict who reads one - so anything in `docs/` is readable from the moment it builds.

  A host that publishes at no address bootstrap can work out - a self-hosted GitLab - reports `not checked` rather than claiming a site was found.

- **Changed:** where the project comes from is decided at `--configure` time, as a question naming every path (#332).

```
buckwem/report-windows-v1 already exists on github.com and has
content in it.
Do you want to:

1. clone the full repo 'buckwem/report-windows-v1', then leave
the existing
   git records and sync origin unchanged
2. clone the full repo 'buckwem/report-windows-v1', then delete
the existing
   git records and set up a new remote repo
3. start from the template instead, discarding nothing on the
host -
   but a first push would then replace what is in buckwem/
```

```
report-windows-v1

Select 1, 2 or 3:
```

Both of the first two clone the repository itself: the difference is what becomes of its history and its remote, which is the only part there is to decide. "Existing project or template" framed that wrongly - somebody starting again still wants the contents that are already there.

The template is named rather than implied. It is what happens when nothing else is chosen, and a reader who cannot see it among the options has to infer it from the absence of anything else.

A repository that is empty, or not there at all, is a *message* rather than a question: cloning an empty one would leave no `zensical.toml`, no `requirements.txt` and no `tools/`, so every later stage would fail on the absence. The permissions an issued repository carries are not lost by that - they belong to the repository on the host, and `origin` is pointed at it either way, which the message says so it does not read as the repository being ignored.

No default. One answer deletes commits that cannot be recovered, and none of them is safe enough to be taken by pressing Enter.

The answer is recorded, so the run follows an explicit path rather than re-deriving the decision from what `origin` happens to say - and a rerun does not ask again.

- **Fixed:** the history stage offered to delete a project's real history because `core.fileMode` was unset, and adopting an existing repository happened silently (#332).

  On a clone that already carried its own history, the stage reported wrong only because `core.fileMode` was off - and its plan was still `rm -rf .git`. Its plan is now that one setting and nothing else, and is not marked destructive, because nothing there destroys anything.

  Adopting an existing project is put to the reader, saying what each answer means: cloning brings their work and its history, and the history stage will not offer to delete it; answering no takes the template, whose history that stage then deletes with `git init -b main`.

  The report says which repository was used - `from the template` or `your own project` - so the decision is still visible once the prompt has scrolled away.

- **Fixed:** answering no to the history reset showed the commands anyway (#330).

  ```
  Delete the template's history and start a new repository? [Y/n]: n
  ```

printed `rm -rf .../.git` and asked again. The answer was collected and discarded - written as a bare acknowledgement, which is fine for "have you uploaded the key?" and not for a deletion with no undo.

That prompt also defaulted to yes on a destructive plan. The rule that this one plan must not default to yes had been applied to the command prompt below it, and not to the question a reader reads first.

The same stage reported `no clone yet` and then offered to delete a `.git` that did not exist, and run `git init` in a directory that did not either. It waits for the clone stage now, the way the two stages after it already do.

1.24 0.27.0 (2026-08-12)

- **Fixed:** a second machine cloned the template over a project that already existed (#327).

  Set up on one machine, pushed, then run on another, bootstrap cloned the *template* - giving the reader template content in a checkout whose `origin` the next stages repointed at their real work. Nothing errored; every stage reported done.

  The project on the host is cloned instead when it already holds commits. A project that exists but is empty - created in the browser, never pushed to - still gets the template, since cloning it would leave nothing to work on. `git ls-remote` tells the two apart on evidence rather than a flag.

  This is also the case where a taught module hands a student a repository that already holds their starting point. Deleting its history would throw that away, so the history-reset stage is not offered for a clone of the reader's own project - now asserted rather than left to fall out of how `origin` happens to compare.

  An explicit `source_url` still wins: detection is a default, not an override.

- **Added: a Documentation badge**, and dropped two that cannot work on a private repository (#326).

  `prodockit sync-repo` kept `site_url` correct in the config while the README - the page a human actually lands on - had no way through to the published site. It leads the badge row now, reporting whether the site is up where shields.io can reach it and linking plainly where it cannot.

  The star and fork badges rendered `Stars: repo not found` on a private repository, which is what `prodockit bootstrap` tells readers to create - so two of three badges were wrong by default. They are emitted only when an anonymous visitor can see the repository, which is exactly the view shields.io has. A visibility check that cannot be answered - offline, a timeout - changes nothing and says so.

- **Fixed:** nothing enabled GitHub Pages, so the first push failed in CI (#324).

  Bootstrap reported every stage done, the initial commit and push both succeeded, and the Documentation workflow then failed with `Get Pages site failed` - which names the site rather than the setting nobody had been asked to switch on.

  Two GitHub-only problems, both now said while the reader is still in the browser rather than left to be found from a failed build:

  - Pages has to be enabled by hand, under Settings > Pages, with Source set to 'GitHub Actions'. GitLab needs no equivalent - its CI job configures its own.
  - The repository stays private but the site built from it does not, which is what GitHub itself warns when Pages is switched on. Said here because drafts and notes in `docs/` are published the moment they build, not when their author decides they are ready.

  Both are `Host` fields rather than branches in the stage, following the rule that a difference between hosts is a value.

1.25 0.26.7 (2026-08-12)

- **Fixed:** the two browser stages could not see what you had just done in the browser (#321).

  A repository created on the host was reported as missing, and "Try again?" repeated the same answer however many times it was accepted:

  ```
  not there yet - git@github.com:you/report.git is not reachable
  Try again? [Y/n]:
  ```

  A regression from the connection-reuse work in 0.26.4. Answers about the host are remembered within a pass and dropped after each applied command - but a browser stage's plan is instructions only, so no command ran, nothing was dropped, and the verification read an answer from before the reader went to the browser.

  Anything remembered is dropped now before every re-check that follows a human step, in both the verification and the retry loop. The saving those memos exist for is unaffected: it comes from repeats within a single pass, none of which have a person acting in between.

1.26 0.26.6 (2026-08-12)

- **Fixed:** creating the SSH key failed on a machine with no `~/.ssh` directory (#318).

`ssh-keygen` does not create the directory it is asked to write into, and the error names the *key* rather than the missing folder:

```
Saving key "C:\Users\you\.ssh\id_ed25519_github" failed:
No such file or directory
```

The stage creates it first now, on all three platforms - Windows is where it bites, since macOS and Linux usually have `~/.ssh` already from some earlier ssh use, but a genuinely fresh machine of any kind has no such directory.

Only when it is absent, so an existing directory keeps whatever permissions its owner chose. One this created gets `700` on macOS/Linux: ssh refuses a key others can read, and the same applies to the directory holding it.

- **Fixed:** accepting a host's key stopped the run, even though the connection had just succeeded (#316).

  `ssh -T` against a git host exits non-zero *even when it works* - there is no shell to give you, so the exit code says nothing at all. The checks have always known this and match on the greeting instead, but the apply loop was still reading the code:

  ```
  Hi buckwem! You've successfully authenticated...
  failed: exit status 1 - see the output above
  ```

  That probe is never fatal now. A genuine rejection is still caught, by the stage's own check re-running it and reading the greeting - which is the one thing that can tell the two apart.

1.27 0.26.5 (2026-08-12)

- **Fixed:** on Windows, a rerun stopped at "Git install failed" when git was already installed (#309).

  winget reports "already installed, no upgrade available" as exit `2316632107` (`0x8A15002B`), which bootstrap read as a failed install. It stopped there, so the `git config --global user.name` and `user.email` behind it in the same plan never ran - leaving git installed and unconfigured, with the run blaming the install.

  That code from winget is treated as "nothing needed doing" now, and the rest of the plan runs. Deliberately narrow: it matches on the program as well as the code, so the same number from anything else is still a failure, and only codes actually observed to mean "already done" are listed - guessing at more would risk swallowing a real one. - **Fixed:** two stages acted before the history reset, and a run could finish with `origin` still pointing at the template (#311).

`rm -rf .git` deletes every remote. A reader who declined the reset, let the remote stage repoint `origin` and run `sync-repo`, then changed their mind and reset, silently lost the repoint - ending with a fresh repository, no origin, and a `sync-repo` that had run against a state no longer there.

Leaving `origin` at the template was worse. For a student it fails at the first `push`; for anyone with write access to the template it pushes their own work into it, and the template is public and cloned by every new reader.

Both stages now report `blocked` while the clone still points at the template - counting as work outstanding, so nothing reads as finished, but building no plan, so no command runs that the reset would undo. A clone made from `source_url` is unaffected: its origin is the reader's own, so there is nothing to wait for.

- **Added: github.com as a supported host.** Bootstrap could only be pointed at `gitlab.surrey.ac.uk`, so when that server stopped answering there was no way to run or test any of it.

  The `Host` record already carried everything github.com needed - its own greeting string (`ssh -T` exits non-zero on success against both, so the greeting is the only signal), its own key-form labels, and `repository` / `organisation` in place of `project` / `group`. Turning it on was enabling the record and widening the hostname check, not rewriting any stage.

  Only Surrey clones from Surrey: it mirrors the template onto its own GitLab so a student never needs a GitHub account. Every other host clones the GitHub original.

  gitlab.com stays unsupported, and a self-hosted instance of either family still gets the "not supported yet" answer - naming a family is not the same as having been run against that server.

1.28 0.26.4 (2026-08-11)

- **Fixed:** the SSH key was loaded into the agent for one session only, so a machine bootstrap had set up stopped authenticating after the next reboot (#303).

  `ssh-add` loaded the key into the running agent and nowhere else, and the `Host` stanza named the key with `IdentityFile` but had nothing that would reload it. The stage reported itself `OK`, quite correctly at the time, and the machine then broke silently days later.

  The failure does not name its own cause: SSH offers the *public* half of the key without needing a passphrase and only fails at the signing step, so the error looks like a key the host has rejected. It is the same trap as #246, one layer further out.

`ssh-add --apple-use-keychain` is used on macOS now, and the stanza carries `AddKeysToAgent yes` on every platform plus `UseKeychain yes` on macOS. The config check reports a stanza missing them rather than passing it, since a setup that works only until the next reboot is not a finished one.

`UseKeychain` is written on macOS alone, deliberately: an OpenSSH that does not know the keyword rejects the whole config file rather than skipping the line, which would take every other host in it down too. - **Fixed:** bootstrap logged in to the host far more often than it needed to, and a host that stops answering was reported as a rejected key (#304).

Every pass ran `ssh -T` for the check and again to decide whether the plan needed the terminal, plus a `git ls-remote` - and the `--apply` loop re-derives a plan and re-checks after every stage. A single run made dozens of logins within seconds, which is what provokes a server into refusing.

Repeats within a pass are now answered from the first connection. The memo is dropped after any applied command, so the verification re-check - the one claim bootstrap makes that is worth anything - always connects for itself.

A run now also reports how many connections it made, which is the measurement any later throttling should be built on rather than guessed at.

Separately, a connection the host accepts and then closes is reported as what it is, instead of as `could not confirm authentication` or a key the host rejected. `Permission denied` still reports as a rejection: that is a clean answer from a working server, and telling the reader to wait would be wrong.

1.29 0.26.3 (2026-08-11)

- **Fixed:** Windows stopped after installing git, saying git was not installed (#300). winget reported `Successfully installed`, and the `git config` on the next line failed with `git: not found`.

  A Windows installer adds itself to `PATH` by writing the registry and broadcasting a change. A process that is already running never sees it - its environment was copied when it started - so any stage that installs a tool and then uses it failed on a machine where the install had just succeeded.

  `PATH` is re-read from the registry between commands now, which is what opening a new terminal does. It is a different fault from #292 and #295, which were about `.cmd` shims and `PATHEXT`; this one affects `git.exe` and every other real executable too.

1.30 0.26.2 (2026-08-11)

- **Added** `prodockit init-mathjax`, and with it one implementation of

something that had two (#276). The MathJax configuration was written by bootstrap's stage 18 *and* by a template's CI, which never runs bootstrap - two copies of a file whose whole failure mode is being subtly wrong, since both produce valid JavaScript and the site simply typesets one way locally and another when published.

The delimiters are why it mattered: `inlineMath: [["\\(", "\\)"]]` carries four layers of escaping, and a copy that looks right can be wrong.

Stage 18 now runs the command, the same way the repoint stage runs `prodockit sync-repo`, and anything that builds a site without bootstrap can run it too.

- **The Ubuntu VS Code install names your architecture instead of working it out in a shell** (#287). The command carried `case "$arch" in amd64) arch=x64 ;; esac`, which reads as a hardcoded target even though it only maps dpkg's name onto VS Code's. It behaved correctly on arm64; it just could not be trusted at a glance, and a reader is being asked to approve it.

  It is resolved when the plan is built, so the command shows `linux-deb-arm64/stable` on an arm64 machine and `linux-deb-x64` on an amd64 one.

  The download and the install are two commands now rather than one shell line. The download needs no privileges and the install does, so splitting them keeps `sudo` at the front of a command where it can be seen - and where a credential timestamp expiring mid-run prompts visibly, rather than inside a shell whose output nobody is watching, which is how that stage reached its 30-minute timeout.

- **Fixed:** three more Windows commands that could not have worked (#295), found by reading the plans after #292 rather than by reaching them.

  `npm` would not have been found at all. On Windows it is `npm.cmd`, and Python's `subprocess` uses `CreateProcess`, which does not apply `PATHEXT` - so a bare `npm` reports "not found" on a machine where Node is installed correctly, and neither render toolchain installs. It is resolved by path now, exactly as VS Code's CLI is. That is also *why* #292 happened: `code` is `code.cmd` for the same reason.

  MSYS2 was assumed to be at `C:\msys64` - winget's default, not a guarantee. A machine with it elsewhere got "file not found" about a path bootstrap invented, and what breaks is the PDF build much later, since Pango is what WeasyPrint draws text through. It now says where it looked and what to do.

  The citation style download turns PowerShell's progress bar off. On PowerShell 5.1, still the Windows default, `Invoke-WebRequest`'s progress rendering makes a download dramatically slower - which reads as a hang.

- **Fixed:** Windows reported VS Code as broken immediately after installing it

(#292). The installer adds `code` to `PATH` itself - but `PATH` is read when a process starts, so the shell that has just run `winget install` cannot see it:

```
ran, but still not right: VS Code is installed, but the `code`
command is not on PATH
```

and the advice offered was a Command Palette action that exists on macOS and not on Windows.

The executable is looked for where the installer puts it now, and used by its full path - so the extensions stage works in the same session too, rather than the reader being sent to open a new terminal and start again. macOS is untouched: there the application really is installed without the command, and that Command Palette action really is how it is added.

- **Fixed:** a long install looked like a hang (#244). Applying captured every command's output, so `sudo apt update`, a 100 MB download and `apt install` behind it produced minutes of silence after `These need administrator rights.` - and a silent terminal is indistinguishable from a hung one. Installs that were working were interrupted.

  Installers now write to the terminal as they go. The re-check that follows is still captured - it *reads* what a command printed, and there are dozens per run - so both ends of it are announced, since a silent check straight after a visibly finished command reads as a hang of its own.

- **`--apply` shows every stage, numbered by where it actually is** (#284). Stages already set up were skipped in silence, and the ones remaining were numbered by their position in the queue - so `[1/17] Git` appeared while standing at stage 2 of eighteen, agreeing with nothing the reader could check against `prodockit bootstrap`'s own listing.

  ```
   1  ok    Visual Studio Code
   2  ok    Git, installed and configured
   3  ok    SSH keypair
   4  ok    SSH config points at the key

  [5/18] Key loaded into the ssh agent
         id_ed25519_gitlab is not loaded into the agent
  ```

  A stage waiting on a configuration answer is named too, rather than vanishing from the run.

- **Documentation:** the Windows prerequisites now set PowerShell's execution policy (#288). Windows blocks all scripts by default and activating a virtual environment *is* a script, so the step immediately after failed:

```
.\.venv\Scripts\Activate.ps1 : File ...\Activate.ps1 cannot be
loaded
because running scripts is disabled on this system.
```

which names the script rather than the policy blocking it, and so reads as a broken file. `Set-ExecutionPolicy -Scope CurrentUser -ExecutionPolicy RemoteSigned` comes first now, with what it does and does not change, and the CMD alternative for anyone who would rather not change it. The "activate it again" section points back at it.

1.31 0.26.1 (2026-08-11)

- **Fixed:** naming an existing repository to clone did not work (#283). The prompt asks for "an existing repository to clone instead of the template", and a repository is called `report-az1234` - not `git@gitlab.surrey.ac.uk:comm058-2026/report-az1234.git`. That answer reached `git clone` verbatim:

  ```
  Will run:
    git clone report-mb0105-v13 /Users/…/report-mb0105-v13
  failed: fatal: repository 'report-mb0105-v13' does not exist
  ```

  which reads as though the repository were missing rather than the address incomplete. Three forms are accepted now - a full URL used exactly as given, `group/name`, or just `name`, expanded against the configured host and namespace. The prompt says so.

- **Fixed:** a stage's instructions could describe the machine as it was when the run *started*, not as it is when the step is reached (#281).

  `--apply` built every plan up front, before applying anything. The SSH upload step is where it showed: it embeds your public key, and on a fresh machine the keypair stage has not run when the plans are built - so it fell back to "paste the contents of `~/.ssh/id_ed25519_gitlab.pub`" about a key that existed perfectly well by the time the reader got there.

  Each plan is now built when its stage is reached. The commands were never affected - applying a stage already re-derived its plan - so this only ever changed what was shown, which is precisely what made it hard to spot.

- **Fixed:** a first run reached configuration without ever being asked which git host to use (#279). `prodockit bootstrap` offers to fill in what is missing, and `host` has a default - so it is never *empty*, never reported missing, and was never asked on that route. Only `--configure` asked it, which is not the path a first-time reader takes.

  It is now asked on a first run - when there is no configuration file yet - and

still not re-asked of somebody who already answered it, since that is what `--configure` is for. The scripted message lists it too.

- **Accepting a host's fingerprint no longer means opening another terminal.** The SSH upload stage used to end with "run `ssh -T git@gitlab.surrey.ac.uk` in a terminal once and answer `yes`" - a step in the middle of a guided run that sends the reader somewhere else to take it.

  Bootstrap offers to run it now, with the terminal handed over, so ssh shows its own fingerprint and asks its own question in place. What has not changed is who answers: trusting a host key is still the reader's decision, and nothing answers it on their behalf.

  `BatchMode=yes` is dropped for that one command only. It is what makes a *check* safe - a check that can block is a broken check whatever it reports - and it is also exactly what suppresses ssh's fingerprint question, so it comes off deliberately and only after the reader has agreed to connect. `ConnectTimeout` stays, so an unreachable host still fails rather than hanging.

1.32 0.26.0 (2026-08-11)

- **Fixed:** the website showed raw TeX where an equation should be (#263). MathJax was loaded with no configuration at all, so `pymdownx.arithmatex`'s markup was emitted and never typeset. The configuration has to load *first* - MathJax reads `window.MathJax` once at startup, and one that arrives afterwards is ignored.

  A new **stage 18** writes that configuration and installs the browser bundle by copying it out of `tools/mathjax`'s own pinned install - the very one `prodockit pdf` pre-renders through, so a formula cannot typeset one way on screen and another in print.

  Installed, not committed. The bundle is somebody else's code and does not belong in a project's repository, so it is added to `.gitignore` alongside `tools/*/node_modules`, which it comes from. Nothing is fetched from a CDN, so the site typesets offline.

- **The source bundle's footer names the repository it came from** (#262). A bundle is a thing people hand in, and the reader of one could not tell which repository the source belonged to. The git remote now sits on the left of the footer, opposite the page number.

  The remote rather than the directory, because two checkouts of the same project have different local paths and the same remote - and a path on somebody else's machine identifies nothing. A directory with no remote falls back to its absolute path, which at least says which copy. Any credentials embedded in the remote URL are stripped: a bundle is submitted, printed and emailed.

- **Documentation:** the "activate it again" instructions now change directory first (#264). A new terminal starts in your home directory, and `source .venv/bin/activate` is a relative path - so from anywhere else it is simply not there, and the error says the file does not exist rather than that you are in the wrong place.

- **The email prompt names the host rather than a university** (#265). "Your university email address" is wrong for anybody outside a university, and wrong inside one for a reader whose GitLab login is not their university address. It now reads `The email address used for your gitlab.surrey.ac.uk login`, taking the host from the answer given a moment earlier - which is what asking the host first is for.

- **`--apply` no longer prints a Python script at the reader** (#261). Stage 16 carried an entire script as one argument, so the prompt filled the screen with source and asked for approval of it. It says what it does instead:

  ```
    Will do:
      Update ~/GitLab/report-x/.vscode/settings.json so Markdown
  opens in
      Zensical Studio's editor, and LTeX+ checks your writing as
  en-GB
  ```

  Any command carrying a script in an argument is collapsed rather than printed whole, so this cannot come back through another stage. `--dry-run` still prints commands exactly, which is what `--dry-run` is for.

- **Each manual step now asks about the thing it asked for** (#260). Every one of them ended in `Tell me when that is done` - including the step whose entire content is "this deletes your history and cannot be undone", where nothing has been asked of the reader at all and the honest question is whether to go ahead:

  ```
  Delete the template's history and start a new repository? [y/N]:
  Have you added the key to your gitlab.surrey.ac.uk account? [Y/
  n]:
  Have you created the project on gitlab.surrey.ac.uk? [Y/n]:
  Have you run the 'Shell Command' action in VS Code? [Y/n]:
  ```

  A test walks every stage on all three platforms and fails any that still asks the generic question.

- **Every `--apply` prompt defaults to yes, except the one that cannot be undone** (#259).

  The default used to follow the check's status - `MISS` meant yes, `WRONG` meant no - which is a rule a reader cannot see from the prompt, so the same

key press meant different things at different stages for reasons that were never on screen.

A plan now declares whether applying it destroys something, and exactly one does: resetting the template's commit history. Pressing Enter through a run installs things and never deletes a repository's history.

- **`--apply` now says what it is doing before it starts** (#258). It opened straight into `[1/11] Visual Studio Code`, which tells a reader which step they are on and nothing about what they have started or where it will land. It now names the version, the host, the project directory and how many stages need work, and says to run it from the project directory with the virtual environment active.

- **The SSH key form is filled in the order it actually works in** (#257). GitLab fills the Title in from the key's own comment the moment a key is pasted, so a title typed first was silently replaced - and a reader following the steps in order ended up with a list of keys all named after their email address. Key first, then Title, with the suggestion to replace the address GitLab puts there with this machine's name, which is what a key title is for.

- **The git host is now the first configuration question** (#255). Everything else is shaped by it - which URLs the browser steps send you to, which key file is looked for, whether you are creating a project or a repository - and it was not asked at all, only defaulted.

  An unusable answer is refused at the prompt and asked again, rather than stored and refused by the run five questions later, which is the whole value of asking first. The prompt and `build_context` decide through the same function, so one cannot accept what the other rejects.

  It is asked as a **hostname** - `gitlab.surrey.ac.uk`, the thing in the address bar - rather than a nickname, and judged three ways before it is stored: a host that is not a GitLab is refused, one that is not supported yet says so, and one that does not answer on port 22 is reported as unreachable.

  That last check is worth the second it costs. Without it, the first sign of an unreachable host is stage 6 reporting a rejected key - after a key has been made and pasted into a web page - and "I cannot reach this server" looks nothing like "this server refused you", which is a confusion these stages have produced three times. Re-asking is a real retry: connect the VPN, press Enter, and the second attempt succeeds.

  Configurations written before this stored `host = "surrey"`, and still resolve. Pressing Enter still gives Surrey's GitLab, the only host implemented today.

1.33 0.25.0 (2026-08-11)

- **Windows is automated end to end** (#217 phase 4). All seventeen stages now

produce commands or instructions there. MSYS2 and Pango - which WeasyPrint draws text through, and which the User Guide walks the reader through a MINGW64 shell and the Environment Variables dialog to install - are installed unattended, with the `PATH` entry added only when absent.

None of it has been run on a Windows machine, and those two facts belong in the same sentence. The evidence is that the command lists match what the guide prescribes, asserted from macOS.

- **Fixed:** every `winget install` could stop for a human. winget asks for agreement to its source terms on first use, and to a package's own terms when it carries them - on the terminal, so a captured, timed subprocess waits. That is the `sudo` failure of #243 reached by another route, and it would have met every Windows reader at stage 1. All of them now pass `--accept-source-agreements`, `--accept-package-agreements` and `-e`.

- The two steps Windows genuinely cannot automate - the `ssh-agent` service, which needs an Administrator window, and the PDF fonts, which Windows has no package manager for - are now **checked** rather than merely suggested.

- **Fixed:** five stages installed things nothing then checked (#224).

  The pattern behind four earlier bugs - a stage's check narrower than its own plan - had produced three more, all introduced in the two days before this. On a machine with node and pandoc present and nothing else, both stages reported `ok` while their plans would have installed Chromium, written shell exports, run `npm ci` for two toolchains, and installed two fonts. A reader who had installed Node themselves was told the stage was done, got no toolchains, and found out at the first diagram.

  The node check now verifies both toolchains and, on Ubuntu, that Puppeteer has a system Chromium *and* is pointed at it. The pandoc check verifies the PDF fonts - and says nothing when the machine cannot be asked, since a false alarm sends the reader to reinstall fonts they already have. The history check verifies `core.fileMode`.

- **Added:** a test that holds across every stage, so the next one inherits it rather than being audited by hand. Two gates: each stage must declare what its plan produces, and no stage whose plan installs something may report `ok` about a machine where nothing is installed. Both were confirmed to fire on a deliberately careless new stage.

- **Carried the User Guide's three ARM64 findings into bootstrap** (#249, from prodockit-userguide#104). All three were found on the same fresh Ubuntu ARM64 machine, and all three fail in a way that does not point at itself.

- **Fixed:** `npm ci` fetched a Chrome it could not run. Installing the Mermaid toolchain triggers Puppeteer's own postinstall download, and that download is not guaranteed to match the CPU it lands on - on ARM64 it fetches an

x86_64 build. Nothing fails at install time; the symptom is a diagram that will not render, much later, with nothing to connect it to the install. Ubuntu now installs a system Chromium and exports `PUPPETEER_EXECUTABLE_PATH` and `PUPPETEER_SKIP_DOWNLOAD` *before* `npm ci`, and appends them to `~/.bashrc` once - checked first, so a rerun does not leave four copies. macOS and Windows are untouched, where Puppeteer's own download is fine.

- **Fixed:** the PDF's fonts were never installed. The website loads Inter and JetBrains Mono from a CDN when a page is viewed, but a PDF has to embed the files - and WeasyPrint substitutes a fallback **silently** rather than failing. The build succeeds, the PDF looks plausible, and the only symptom is a test reporting `No 'Inter' font found`. They are installed with the graphics stack now, by cask on macOS, apt on Ubuntu, and as an instruction on Windows, which has neither.

- **Added:** the citation style the first build needs. `prodockit.bibliography` is enabled by default and points `csl_style` at `harvard-cite-them-right.csl`, which is fetched rather than committed - so `zensical serve`, `zensical build` and `prodockit pdf` all failed outright on a fresh clone. An empty file counts as `WRONG` rather than done, since a failed download leaves one behind; and a project configured for a different style is told where to find its own rather than given Harvard.

- **Bootstrap now leaves a machine you can start writing on.** Comparing it against the User Guide step by step - prompted by `ssh-add` turning out to be missing entirely in #246 - found six things it did not do at all (#248). Three new stages, and the count goes from thirteen to sixteen.

- **Fixed:** WeasyPrint was never verified, though a stage was named for it. Stage 12 read "Pandoc and WeasyPrint's libraries" and ran `pandoc --version` and nothing else, so it reported `ok` on a machine whose first PDF build would fail at `cannot load library`. Importing WeasyPrint is the test now, and a strict one: the import loads Pango through the system linker, so success proves both the Python package and the native libraries. `pip` exiting zero proves neither. The pandoc stage is renamed to what it actually checks.

- **Fixed:** the VS Code extension list disagreed with the guide. Even Better TOML and LTeX+ - both required - were missing, while Code Spell Checker, which comes from the *optional* tooling page, was installed in their place. Marketplace identifiers were checked rather than guessed: the obvious `valentjn.vscode-ltex` returns 404, and the maintained fork is published under `ltex-plus`.

- **Added:** the project's own virtual environment, and its dependencies. Bootstrap cloned a template shipping a `requirements.txt` and never installed it, so Zensical itself was absent from the project. The new stage creates `<project>/.venv` and installs into it by naming that interpreter

explicitly - a bare `pip install` would find bootstrap's own pip, install the project's dependencies into bootstrap's environment, exit zero, and leave the project's `.venv` empty.

- **Added:** a history of your own. The guide resets the template's commit history; bootstrap carried its whole log and branches into every project. This is the only stage that destroys anything, so it reports `WRONG` rather than `MISSING` - deleting history should not happen by pressing Enter - and is judged by whether `origin` still points at the template, never by whether commits exist. The latter would tell somebody who had been writing for a month that their history needed deleting.

- **Added:** the editor's settings for the project. Markdown is associated with Zensical Studio's language mode, and LTeX+ is set to **the language the machine is in** rather than the guide's `en-GB` - bootstrap runs on other people's computers, and a document checked against the wrong variety of a language is worse than one not checked, because the corrections are confident and wrong. When the locale cannot be read the setting is left out rather than guessed. Existing settings are merged, never overwritten.

- **Fixed:** `git remote set-url` failed on a repository `git init` had just created, which is exactly what the new history stage leaves behind. The repoint now adds the remote when there is none.

- **Fixed:** a passphrase-protected key could never pass the SSH stage, because nothing ever loaded it into an agent (#246).

  Stage 3 tells the reader to set a passphrase, and every ssh command bootstrap runs carries `BatchMode=yes`, which forbids prompting for one. Those two are only compatible if an agent holds the decrypted key - and `ssh-add` appeared nowhere in the code.

  The failure is a quiet one. `ssh -T` reads the `.pub` file and offers the public half without needing a passphrase; the host then challenges it to sign, that needs the private half, and there is nobody to ask. Authentication fails, and the upload stage reports `the host rejected the key` about a key that is correct, uploaded, and simply locked - the third failure in this area to lie about its own cause, after #234 and #239.

  A new **stage 5** checks the key's fingerprint against `ssh-add -l` and runs `ssh-add` when it is absent, with the terminal handed over so the passphrase prompt has somewhere to appear. It is the only command in bootstrap that gets the terminal.

  Starting an agent is not automatable and is not attempted: `eval "$(ssh-agent -s)"` exports `SSH_AUTH_SOCK` into the shell that runs it, and a subprocess cannot export into its parent - so bootstrap would start an agent, set the variable in a shell that then exits, and change nothing. When none is running it says so and gives the line to run. On Windows the agent is a service, and enabling it needs an Administrator window.

- Bootstrap now reports **thirteen** stages, and the numbers after the new stage 5 shift by one.

- **Fixed:** installing the VS Code extensions defaulted to *no* (#242). With none of the three installed the prompt read `Run 3 commands? [y/N]`, so pressing Enter declined the install the reader had run bootstrap to get. The prompt's default follows the stage's state, and `WRONG` defaults to no because reapplying over something can destroy work - but nothing installed is `MISSING`, and there is nothing there to destroy. A partly-installed set stays `WRONG`, and still asks.

- **Fixed:** a slow install was killed and reported as failed while it went on to succeed (#243).

  Two causes, and both needed fixing. `sudo` reads its password from `/dev/tty` exactly as `ssh` does, so it asked from inside a captured subprocess - the reader typed a password into a command whose output was being swallowed, and their thinking time counted against the clock. `sudo -v` now runs first, with the terminal attached, purely to refresh the credential; plans needing no privileges are never asked.

  And an install is now allowed 30 minutes where a check gets 5. VS Code's `.deb` is around 100 MB, and a download plus `apt install` on a virtual machine can legitimately run past five minutes - which printed `failed` next to a VS Code that was, by then, installed. A timeout is also reported in the reader's terms now, rather than as the raw command repr with the useful sentence at the end of it.

- **Fixed:** `apt` gave up instead of waiting for the dpkg lock (#244). On a freshly installed Ubuntu the process holding it is usually `unattended-upgrades`, which starts on boot and can hold it for minutes, so a first run met `Unable to acquire the dpkg frontend lock` - which reads as a broken machine rather than "something else is mid-update". Every apt call bootstrap makes now passes `DPkg::Lock::Timeout`, so apt waits, which is what a human would have done.

- **Fixed:** bootstrap never wrote `~/.ssh/config`, so ssh had no way to know which key belonged to the host and asked for a password instead (#239).

  Without a `Host` stanza, ssh offers its own defaults - `id_rsa`, `id_ed25519` - never tries `id_ed25519_gitlab`, and falls back to `git@gitlab.surrey.ac.uk's password:`. That is indistinguishable from a key the host has rejected, so the reader goes back and re-pastes a key that was never the problem, because it was never offered.

  A new **stage 4** writes the stanza in the User Guide's own shape, before the upload stage that depends on it. It is appended, never written over - an ssh config is the reader's own file - and a stanza that already exists pointing elsewhere is explained rather than edited, since ssh takes the first match and

rewriting somebody's ssh config underneath them is not an installer's business.

The same stage sets `chmod 600` on the key and the config. ssh ignores a private key others can read (`Permissions 0644 ... are too open. This private key will be ignored`) and then falls back to a password - the same symptom from a different cause. Windows has no `chmod`, and restricts a profile file to its owner already.

- Bootstrap now reports **twelve** stages rather than eleven, and the later stage numbers shift by one.
- The SSH upload stage now **prints the public key** between `======= PUBLIC KEY =======` markers, rather than naming the file it lives in (#238).

  "Paste the contents of `~/.ssh/id_ed25519_gitlab.pub`" asked a first-time reader to find a dotfile, open it in something, and copy the right one of two files whose names differ by four characters - where picking the wrong one uploads the *private* key. Only `.pub` is ever read, and the markers matter as much as the key: it is one long line that wraps in a terminal, and a key pasted a character short is rejected exactly like one never uploaded.
- The same stage now follows the **User Guide's own wording**: the keys page is reached through the host's menus (profile avatar → *Edit profile* → *Access* > *SSH Keys*) with the URL kept as a shortcut rather than as the only way in, and the form's fields are listed under one step instead of numbered as separate errands.
- **GitLab's expiry date is now spelled out** (#238). GitLab requires one, fills it in a year ahead, and will not let you clear it - so a reader who accepts the default is locked out mid-course, and the failure arrives months later as a permission error indistinguishable from a misconfigured key. Bootstrap did not mention the field at all. GitHub has no such field, and says so.

1.34 0.24.1 (2026-08-10)

- **Fixed:** `--apply` could not install VS Code on Ubuntu (#233).

  The plan asked the reader to download a `.deb` from the website and then ran `sudo apt install -y ./code.deb` - a file that exists under that name nowhere. The download is called `code_1.132.0-…_arm64.deb` and it arrives in `~/Downloads`, so the command failed whether or not the reader had done their half.

  VS Code is now downloaded like pandoc already is, from Microsoft's own `linux-deb-$arch/stable` redirect - permanent, so there is no version to pin and let go stale - with `dpkg --print-architecture` choosing between them. The stage is fully automated on Ubuntu now, and nothing is asked of the reader.

- **Fixed:** a failing command could be reported by its *warning* rather than its error (#233). apt prints `WARNING: apt does not have a stable CLI interface` every time it runs from a script, and that was the line shown as the reason the stage failed - describing nothing that went wrong, and pointing at the reader's scripting rather than at the missing file two lines below it. Warnings are now skipped while any other line remains.

- **Fixed:** `--apply` stopped the whole run at the SSH key stage on any machine whose key was not yet on the host - the one state that stage exists to fix (#234).

  Stage 4 carried `ssh -T` as a plan *command*, so 0.24.0's commands-before-instructions ordering ran the probe before saying a word about uploading anything, read its exit code as a failure, and ended the run with `Stopping - later stages depend on this one.`

  The probe could never have passed: `ssh -T` against a git host exits non-zero even on success, which is why the greeting - not the exit code - is what the check reads. Both browser stages are now instructions only, and the re-check that already follows every apply does the verifying.

- A plan's manual steps are now ordered against its commands rather than merely coexisting with them. `instructions` come *before* the commands because the commands depend on them ("Download the .deb", then `apt install ./code.deb`); the new `follow_up` comes *after*, for the step that only makes sense once they have run (VS Code's Command Palette, after the `brew install` that provides it).

  Both orderings had shipped broken - instructions-only skipped the install entirely (#230), and commands-first broke the guide-and-verify stages (#234) - so the order is now stated by each plan instead of being a convention the caller has to guess.

1.35 0.24.0 (2026-08-10)

- **Fixed:** `--apply` skipped the install commands when a stage had both commands and instructions — VS Code on a fresh macOS got only the "open the Command Palette" instruction without the `brew install` that puts the application there in the first place (#230).

- Bootstrap on Ubuntu now downloads the pinned pandoc release from GitHub instead of `apt install pandoc`, which installs a version several major versions behind — far enough to render code blocks as justified prose (#207). The download uses `dpkg --print-architecture` so the same command works on amd64, arm64 and under Rosetta.

- The pandoc check now reports `WRONG` when the installed version is too old (< 3.x), rather than `ok`. Ubuntu's apt package is 2.x on some LTS releases,

and a check that merely asks "is pandoc installed?" would pass on those and leave the reader to discover the problem at their first `prodockit pdf`.

- Ubuntu's git plan now runs `sudo apt update` before the first `apt install`, so a clean machine with an empty package index does not fail to find the package.

- `prodockit bootstrap --help` said "ten stages" a release after there were eleven. The count is now asserted against the stage list, so prose cannot drift from the thing it describes.

1.36 0.23.0 (2026-08-10)

- **Fixed:** bootstrap asked for an email and then never applied it (#222).

  A new stage 8 sets `user.name` and `user.email` on the clone, and checks them with `git config --local`. The old check read them without `--local`, which falls back to the global value - so it passed on any machine with any identity at all, the plan never ran, and commits went out under whatever address git already had.

  On Surrey's GitLab a commit whose author address matches no known account is not linked to one, so coursework can appear to be authored by an unrecognised user - with nothing to suggest why, since every stage reported `ok`.

  Per-repository rather than global: a global `user.email` is a legitimate personal preference, and a tool that sets up one university project should not rewrite the identity used for everything else. Eleven stages now, not ten.

- The bootstrap page now explains how to meet its own prerequisite (#223).

  It named Python as the one thing bootstrap cannot install and then left the reader there - the worst place for a gap, since it is the first thing they hit and the point at which they have no working tooling to fall back on.

  Per-platform instructions for Python and a virtual environment, the `externally-managed-environment` refusal and why a venv is the answer to it, the three things Windows' installer gets wrong (PATH, path length limit, the Microsoft Store placeholder), `python3-venv` being a separate package on Debian, reactivating in a new terminal, and checking `prodockit --version` afterwards - an older install on `PATH` shadows a newer one silently.

- **Fixed:** `prodockit bootstrap` could stop dead at a password prompt (#225).

  On a machine whose SSH key was not yet uploaded, the stage 4 check fell back to password authentication and simply waited - a check that can block is a broken check, and it stopped a test run outright.

  The cause was subtler than it looked: `ssh` reads passwords and

passphrases from `/dev/tty` directly, deliberately bypassing stdin, so the existing `stdin=DEVNULL` never could have prevented it. Every command bootstrap runs now also gets an environment that cannot ask - `BatchMode=yes` for ssh, `GIT_TERMINAL_PROMPT=0` for git, both reaching `git clone` and `git ls-remote` through `GIT_SSH_COMMAND`, which had the same hang waiting in stages 5 and 6. A `GIT_SSH_COMMAND` you have set yourself is left alone.

An unknown host key is now reported, with the one command that fixes it, rather than auto-accepted: trusting a host is a decision that belongs to you, not to an installer.

1.37 0.22.0 (2026-08-10)

- `prodockit bootstrap` - phase 2: configuration and installing the template (#217).

  `--configure` asks each question with the stored answer as its default; `--apply` sets up the stages that need it, asking before each. A stage that is simply absent defaults to yes, one that exists but is wrong defaults to **no** - reapplying over something already there is the case that can destroy work.

  Every stage is re-checked after being applied. A command exiting zero says the installer ran, not that the thing works.

  The template is cloned from the configured host's own copy - Surrey mirrors it onto its own GitLab, so a student there never needs a GitHub account. `source_url` overrides it with an existing repository for a reader who has been given one.

- `prodockit bootstrap` - phase 1: check and plan a full machine install (#217).

  The User Guide's install sequence is long and easy to get half-right in ways that surface much later. This turns it into ten stages that can each be checked and repaired individually.

  Nothing installs anything yet. Phase 1 reports by default - `prodockit bootstrap` with no options checks every stage and changes nothing - with `--dry-run` to print the exact commands a real run would use. Read-only is the default deliberately: the alternative, once applying exists, is a command that starts installing software because somebody typed it to see what it did.

  Two stages are deliberately never automated: uploading an SSH key and creating your own project both need an authenticated human in a browser, and the alternative is a Personal Access Token typed into a tool aimed at first-time students. They guide and then *verify* - `ssh -T` and `git ls-remote` - which is the half a written instruction cannot do.

Surrey's GitLab is the only supported host; `gitlab.com` and `github.com` are declared but refused, so adding them later is filling in a record rather than rewriting the stages.

1.38 0.21.0 (2026-08-10)

- A Zensical rename of `render()`'s result keys now stops the PDF build with a message naming Zensical, the installed version, and the page being rendered - not a bare `KeyError` (#171).

  `zensical.markdown.render.render` is undocumented - `zensical/__init__.py` exports only `build` / `serve` / `version` - so both the function and the shape of what it returns can change in a **patch** release without registering upstream as a breaking change. `result["content"]` read unguarded meant a rename surfaced as `KeyError: 'content'`, raised from inside a loop over nav pages, with nothing naming Zensical, the installed version, or the upgrade that caused it. The reader sees prodockit's own traceback and reasonably concludes prodockit is broken.

  Deliberately not #167's warn-and-degrade shape: there is no sensible degraded PDF to produce, and a page silently rendered with empty HTML would be exactly the kind of build-succeeds-output-broken failure this project keeps landing on. The build still stops - it now just says why. `TypeError` is caught alongside `KeyError`: if `render()` starts returning an object rather than a dict, that is what a subscript raises instead, and the diagnosis is identical.

- `prodockit source-bundle` is a new command, split out of `prodockit pdf` (#212).

  The two PDFs a project can produce serve different purposes - a rendered document, and a record of what was written - and previously could not be built independently. `pdf_source_bundle = true` made every `prodockit pdf` run also pay for a `git ls-files` scan and a second `weasyprint` invocation regardless of whether anyone wanted the result, and a project that wanted only an updated source bundle still paid for the far more expensive Pandoc/WeasyPrint document pipeline - Mermaid/TeX pre-rendering included - to get it. `pdf_source_bundle` is gone; `prodockit pdf` never builds a source bundle as a side effect now, under any config.

  The default set of files bundled is also narrower: every `.md` file under `docs_dir` plus `zensical.toml`, rather than every text file `.gitignore` doesn't exclude. A project's custom Python extensions, Lua filters, CSS and tests are source code, not documentation, and a report built from a template has no reason to bundle its own tooling alongside the document it produced. The wider, whole-repository bundle - useful for a project doing its own academic-integrity verification of custom code, not just prose - is still available from Python directly (`build_source_bundle()` with no `files`

argument, or `discover_source_files()`'s own result passed explicitly); the CLI command itself has no flag for it, by design.

Output moves from the project's top-level directory into `docs_dir` by default, so Zensical serves it with no separate copy step - `prodockit-template` and `prodockit-userguide` both carried one before this.

- `prodockit pins` now manages pandoc too, matched as a `PANDOC_VERSION` CI variable (#209).

  Pandoc never appears as a pip specifier - it is a build-provided binary, not a Python dependency - so it needed a fourth declaration shape alongside the pip specifier, GitHub runner label and container image tag `pins` already understood: `PANDOC_VERSION: "3.10.1"`, the same in a GitHub `env:` block or a GitLab `variables:` block.

  It joins the default managed set, alongside Zensical and WeasyPrint, for the reason #209 raised directly: three sibling projects were keeping the same `PANDOC_VERSION` in step by hand, across workflow files, with a comment saying "keep in sync" - which is exactly the kind of drift this module exists to catch instead. `prodockit pins --check` in this project's own CI now verifies its three copies agree with no workflow change required.

  A CI variable's name keeps whatever case it was declared in on rewrite - `PANDOC_VERSION`, never `pandoc_VERSION`. Every other managed shape can safely reuse its lower-cased lookup key as the replacement text, because a runner label or image tag is conventionally lower-case already; an environment variable is conventionally not, and using the lookup key there would silently break the workflow step that reads it back. Found and fixed twice over: once in the rewrite itself, and again in the CLI's own progress-line display, which had the same bug independently - caught only by running `--set` against copies of this project's real workflow files rather than trusting the unit tests alone.

1.39 0.20.1 (2026-08-09)

- CI builds with a pinned upstream pandoc instead of the runner image's (#209).

  `ubuntu-24.04` ships pandoc 3.1.3. That is what made the code-block fault below invisible: CI published correct PDFs on the old package while every local build on a current pandoc was wrong, and the next runner-image bump would have broken the published output with no commit to blame.

  `ci.yml`, `docs.yml` and `drift.yml` now install a pinned release from pandoc's own downloads, the way `prodockit-template` already did. `drift.yml` matters as much as the other two: it exists to detect published output drifting, and an unpinned pandoc is a source of drift it could not see.

The documented CI recipe is updated to match, in both its GitHub and GitLab forms, and the limitations page records the whole episode.

- Fenced code blocks keep their line structure in the PDF (#207).

  Reported as stretched word spacing inside code blocks. The spacing was real, but it was a symptom: the blocks had stopped being code blocks at all.

  Pandoc's HTML reader only accepts `<pre><code>` as a code block when that `<code>` holds nothing but text. Zensical's highlighter emits per-token `<span>`s, a `__codelineno` anchor per line, and a leading empty `<span></span>` - each of which, on its own, makes the reader give up and treat the block as ordinary inline content.

  The `<pre>` was then absent from what Pandoc handed WeasyPrint, so `white-space: pre-wrap` had nothing to apply to. Every newline collapsed, the block reflowed and justified like a paragraph, and each token became its own inline `<code>` - which is where the wide gaps came from, and why they varied from line to line. A six-line install snippet came out as four wrapped rows with `".[dev]"` split across two of them.

  Each `<pre>` is now reduced to a single plain-text `<code>` child before Pandoc sees it. Nothing is lost by dropping the token markup: the PDF stylesheet defines no syntax colours, so it was never rendering anything - it was only destroying the block.

  Both a unit test on the HTML handed to Pandoc and a check on the finished PDF, since the intermediate shape being right does not prove the artefact is.

1.40 0.20.0 (2026-08-09)

- `prodockit sync-repo` now keeps `site_url` in step too (#200).

  `site_url` is the address a site is *published* at, and it was the one host-coupled setting nothing managed. Zensical puts it in every page's `<link rel="canonical">` and every `sitemap.xml` entry, so a wrong one tells search engines the real home of the documentation is somewhere else - while the site builds and looks perfect.

  This was not only a hazard when moving hosts. The project template shipped `site_url` pointing at the *repository*, so a fresh clone published a GitHub page as the canonical URL of every documentation page from its first build.

  Only GitHub Pages is derived, since only its shape follows from the remote. GitLab is not guessed at: a self-hosted instance serves Pages from `pages_external_url`, which the remote reveals nothing about, and gitlab.com now issues unique domains rather than the old `<group>.gitlab.io/<project>` path. A confidently wrong canonical URL is

worse than none, so those projects set `pages_base` and the repository name is appended to it.

An existing value is replaced only when it is already a Pages URL, or points at a code host and so cannot be a site. Anything else is a custom domain and is left alone with a note - `--check` is a CI gate, and rewriting a deliberate value would redden every later build for a correct config. An absent `site_url` is not invented.

- `prodockit sync-repo` keeps the whole GitLab namespace, instead of dropping everything between the first group and the project (#201).

  GitLab nests groups, and only the first segment was kept, so `cs-dept/year3/report` became `cs-dept/report` - a project that does not exist. That went into `repo_url`, the edit links and the badges alike, so on an instance where the reader is signed in and the links would otherwise work, they led to a 404 instead. Nested groups are the normal arrangement on university and company instances.

  Silent, as ever: `sync-repo` reported success, `--check` reported being in sync, and the site built.

  The full path now drives every URL. `repo_name` is the exception - it is a header caption rather than a link, so it shows the immediate parent (`year3/report`) to keep the header readable, while the header still links to the correct `repo_url`. A single-segment namespace, which is every GitHub project and most GitLab ones, is unchanged.

  An existing test had asserted the truncated form, with a comment saying it was what the badge and edit URLs needed - the opposite of what they need. That is corrected rather than worked around.

- `prodockit sync-repo` now fills an empty `repo-badges` block, and points GitLab badges at the instance the remote actually names (#198).

  Two faults, which together made the feature look absent rather than broken.

  The block pattern required a newline immediately before the end marker, and the start group had already consumed the only one present when the two markers sit on consecutive lines. That empty pair is precisely how a template ships a badge row for `sync-repo` to fill in - the shape this project's own documentation gives as the example - so the one case that had to work was the one that could not. Worse, the failure to match was then reported as `no repo-badges markers in README.md`, which is what a reader sees when the markers are absent. The markers were there; nothing said so.

  Separately, the badge set is chosen by host *kind*, and `gitlab` matches any instance. The URLs were built with `gitlab.com` hardcoded, so a university or company GitLab got a badge row pointing at a `gitlab.com` project that

does not exist - plausible-looking links to nothing, which is worse than the stale GitHub badges they replaced.

Badge links are now built from the real host. The GitLab build badge is the one the instance serves itself rather than shields.io's, which is also the only version that can work on a private instance: the reader is already authenticated against the host that would refuse an external badge service. Stars and forks have no instance-served equivalent and shields.io reads `gitlab.com` only, so those two are emitted for `gitlab.com` alone rather than as guaranteed broken images everywhere else. - The CI recipe in the documentation matches the workflows again (#202).

The `concurrency` group that serialises Pages deploys had been in `docs.yml` since July and was never copied across, so the recipe handed a reader the race it was written to prevent - on a page whose neighbouring section describes that failure in detail. Also pins the runner to `ubuntu-24.04` rather than `ubuntu-latest` (the page's two other blocks already did), adds the `workflow_dispatch` trigger the real redeploy workflow carries, and recommends `prodockit pins --check --offline` on a gate, since the bare form asks PyPI a question a pull request should not be able to fail on.

1.41 0.19.1 (2026-08-07)

- `prodockit pdf` now renders Mermaid diagrams on Windows (#195).

  `npm` writes three shims for each tool into `node_modules/.bin` on Windows: an extensionless POSIX shell script, plus `.cmd` and `.ps1` wrappers. Only the wrappers can be started by `CreateProcess`, but the extensionless one is the spelling every platform's documentation uses, and it is the one prodockit looked for.

  `os.path.exists` confirmed it, so detection succeeded and the build proceeded - then every diagram failed at the point of use with `[WinError 193] %1 is not a valid Win32 application`. Existing and runnable are different questions on Windows, and only the first was being asked.

  The result was the failure this project keeps producing: a PDF that built successfully, exited zero, and printed `Wrote docs\site_documentation.pdf`, with each diagram silently left as its own `graph LR` definition text. The per-diagram warning named a Win32 error code rather than a missing shim, which points at nothing a reader can act on.

  Every candidate location is now tried with an executable suffix first, so a default path or a `pdf_mmdc_bin` naming the bare `mmdc` resolves to the `mmdc.cmd` beside it. `tex2svg` was never affected - it is invoked as `node tex2svg.js`, so Node is the executable and the script is just an argument.

1.42 0.19.0 (2026-08-07)

- `prodockit pins` no longer treats a version specifier written in a comment as a declaration (#184).

  Declarations are matched textually, and `zensical ⩾ 0.0.52` reads the same whether it is a dependency or a sentence about one. A comment explaining why a package was pinned therefore became a phantom declaration site: reported in the inventory, counted towards the consistency check, and rewritten by `--set` - turning correct prose into a statement that was no longer true, in a file nobody thought they were changing.

  That mattered more after 0.18.1 made `pins --check --offline` a pull-request gate and widened the managed set to four packages including `markdown`, a word that turns up in comments far more often than `zensical`. One explanatory sentence could redden a build.

  Only the part of a line before a `#` is scanned now, in both directions: discovery ignores comments, and `--set` substitutes only over the code part, so a declaration carrying a trailing comment that quotes the same specifier no longer has both rewritten. A `#` inside quotes is not a comment, and a trailing comment still leaves its own declaration findable - narrowing far enough to break either would have been worse than the bug.

- `prodockit pdf` now prints the failing command's own stderr instead of only its exit code (#188).

  `PdfBuildError` and `SourceBundleError` have always captured the stderr of the process that failed - `pandoc`, and through it whichever PDF engine pandoc invoked - and the CLI printed only the exception's message. That message names a command and an exit code, so the single most useful thing prodockit knew about the failure was collected and then thrown away.

  Found the hard way: following the User Guide on a clean macOS machine produced `Error: pandoc exited with status 43` and nothing more. Status 43 is pandoc's `PandocPDFError` - the PDF engine failed, not pandoc - and the engine's own message said WeasyPrint could not load `libgobject-2.0-0`, named the libraries it needs and linked its installation instructions. Recovering it needed a throwaway script calling the Python API to reach `PdfBuildError.stderr`. The same build now says so directly.

  Printed whole rather than summarised: warnings come first and the real error last, so a head-truncated excerpt would hide exactly the part worth reading. Nothing is printed when the failure captured no stderr.

- Subprocess output is now decoded as UTF-8 explicitly, instead of with whatever the locale says (#191).

  Seven `subprocess.run(..., text=True)` calls named no encoding, so

Python used the locale's. That is UTF-8 on macOS and Linux and `cp1252` on a default Windows install, and every tool prodockit runs - pandoc, git, mermaid-cli - emits UTF-8. One accented author name in a `.bib` file was enough to stop `zensical serve` on Windows.

The symptom pointed nowhere near the cause. The decode fails on a reader thread inside `subprocess`, so the traceback named `threading` and `cp1252`; `run()` then returned with `stdout=None`, and the next line raised `TypeError: Incoming markup is of an invalid type: None` from BeautifulSoup. What a user saw was a type error about markup.

Guarded by a test that walks the source for text-mode subprocess calls with no `encoding=`, so a new one cannot reintroduce it. Deliberately a source check rather than a behavioural one: where this suite runs the locale is already UTF-8, so a behavioural test would pass with or without the fix.

- A new `unbookmarked` heading class removes a heading from the PDF's bookmark outline - the navigation pane a PDF reader shows down the side - separately from `unlisted`, which only keeps it off the generated Table of Contents *page* (#173).

  The two are built by different tools. Pandoc's own `pandoc.structure.table_of_contents()` builds the contents page and honours `unlisted`; WeasyPrint builds the outline separately, straight from its own UA stylesheet, and nothing about `unlisted` (or any other class prodockit stamps on a heading) ever reached it - `prodockit.pdf.css` set no `bookmark-level` rule at all. An `unlisted` `h1` therefore still became a top-level outline node, and because outline nesting follows heading level, every later, lower-level heading nested underneath it instead of under its real chapter. Silent either way: the build succeeds and exits zero.

  `unlisted` itself is unchanged, and keeps meaning exactly what Pandoc means by it. `unbookmarked` is new and additive - `prodockit.pdf.css` now gives `h1.unbookmarked` - `h6.unbookmarked` `bookmark-level: none`, and nothing prodockit generates itself carries the class, so no existing PDF's outline moves. The back-of-book index's own A/B/C letter headings, the Table of Contents title, and every cover-page heading all carry `unnumbered` `unlisted` and deliberately stay off `unbookmarked`, since a reader still needs them in the outline to navigate - keying the new rule off `unlisted` itself, instead of adding a separate class, would have removed all three.

  Documented alongside `unnumbered` in prodockit.headings and PDF generation, where the two-tables-of-contents split is now explained as its own surface for the first time. This project's own `docs/extensions/refs.md` (whose illustrative headings needed exactly this, previously worked around with a project-local `.example-heading { bookmark-level: none; }` in `docs/stylesheets/extra.css`) now uses `unbookmarked` directly instead.

- The docs build now pins `Markdown` and `pymdown-extensions`, and `prodockit pins` manages them (#178).

  Pinning a dependency exactly does nothing for the packages underneath it. `zensical` was pinned; it declares only floors for the two libraries that actually turn every page's Markdown into HTML, so every build rendered with whatever those resolved to that morning. A release of either could have changed the HTML on every page, and the site would have published it, green, with nothing committed.

  That matters more here than in most projects that render Markdown: `prodockit.pdf.css` and `prodockit.pdf.lua` both match on the specific class shapes pymdownx emits, so the renderer is an input to the PDF's own correctness, not only to the website's appearance.

  `prodockit pins` covers both by default now, so `--check` and the weekly drift job watch them alongside `zensical` and `weasyprint`. The drift job installs and reports them on both sides of its comparison - without that, a pymdownx release would have appeared in *both* builds and diffed to nothing.

  `Markdown`'s floor in `pyproject.toml` moves from 3.4 to 3.10.3 to match the pinned version, the same convention `zensical` already follows. Older releases are not known to break; the floor records what this is built and tested against.

  `ci.yml` now runs `prodockit pins --check --offline`, so a commit that moves a version in one file and forgets another fails the build instead of reaching `main`. The `--offline` form is deliberate: plain `--check` also fails when a package is behind PyPI, which would turn every open pull request red the day upstream ships a release, for a reason no contributor could act on. Watching for newer releases stays with the weekly drift job, which reports rather than fails.

- `prodockit pins --set PACKAGE=VERSION` no longer prompts for the packages it was not given, and no longer throws away the one it was.

  `--set` says "here is the version, do not ask" - the help text has always said it implies `--no-input` - but it only pre-answered the package named. Any other package in the managed set still prompted, so `pins --set zensical=0.0.53` in a release script or a drift job stopped at `weasyprint: version to set [69.0]:` and, with no stdin to answer it, aborted. Nothing was written *including the zensical sites it had been told explicitly to set* - the sharper half of the bug, because the run reported an abort rather than a partial write, and the workaround (`--package zensical --set zensical=0.0.53`) reads as though the two options mean different things.

  `--set` and `--latest` now suppress prompting outright. A package with no version given is reported under "Left untouched (no version given)" and its files are not opened. `--no-input`, which the help text named but the

command never accepted, is now a real flag for the same behaviour with nothing set.

An interrupted *interactive* run now keeps what it was already told: answers given before the interrupt are applied, the rest is left alone, and it still exits non-zero and says so. Writing nothing was the wrong reading of a cancelled prompt - the answers above it were not cancelled.

- Zensical pinned to **0.0.53** (from 0.0.52), after a byte-for-byte comparison of the site and PDF built under both.

  The PDF is **byte-identical** - same 994,803 bytes, same SHA-256, 151 pages, 268 outline entries. The site changes 22 HTML files, each by exactly three lines: the `generator` meta tag and two asset-bundle hashes. No rendered page content changes.

  The bundles themselves grew 146 bytes (CSS) and 121 (JS). The whole CSS delta is `.md-search__button` - offsets and `text-align` moved out of the base rule into `[dir=ltr]` / `[dir=rtl]` variants, which is the release's right-to-left search fix. Every class prodockit couples to is unchanged (`md-typeset` 778 occurrences either side, `md-nav` 237, `md-content` 60; glightbox and mermaid hooks identical). Unlike 0.0.52, this release redraws no icon: icons are inlined as SVG into the HTML, and the HTML is identical apart from those three lines.

  Two builds of 0.0.52 were compared first, to establish that a difference means something: they were identical down to the PDF's raw SHA-256, so the build is fully deterministic and nothing above is build noise.

  The release notes list a `pymdownx` bump to 11.0, but that is a **floor raise only** - 0.0.52 declares `pymdown-extensions ⩾ 10.21.3` and 0.0.53 declares `⩾ 11.0`, and both already resolve to 11.0.1. It was therefore not a variable in this comparison. See #178: the library that actually renders Markdown is not pinned at all.

  Every undocumented Zensical API on the Zensical coupling page was re-checked against 0.0.53 and still resolves; `__all__` is still exactly `build` / `serve` / `version`. The page's "last verified against" now reads 0.0.53.

- The "this document contains TeX maths" warning no longer fires on a page that merely *documents* maths handling (#176).

  A page quoting `<div class="arithmatex">` in a code span reaches the rendered HTML as `&lt;div class="arithmatex"&gt;` - Python-Markdown escapes the angle brackets and leaves the attribute text alone. The detector matched the bare `class="..."`, so prose about maths counted as maths, and `prodockit pdf` warned that formulas would ship as raw LaTeX in a document with no formulas in it. This project's own `devcons/limitations.md` did exactly that on every build.

It now anchors on a real `<div` / `<span` opening tag, the way the Mermaid detector always has - which is why Mermaid never had the problem. A false alarm matters more here than most: the warning exists to make a *silent* degradation visible, and one that cries wolf teaches the reader to ignore the only signal there is.

- The MathJax toolchain `prodockit pdf` uses to pre-render TeX maths is now committed and installed in CI, alongside the Mermaid one (#175).

  `tools/mathjax/` gains `package.json`, `package-lock.json` and `tex2svg.js` exactly as `prodockit init-tools` scaffolds them, with `npm ci --prefix tools/mathjax` added to `docs.yml` and `drift.yml`, both of which installed Mermaid only. Both sibling repos already tracked all three files; this brings extensions in line.

  This fixed no live defect. These docs contain no maths, so nothing was being published as raw LaTeX. The warning that prompted the work was the false positive fixed above. What it buys is that the first page to add a formula renders it, rather than shipping the source the way the Mermaid diagram in `extensions/bibliography.md` once did.

  The lockfile is committed deliberately, matching `tools/mermaid` - without it CI resolves whatever MathJax is newest rather than the version the output was checked against, which is the drift `prodockit pins` exists to catch.

  Worth knowing when relying on this: as merged, nothing here is guarded by a test that can fail. `test_no_page_contains_unrendered_tex_source` passes with the toolchain removed again, because with no maths in these docs it can only confirm that nothing unrendered reaches the PDF, not that the renderer is present. Closing that needs `pymdownx.arithmatex` enabled on this project's own docs, so the toolchain is exercised by the build that ships it.

- Documentation restructured: the six build-and-operate pages are now grouped under a **Dev Considerations** section, and `Refs` is renamed **Cross-References** (#170).

  The nav had grown to thirteen top-level entries, enough that the theme cut the tabs off. Nine now, with `docs/devcons/` holding continuous integration, repository metadata, version pinning and drift, testing your built site, Zensical coupling, and limitations and workarounds behind a short introduction.

  `Managing the build` was one 747-line page carrying three unrelated subjects; it is split at its own level-2 boundaries into `continuous-integration.md`, `repo-metadata.md` and `pinning-drift.md`, each promoted one heading level so it reads as a page rather than a fragment. **Every existing heading anchor is unchanged**, so the only part of a link that moved is the file it points at.

  Page URLs *have* moved - `/continuous-integration/`, `/testing/`, `/limitations/` and `/zensical-coupling/` are now under `/devcons/`. Every

link inside the docs and the README was updated with them; any external bookmark to an old URL will 404.

1.43 0.18.1 (2026-08-04)

- New Zensical coupling page, listing every Zensical API prodockit depends on that Zensical neither documents nor treats as public (#166).

  Zensical exports exactly three names - `build`, `serve`, `version` - and prodockit uses none of them. Everything else it reaches for is a module-level import from inside the package, so any of it can be renamed in a *patch* release without that counting as a breaking change upstream. Nothing in these docs said which, so the coupling was invisible until a release broke it.

  The page records each API with its call site and why it is needed, the undocumented *data shapes* prodockit reads (the resolved nav tree's `url` / `is_index` / `children`, `env.conf`, the packaged icon directory layout), and - deliberately - a list of the Zensical features that *are* documented, so it doubles as a triage aid when something breaks.

  It also sets out what a Zensical upgrade actually needs: not a green test run, but a build of both site and PDF with the **output compared**, since 0.0.52 silently redrew an icon with no source change. Recorded as last verified against 0.0.52.

- prodockit now says so when an undocumented Zensical API it depends on moves, instead of failing in a way that points at its own internals (#167).

  `prodockit._zensical`'s two Zensical lookups guarded the *import* and nothing after it, so they handled exactly one failure mode - "Zensical isn't installed". A renamed `ContextPreprocessor.from_markdown`, a changed signature, or a `Page.path` renamed to something else all import perfectly and then raise `AttributeError` / `TypeError` from deep inside a `zensical build`, with nothing connecting it to the version bump that caused it. None of these APIs is public - `zensical` exports only `build` / `serve` / `version` - so a rename can arrive in a patch release.

  The guards now cover the attribute access and the call as well, and emit a warning naming the API, the installed Zensical version, and what actually degrades. Deliberately *not* a bare `except Exception: return None`: `page_source()` returning None silently makes every page share one default source, each render wiping the previous page's registry entries, so cross-page references, citations and glossary terms resolve to `??` on a site that still builds and exits zero. That is #54 again, in a form no test would catch.

  A plain `ImportError` stays silent - not running under Zensical is a legitimate state for any other Python-Markdown consumer - and each API is reported once per process rather than once per page, since `page_source()` runs on every render.

1.44 0.18.0 (2026-08-02)

- A reference to a page's *title* heading now resolves in the PDF (#163). Each page's own anchor was written onto its first heading, **replacing** whatever id that heading already had - so `\ref{chapter-two}` pointed at an anchor that no longer existed. The reference still rendered its text, so nothing looked wrong; the link was simply dead, and `\autoref` printed "on page" with nothing after it.

  Sections *within* a page were unaffected, which is why it went unnoticed - the broken case is the most natural reference to write.

  The page anchor is now carried by an empty span inside that heading, so both exist: the page's own anchor for a cross-page link with no fragment, and the heading's for a reference to the heading itself. Inside rather than before, because a numbered `h1` breaks to a new page and an anchor placed before one would sit at the foot of the previous page, reporting a page number one too low.

- Cross-references say what they point at. `\ref{id}` renders the target's **number and name** - "1.1 Configuration" - because a bare "see 1.1" tells a reader nothing about what they are being sent to, and having to look it up in the contents defeats the point of the cross-reference (#151).

  An appendix needs nothing special: its letter is already the first segment of its number, so a reference to an appendix section renders "A.1 Terms".

  An `unnumbered` heading - a cover page, appendix front matter - has no number but does have a name, so it resolves to just the name. Only a genuinely unknown id is unresolved.

- New `\autoref{id}`, for references that still work on paper. It renders the same text as `\ref{id}` and additionally carries the target's **page number** in the PDF: *"Configuration is covered in 1.1 Configuration on page 12."*

  The suffix comes from prodockit.pdf's own stylesheet, so it appears only there - a page number on a scrolling website would be meaningless - and needs no configuration. Which to use is a per-reference decision: `\autoref{id}` where a printed reader needs to turn to something, `\ref{id}` where "on page N" would just be noise.

  Implemented with CSS `target-counter()`, which resolves the target's page at layout time and so needs no second pass - unlike the back-of-book index, which has to deduplicate a term repeated on one page and therefore cannot use it.

- The generated index now appears in the Table of Contents (#141). Its heading carried Pandoc's `unlisted` class, which is exactly what

`pandoc.structure.table_of_contents()` honours - so an index a reader goes looking for was the one section the contents never mentioned.

It stays `unnumbered`, so it takes no chapter number and appears at the end alongside the last chapter. Two things deliberately keep `unlisted`: the Table of Contents heading itself, since a contents listing itself is noise, and the index's own A/B/C letter headings, which would otherwise fill the contents with single letters.

- The default bottom margin is now `2.5cm` rather than `2cm`, so a multi-line running footer is not cropped when printed (#139).

  The footer is top-aligned in the bottom margin and grows *downward* as it gains lines, so whatever the margin does not use is the space left before the paper edge. This project's own footer is two lines - a copyright line and a "Made with" credit - and measured on a real render it ended **6.1mm** from the edge. Consumer and office printers commonly cannot print within 5-6.4mm, so the second line was at real risk of being cropped: the PDF was correct and the paper was not.

  2.5cm leaves about 11.1mm. The other three margins are unchanged, so pages are 5mm shorter and a long document gains a few pages - this project's own went from 134 to 139. Set `pdf_margin_bottom` to `"2cm"` to restore the previous layout, and set it higher for a footer of three or more lines.

  Raising the margin rather than moving the footer within it: both footer boxes are top-aligned with matching `margin-top` / `padding-top` so their border-tops form one continuous rule across the page. Bottom-aligning them to guarantee clearance instead would let a two-line box and the one-line page number sit at different heights and break that rule.

1.45 0.17.5 (2026-08-02)

- `prodockit --version` prints the installed version (#149). There was no way to tell which version a checkout or CI job was actually running short of `pip show prodockit`.

  Prints the bare number, matching `zensical --version` rather than click's own default of `prodockit, version X.Y.Z` - the two are normally installed and reported together, and a matching format means neither needs parsing to compare them.

- `prodockit pins` now sees a requirement with extras (#156). `package[extra]==version` is an ordinary shape - `prodockit[index]`, `uvicorn[standard]`, `celery[redis]` - and the bracket sits between the name and the operator, exactly where the matcher expected one to follow the other. Such a declaration was invisible.

The failure mode was the bad one: `pins` reported "not declared anywhere", which reads as *nothing to do* rather than *could not parse this*. A project could pin something, run `pins --check` in CI, and get a pass while the declaration drifted untouched.

Extras are recorded per site and written back on rewrite, the same way each site's operator already is, so `prodockit[index]⩾0.17.2` becomes `prodockit[index]==0.17.4` rather than `prodockit==0.17.4`. Dropping them would silently stop installing an optional dependency - for `prodockit[index]` that means the back-of-book index quietly stops being generated.

1.46 0.17.4 (2026-08-02)

Documentation only - no library, CLI or CI behaviour changes.

- The three pages covering how a prodockit build is run and kept stable - repository metadata, continuous integration, and version pinning and drift - are now one page, **Managing the build**. They were written separately and read as three answers to the same question; a reader setting up a pipeline needed all three and had no reason to expect the third. Each is now a section, introduced by a short chapter overview naming all three, and continuous integration keeps its own H2 rather than being the page's implicit subject.

  Existing links still resolve: `version-pinning.md` and `repository-metadata.md` are gone, and every reference to them - in the README, in these release notes - now points at the corresponding anchor on the merged page.

- The GitHub Actions example in that page now shows the `verify` job from 0.17.2, so the recipe a reader copies includes the delivery check rather than the version that predates it.

- The 0.17.1 entry below now carries the CI costs that were actually measured, rather than describing them as "paid once per interpreter". Installing WeasyPrint took the 3.13 job from 59 seconds to about 14 minutes - a 13-minute install, compiling from source, against roughly 20 seconds on 3.10-3.12 - and enabling the pip cache brought that install back to 15 seconds and the job to under two minutes. Those are the figures that tell a reader whether the cache is worth enabling in their own matrix.

1.47 0.17.3 (2026-08-02)

- Docs deploys no longer run from a release event, which never worked and failed silently. A release event runs against `refs/tags/<tag>`, so the Pages deployment it created carried a tag ref - and with Pages configured `source: {branch: main}`, that deployment was accepted, reported `success`, and

was then never served. The site simply carried on returning the previous build.

Every release from 0.17.0 to 0.17.2 did this, each needing a manual `gh workflow run docs.yml --ref main` afterwards. The evidence is unambiguous: across nine Pages deployments, every one from `main` went live and every one from a tag ref did not (#147).

It was not a race between the push-triggered and release-triggered runs, which is what it looked like for a long time. The concurrency group serialised them correctly - on 0.17.2 the release run started only after the push run had finished, deployed later, and still lost.

`docs.yml` now triggers on `push` and `workflow_dispatch` only. A new `release-redeploy.yml` handles the post-release rebuild by re-triggering `docs.yml` against `main`, so the deployment carries a branch ref - automating exactly what the manual fix did each time. The `tag: prodockit-v*` entry in the github-pages environment's deployment branch policies is now redundant.

Continuous integration documents the trap, since the obvious fix is the broken one.

1.48 0.17.2 (2026-07-30)

- README, the package description and the module docstring now cover `prodockit pins` alongside the other commands, so a reader arriving from PyPI sees it exists.
- New `prodockit pins` command: shows every place a build-input version is declared across a project, and moves them all together. Pinning a build means writing the same version in several files at once - a floor in `pyproject.toml`, an exact pin in each CI job that builds the docs, another in whatever job checks for drift - and nothing enforces that they agree. When they disagree the failure is quiet: CI builds with one version while the declared floor says another.

  Run it with no options for a prompt per package - press ++enter++ to take the newest release on PyPI, or type a version. Each site keeps **its own operator**, so a library floor stays a floor and a build pin stays exact; one answer updates every file correctly.

  Three shapes of declaration are recognised, because a build input is not always a pip package: a pip specifier (`zensical==0.0.52`), a GitHub runner label (`runs-on: ubuntu-24.04`) and a container image tag (`image: python:3.13`). The last two carry `pandoc`, the fonts a PDF embeds and the Chrome that rasterises diagrams - none of which pip can reach - so they belong in the same inventory. Neither has a package index to ask, so the suggested default is simply what is already set.

`--check` reports and exits non-zero if anything is behind PyPI *or* if the files disagree with each other, for a scheduled job. `--set PACKAGE=VERSION`, `--latest` and `--offline` cover the non-interactive cases.

Scans both CI layouts - `.github/workflows/` and `.gitlab-ci.yml` / `.gitlab/` - plus `pyproject.toml`, `setup.cfg` and root `requirements` / `constraints` files, so the same command works whichever host a project uses. Build output and virtualenvs are skipped, so a stale copy of a workflow inside `site/` or `.venv/` is not mistaken for a declaration.

- `weasyprint` is now pinned in the docs build and the test job, alongside `zensical`. It decides pagination, so a release that lays out one paragraph differently shifts every page number after it - and those page numbers are content, resolved into the back-of-book index and the table of contents. That is a silently wrong document rather than a failed build. Pinned in `ci.yml` too, because the real-render tests assert on where things physically land, which makes the layout engine an input to those assertions rather than an implementation detail.

- New `drift.yml`, so pinning does not mean going quietly stale. Weekly it builds the docs twice in one job - once with the pinned versions, once with the newest - diffs the results byte for byte, runs the built-output checks against the newer build, and opens an issue saying what an upgrade would change. Both builds share a job, so pandoc, Chrome, fonts and the runner image are identical between them and any difference is attributable to the upgraded packages alone. It reports rather than fails, and keeps one open issue updated in place: a scheduled job that goes red every week trains everyone to ignore it.

- CI runners are pinned to `ubuntu-24.04` rather than `ubuntu-latest`. The image is the build input pip cannot reach: `pandoc` comes from it (and distribution packages lag upstream far enough that some markdown edge cases parse differently), as do the fonts the PDF embeds and the Chrome that rasterises Mermaid diagrams. On `ubuntu-latest` all three move the day GitHub migrates the label, with nothing committed - the same silent change the package pins exist to prevent, one layer down.

  `ubuntu-latest` already *is* 24.04, so nothing changes today; it takes effect at the migration, which is exactly when a documentation build wants to be told rather than surprised. Pinned images are retired about a year after the following LTS and the job then fails outright rather than drifting - the better failure, and at a time of your choosing.

- New Version pinning and drift page documenting the whole arrangement - where a version gets declared and why the forms differ, `prodockit pins`, and a drift job for **both** GitHub Actions and GitLab CI, the latter using pipeline schedules and the GitLab issues API.

- `zensical` now declares a floor (`⩾0.0.52`) rather than being left entirely

open. It records the version prodockit is developed and built against - not a minimum below which anything breaks, since 0.0.50 and 0.0.51 both work. Zensical is pre-1.0 and prodockit reaches well past its public surface (the config loader, the per-page render context `prodockit.headings` detects, the icon set `prodockit.pdf.icons` resolves against), so which version produced a given build is worth recording. A floor rather than an exact pin because prodockit is a library: `==` would propagate to every consumer and conflict with any project needing a different Zensical.

What prompted this is worth keeping visible. Rebuilding against 0.0.52 and diffing the output byte for byte against 0.0.51: the PDF is identical, and every page of the website differs - Font Awesome moved 7.2.0 to 7.3.1 and redrew the GitHub brand icon used in the header's repository link and the social footer. One visible change out of the nine icons the site uses, arriving with nothing committed. The rest is the generator version string, content-hashed asset filenames, and minified bundle churn.

A floor does not prevent that recurring: dependency resolution still takes whatever is newest, so the next Zensical release can change the published site the same way. Reproducible builds need a constraint on the *build* side - in `docs.yml` - rather than in library metadata.

1.49 0.17.1 (2026-07-29)

- This project's own PDF now has the back-of-book index its own docs describe. `zensical.toml` never set `include = true` for `prodockit.index`, so the live `\index{}` markers in `docs/extensions/index-terms.md`'s `=== "Result"` tabs produced nothing: the page documenting the feature sat in a PDF that didn't have it.

  Turning the setting on alone would only have indexed those five demo markers, so the docs are now marked properly throughout: every module at its own page's opening sentence, every option/setting/fixture in the reference table that defines it, and the concepts and external tools where each is introduced. Options are nested under what they belong to, so the `source` / `registry` / `unresolved` that five different extensions each define group per module instead of collapsing into one misleading entry. That gives a real two-page index of about 95 entries, covering every marker shape the extension supports.

  Costs one extra `pandoc` +WeasyPrint pass on that one build. The twelve single-page `prodockit pdf -m` builds in `docs.yml` pay nothing: `build_pdf()` skips the second pass entirely for a document with no markers, and none of those pages has any. Mermaid and TeX pre-rendering both happen before either pass, so neither is repeated. Measured locally at roughly 8s → 12s for a 119-page document, against a docs workflow dominated by apt, Chrome and npm setup.

- The 0.17.0 fix above now has a permanent CI guard. Its regression test needs

a real `pandoc`+WeasyPrint install, so it is deselected in the `test` job, which installs neither - the shipped fix was effectively unguarded. `tests/test_built_docs.py` gains three `built`-marked checks, which run in `docs.yml` after a real build: that no marker reached the PDF's text layer, that the index was generated with every marked term in it, and that each entry cites a page the term is actually on.

The text-layer check matches only a marker followed by a real digit, rather than the blunter substring the synthetic test can afford. These release notes legitimately print `⟦prodockit-index-N⟧` (with a literal "N") in the 0.17.0 entry below, and name the `h2.prodockit-index-letter` CSS class in an earlier one - both are prose about the feature, and both belong in the text layer.

- A long index term no longer overflows its own column. An index entry is often a single unbroken token with nowhere to wrap - a dotted module path, a long option name, a function signature - and `div.prodockit-index-entry` set no `overflow-wrap`, so such a term ran straight over the column rule into the next column's entries, and off the page edge entirely from the right-hand column. Found while indexing this project's own docs. `overflow-wrap: break-word` (not `break-all`, so ordinary multi-word terms still break at their spaces first) fixes it. Covered by a real-render test that measures where the glyphs actually land, since it renders as perfectly ordinary text either way and only its position gives it away.

- The generated index's pages are now headed "Index" (or whatever `prodockit.index`'s `title` setting says) rather than by the last chapter of the document. The index's own `h1` is `unnumbered` - correct, since it must not take a section number or a Table of Contents entry - but `unnumbered` is also what excludes a heading from feeding the running header's `chapter-title` string. That exclusion suits the Table of Contents, which sits at the front where `chapter-title` is still empty, and fails for the index, which is always the very last thing in the document: whatever chapter came last simply stayed in the header. This project's own PDF was headed "18. License" across its entire index. The heading now carries a `prodockit-index-title` class with a `string-set` rule of its own; nothing else about it changes.

- CI now installs WeasyPrint for the `test` job, so the nine real-render tests that assert on where things actually land in a finished PDF - which page a heading starts on, what the running header says, whether a long index term stayed inside its column - finally run. They are gated on a real `pandoc`+`weasyprint` install and `ci.yml` had only `pandoc`, so they skipped silently on every run; `docs.yml` does install WeasyPrint, but runs only `-m built` against `tests/test_built_docs.py`, so it never reached them either. The behaviour they exist to pin was going unchecked everywhere, including both fixes above.

  The tests themselves add about 20 seconds per matrix entry. Installing WeasyPrint costs more, and unevenly: it pulls in packages with no wheel

published for every interpreter the matrix covers. On 3.10-3.12 the install step took roughly 20 seconds; on 3.13 it spent **13 minutes** compiling them from source, taking that job from 59 seconds to about 14 minutes.

`actions/setup-python`'s own pip cache is now enabled, which holds pip's *built* wheels as well as its downloads. That brought the same 3.13 install back to **15 seconds** and the job to under two minutes.

1.50 0.17.0 (2026-07-28)

- Back-of-book index markers no longer leave anything in the PDF's text layer (#133). Every `\index{Term}` used to deposit a `⟦prodockit-index-N⟧` token next to the word it marked - 67 of them in this project's own User Guide. They were invisible on the page, but real text in the file, so they surfaced in copy and paste, in the reader's own search, in text extraction, and, worst, in screen readers, read out mid-sentence.

  The markers had to stay findable, which is why they were text: the second pass locates each one to learn its page number, and a `font-size: 0` span is dropped from the text layer entirely, leaving nothing to find. Shrinking further was never an option.

  Each occurrence is now marked with an *empty* span carrying only an `id`, and its page is read back from the PDF's own named destinations - WeasyPrint emits one per element `id`, whether or not anything links to it. Nothing is encoded as text, so nothing can leak. An empty span also occupies no width, which removes the previous design's awkward question of whether stripping 67 tiny spans between passes might reflow the very pages whose numbers they had just recorded.

  Needs no configuration change, and no new dependency: `pymupdf` was already required for a generated index, and the API used has been available since well below the existing floor.

1.51 0.16.0 (2026-07-28)

- The website's `prodockit-v0.41.0` and the PDF's `{RELEASE}` come from deliberately different sources - `git describe --tags` on the local checkout, and the host's releases API - and could disagree with nothing saying so (#125). A reader comparing a published site with its downloadable PDF could see two different release numbers, and neither build had failed.

  Neither source changes: each is right for its own context. `prodockit-v0.41.0` is re-evaluated on every website rebuild, including every save under `zensical serve`, so it must not make a network call; `{RELEASE}` serves a cover page that isn't part of a macro-rendered site at all. What was missing is that the disagreement was invisible.

`prodockit pdf` now warns when the two will show different things, naming both values and where each came from. The macros pass warns separately when `prodockit-v0.41.0` came back empty *because* the checkout is a shallow clone, which fetches no tags even from a repository that has them - the failure behind #122, silent because an empty value just renders as a missing line. The warning names `fetch-depth: 0` and `GIT_DEPTH`, and fires once per process rather than once per rebuild.

A project with no tags at all stays silent: that is a normal state, and warning about it would only train people to ignore the message. See Limitations and workarounds.

1.52 0.15.2 (2026-07-28)

- `prodockit.testing`'s Mermaid check now detects a diagram whose arrows were swallowed by font ligatures. A PDF set in a font with programming ligatures - JetBrains Mono, a common choice for code blocks - renders `-->` as a single glyph that extracts back out as `//>`, leaving every arrow-based pattern blind to an unrendered diagram. Node-definition brackets (`id[Label]`) survive extraction, so they now count as evidence too. Found in `prodockit-template`, whose own PDF uses that font.

  Accepting bracket syntax needed the keywords tightened first, or it would have fired on ordinary prose. `graph` / `flowchart` now require their direction token (`graph LR`), which no sentence produces by accident, and the four diagram types that are also plain English words - `gantt`, `journey`, `pie`, `timeline` - accept only arrow or entity-relationship evidence, never brackets. Without that, a line beginning "timeline of the project" followed by `data[1]` read as an unrendered diagram.

1.53 0.15.1 (2026-07-27)

- Documentation only. New Continuous integration page: complete, working GitHub Actions and GitLab CI recipes for building a prodockit site, and the reasoning behind each part (#124).

  This knowledge previously existed only as comments scattered across three projects' workflow files, which is why the same mistakes kept recurring. The page is organised around the failure modes that are *silent* - fonts not installed (WeasyPrint substitutes one and the PDF just looks wrong), a shallow clone fetching no tags (the release line vanishes), the renamed Puppeteer variable, and a release deploying before its own tag exists.

1.54 0.15.0 (2026-07-27)

- New `prodockit.testing` package - `pip install prodockit[testing]`. A pytest plugin giving a project `prodockit_*` fixtures for its own built site and

PDF, resolved from its Zensical config rather than an assumed layout, plus checks for the failure modes every prodockit project shares. See Testing your built site.

Chiefly `assert_no_unrendered_mermaid()` and `assert_no_unrendered_tex()`, which turn the 0.12.0 build warning into a test failure - three projects published PDFs full of raw `flowchart LR ...` source and literal LaTeX before anyone noticed (#124).

The Mermaid check requires a diagram-type keyword *and* Mermaid's own link syntax nearby. Several diagram types (`graph`, `pie`, `journey`, `timeline`) are ordinary English words, and PDF line breaks fall wherever text wraps, so a keyword-only check read "a visual commit graph and richer history browsing" as an unrendered diagram - passing locally and failing in CI only because different fonts wrapped that sentence differently.

The plugin registers through pytest's entry point, so it loads into every test run in an environment where prodockit is installed. It imports nothing heavy at module scope, prefixes every fixture, and fails individual fixtures rather than collection, so an unrelated project is unaffected.

1.55 0.14.0 (2026-07-27)

- New `prodockit init-tools` command: scaffolds the Node tooling `prodockit pdf` needs to render Mermaid diagrams and TeX maths, then prints the `npm` commands, `.gitignore` lines and CI environment variables to finish the job. See Mermaid diagrams and TeX maths.

  `prodockit.pdf` has always looked for `tools/mermaid/node_modules/.bin/mmdc` and `tools/mathjax/tex2svg.js` while shipping neither, leaving every project to hand-write the same two manifests and the same `tex2svg.js` (#124). All three projects using it got that wrong independently: one had no `tools/` directory at all and published PDFs full of raw `flowchart LR ...` source, two set the pre-rename `PUPPETEER_SKIP_CHROMIUM_DOWNLOAD` that puppeteer 25.x ignores, and one committed two config files nothing reads. The canonical copies now live in the library, pinned in one place.

  Existing files are never overwritten without `--force`, since a project will have run `npm ci` against its own committed lockfile.

- The missing-renderer warning added in 0.12.0 now points at `prodockit init-tools` rather than `npm ci --prefix tools/mermaid`, which could not work in the case that actually happened - no `tools/` directory to install into.

1.56 0.13.0 (2026-07-27)

- Documented the Python version requirement. `requires-python = "⩾3.10"` has always been enforced by `pip`, and CI has always tested 3.10-3.13, but

nothing said so anywhere a reader would look: the PyPI classifiers listed only `Programming Language :: Python :: 3`, and neither the README nor the installation page mentioned a version at all. Per-version classifiers added, and Installation now opens with the requirement plus a full table of what `pip` does and doesn't install

- including `pandoc` / `weasyprint`, and the Node tooling needed only for Mermaid diagrams and TeX maths in the PDF.

- New `prodockit sync-repo` command, and the `prodockit.sync_repo` module behind it: keeps `repo_url`, `repo_name`, `[project.theme.icon] repo`, `edit_uri` and your README's badge row matching the git remote a checkout actually uses, so forking or mirroring a project between GitHub, GitLab and Bitbucket doesn't leave stale links, the wrong brand icon, or badges pointing at somebody else's repository. `--check` writes nothing and exits non-zero on drift, for CI. See Repository metadata.

  This was previously a `sync_repo_icon.py` script copied byte-for-byte between two consuming projects (#124). Two things changed in promoting it: the default branch is now detected from the remote rather than hardcoded to `main`, and `repo_name` keeps whichever shape (`owner/repo` or bare `repo`) your config already uses, since Zensical prints it verbatim in the site header and the script's fixed choice would have restyled the header of any project using the other one.

- The command-line entry point moved from `prodockit.pdf.cli` to `prodockit.cli`, now that it has commands unrelated to the PDF build. `prodockit.pdf.cli` re-exports `main`, so an entry point recorded in an already-installed environment keeps working.

1.57 0.12.0 (2026-07-27)

- `prodockit pdf` now warns when a document contains Mermaid diagrams or TeX maths but the renderer needed to turn them into static images wasn't found. Both have always been optional, and both deliberately leave the content untouched rather than failing the build - the right default for a project using neither, but silent for one that *is* using them. That silence let raw `flowchart LR ...` source and literal LaTeX reach the published PDFs of three separate projects, including this one's own. The build still succeeds; the degradation is simply announced, and the warning names the fix rather than just the symptom.
- Fixed this project's own docs: the architecture diagram in `extensions/bibliography.md` had never rendered in any published PDF, for exactly the reason above. `tools/mermaid` and the CI steps to install it are now in place - so the page describing how Mermaid fences are pre-rendered finally demonstrates it.

1.58 0.11.1 (2026-07-27)

No code changes - fixes a regression in 0.11.0's own release:

- The "Release: `<tag>`" cover-page line added in 0.11.0 never actually appeared on the deployed site or PDF - `actions/checkout`'s default shallow clone (`fetch-depth: 1`) fetches no tags at all, so `prodockit.zensical_macros`' `prodockit-v0.41.0` (`git describe --tags --abbrev=0`) always returned `""` in CI, even though it worked correctly in any full local checkout. Fixed by adding `fetch-depth: 0` to this project's own `docs.yml`/`ci.yml` checkout steps. Confirmed directly against the redeployed live site.
- Fixed #120: `.cover-hero-subtitle`/`.cover-hero-release` (this project's own `docs/stylesheets/extra.css`) both rendered in black rather than the intended grey in the PDF - `color: var(--md-default-fg-color--light)` is undefined in `prodockit.pdf`'s own generated CSS, and an unresolved `var()` with no fallback falls back to the *inherited* value instead of erroring.
- Fixed #121: rather than just special-casing the PDF, added a `var()` fallback pointing at this project's own explicit `--prodockit-fg-color-light` custom property - the live website still gets the real, theme-adaptive Zensical variable whenever it's actually defined, but a future Zensical rename/removal of that variable can no longer silently break this project's own PDF-visible text again.

1.59 0.11.0 (2026-07-27)

New `prodockit-v0.41.0` variable in `prodockit.zensical_macros`: the latest git tag reachable from `HEAD` (e.g. `"1.2.0"`, `""` if the checkout has no tags at all), matching `word_count`/`repo_url`'s existing pattern. Promotes a one-off custom macro `prodockit-userguide` already had in its own project-local `macros.py` into the shared library, so every project gets it for free instead of hand-rolling the same `git describe --tags --abbrev=0` shell-out. Resolves identically for the website and for `prodockit pdf`, since both render through the same macro environment - unlike `prodockit.pdf`'s own `{RELEASE}` cover-page marker, which queries the host's GitHub/GitLab API instead, for a project whose cover page isn't part of a live, macro-rendered site at all.

This project's own docs site now shows "Release: `<tag>`" on its cover page, using the `.cover-hero-release` CSS class that was already defined but never actually used - enabling the macros plugin here for the first time in the process. Also fixed a real bug found while wiring this up: `.cover-hero-release` rendered in a bold weight in the PDF (`Inter-Ultra-Bold` instead of `Inter`, confirmed via the PDF's own extracted font info) - it was missing the `font-weight: 400` its sibling `.cover-hero-subtitle` already has, for the same Pandoc-pipeline reason.

Fixes #116. See #120 for a related, separate rendering issue found along the way

(both `.cover-hero-subtitle` and `.cover-hero-release` render in black rather than the intended light gray in the PDF - not fixed here).

1.60 0.10.9 (2026-07-26)

Docs only, no code changes:

- New Limitations and workarounds page, consolidating every confirmed limitation across prodockit's three surfaces (the Python-Markdown extensions, `prodockit.pdf`, and `prodockit.zensical_macros`) and its workaround in one place, including cross-page resolution going stale under `zensical serve`'s live reload (#99) - previously scattered across `pdf.md`'s own "Limitations and workarounds" section, which is now a short pointer here instead.
- `bibliography.md`'s "What this project's own template and user guide currently do" section no longer says neither downstream project has adopted `prodockit.bibliography` - prodockit-template has since migrated, and is now a real worked example of Multiple sections: a cited-only `references.md` and a separate, everything-included `bibliography.md` from a distinct further-reading `.bib` file.
- Added GitHub issue templates (bug report, feature request), matching prodockit-template's own.

1.61 0.10.8 (2026-07-26)

Breaking: the two citation extensions swap syntaxes. `prodockit.bibliography` now uses `\cite{id}` - the natural spelling for the workflow most projects reach for first - and `prodockit.citations` moves to `\citeref{id}`.

If you use `prodockit.citations`, replace every `\cite{id}` in your content with `\citeref{id}`. A `\cite{id}` left behind will no longer resolve: with `prodockit.citations` alone it falls through as literal text, and in a build that also enables `prodockit.bibliography` it will be read as a `.bib` key instead.

The two extensions still own distinct syntaxes, so either can be enabled alongside the other without hijacking it - only which name belongs to which has changed. Each is now pinned by a test asserting it leaves the other's syntax alone; neither had one before, so nothing would have caught the two silently overlapping.

Multi-key citations remain a `prodockit.citations` feature (`\citeref{id1,id2}`); `prodockit.bibliography` matches single keys only, for the reasons its own documentation gives.

Part of #111; the matching updates to `prodockit-template` and `prodockit-userguide` follow separately.

1.62 0.10.7 (2026-07-25)

Two numbering fixes, both cases of the same shape: a raw-text pre-scan and a parsed-document count applying different rules to the same content.

Website and PDF disagreeing on appendix letters. Appendix lettering was computed twice. The website gave a letter to any page whose front matter set `is_appendix`, unconditionally; the PDF's Lua filter counted appendix h1s instead. An appendix page contributing no numbered h1 - none at all, or one marked `unnumbered`, which the filter skips - gave Lua nothing to count, so every later appendix came out a letter early. Reproduced against `prodockit-template`: the same Bibliography page rendered as "Appendix E" on the website but "Appendix D" in the PDF.

`prodockit.pdf.build` now assigns every appendix page's letter once, by position in the page list, and stamps it on that page's heading for `Header()` to read rather than counting for itself. A page with no numbered h1 still consumes its letter, so the two stay in step.

Setext headings invisible to the nav pre-scan. `_count_top_level_headings()` matched ATX headings only, so a title underlined with `=` was never counted - though Zensical's renderer and Pandoc both produce a real h1 either way, and both number it. Later pages' start counts came out short, which broke numbering twice over: the website contradicted itself, giving the next page the chapter number a setext heading had already taken, and it fell one behind the PDF. Each rule of the setext syntax was checked against the real renderer rather than assumed - a single `=` is enough, a `-` underline is an h2, a two-line paragraph followed by `=====` is no heading at all, and `attr_list` puts `{: .unnumbered }` on the text line.

Stale cross-page references under `zensical serve`. `preseed()` is deliberately first-wins so a duplicate id resolves to the first page in nav order. Under `zensical build` that is all it has to do; under `zensical serve` the process outlives the files, and first-wins silently discarded fresh data - an edited definition kept its original text, and a deleted one stayed resolvable, until the dev server restarted. `preseed_attr_from_nav()` now rebuilds the provisional set from scratch on each scan, and `prodockit.headings` keys its cached scan on every nav page's mtime and size as well as the numbering settings, so an edit to a page a given render doesn't touch still invalidates it.

Note this fixes the pre-scan, not Zensical's incremental rebuild: verified against a live `zensical serve`, editing page A does not cause page B to re-render, so B's output still only catches up when B itself is re-rendered. What is guaranteed now is that a page being re-rendered resolves against current disk state rather than a snapshot from server startup.

Fixes #104, #106 and #99.

1.63 0.10.6 (2026-07-25)

Fixed footnote text in the PDF rendering in a column roughly two thirds of the page's content width, wrapping a footnote onto five short lines whose first held just two words.

This had been documented in `prodockit.pdf.css`'s own `.pdf-footnote` rule as an unfixable WeasyPrint `float: footnote` limitation being tracked upstream. That was a misdiagnosis. The real cause is Pandoc's HTML writer hard-wrapping its output at ~72 columns, inserting newlines *inside* the `<span class="pdf-footnote">` carrying a footnote's text. Those newlines are insignificant whitespace in HTML, but WeasyPrint's `float: footnote` width computation treats them as hard break opportunities when sizing the footnote area, so the rendered text collapses toward the longest *source* line rather than the page's content width.

Confirmed by holding the HTML and CSS constant and varying only the Pandoc step: the same document rendered 304.1pt wide through Pandoc but 462.9pt straight through WeasyPrint. `prodockit.pdf.build` now passes `--wrap=none`, giving 474.2pt of a ~482pt content width. WeasyPrint 69.0 is already the latest release, so waiting upstream had no path forward.

This does not stop footnotes wrapping in the PDF - `--wrap=none` governs only newlines in the generated HTML source, never the engine's own line breaking. A long footnote still occupies as many lines as it needs, each now using the full measure: a seven-sentence footnote renders on six full-width lines rather than ten narrow ones.

The misleading `KNOWN LIMITATION` comment in `prodockit.pdf.css` has been replaced with the real cause so it isn't re-derived, and two regression tests added - one asserting the flag at the command level (so it can't be dropped where Pandoc isn't installed), one measuring real rendered text width via a genuine Pandoc/WeasyPrint build, since the CSS is identical either way and only a real render can tell the two apart.

Fixes #101, reported downstream as prodockit-template#95.

1.64 0.10.5 (2026-07-25)

Fixed a real bug found by reproducing it directly: a cross-page `\ref{id}` under `zensical build` depended on whether the page defining `id` happened to be rendered before the page referencing it, in the same Python process - `zensical build` renders pages neither in nav order nor necessarily all in one process, so this was pure luck. Reproduced locally on a 3-page site: the previous release left 1-5 references as `??` per build, varying from one otherwise-identical build to the next (only 2 of 12 clean builds fully resolved).

`prodockit.headings` now pre-scans every nav page's headings (ids, levels,

section numbers) into the shared registry before any page is converted - the same idea `prodockit.citations` / `prodockit.glossary` already use for their own cross-page definitions - so resolution no longer depends on build order. A page actually rendered in this process still supersedes its own pre-scanned entries with the real ones. 20 consecutive clean builds now produce byte-identical, fully-resolved output; `prodockit-template`'s entire built site is byte-identical before and after this change.

Also fixed a second bug found while testing the above: extension order isn't guaranteed, so `prodockit.refs` can construct its own default `HeadingsExtension` and trigger the pre-scan with per-document numbering before a project's configured `numbering = "continuous"` instance runs - silently showing a cross-page reference's number one step behind (`1.1` instead of `2.1`) roughly 1 build in 12. The pre-scan now reruns if a differently-configured instance appears.

Fixes #54. See #99 for a related, separate limitation found along the way (this pre-scan can go stale under `zensical serve`'s live-reload - not fixed here).

1.65 0.10.4 (2026-07-25)

- Added `CONTRIBUTING.md` and a `.github/pull_request_template.md`, adapted from prodockit-template's own - library-specific setup (`pip install -e ".[dev]"`, `pytest` / `ruff` / `mypy`, the real-`pandoc` requirement for `prodockit.bibliography`'s own tests) rather than the template's assignment-writing framing. Linked from README.md's Development section.
- Docs: added an admonition to `headings.md` documenting a real gap this project's own docs hit directly while enabling every prodockit extension on its own site (#87) - Zensical's automatic cross-page id sharing only warns (rather than raising) on a heading name shared across pages, and build order isn't guaranteed stable, so the "keeping the first" winner can change between builds. Documents the fix - an explicit, page-prefixed id via `attr_list` - pointing to this project's own docs as a worked example.

No code changes.

1.66 0.10.3 (2026-07-25)

Docs: several extension pages mixed "why it was built this way" design rationale into their opening paragraph, ahead of the practical "what it does"/"how to use it" content most readers want first - moved that reasoning further down each page instead:

- `bibliography.md`: trimmed the intro to the core value proposition, moved the Pandoc-delegation rationale and architecture diagram out of Requirements (now just what to install) into a new "How it works" section

after Reference, and folded the "can be enabled alongside prodockit.citations" note into Comparing the two approaches.
- `citations.md` / `glossary.md`: moved the "why bundled into one extension, unlike headings/refs" rationale from the intro into their own Syntax section.
- `tables.md`: moved the "auto-enables Python-Markdown's own tables extension" implementation note from the intro into Syntax.
- `index-terms.md`: moved the "why a Markdown extension, not attr_list" rationale from the intro to the end of CSS hooks, wrapped in its own admonition naming the subject explicitly.

`headings.md` / `refs.md` were already lean and needed no changes. No syntax, behaviour, or code changes.

1.67 0.10.2 (2026-07-25)

Docs: `extensions/index-terms.md` described the live website's search as generic "browser/Ctrl-F search" - updated to point at Zensical's own built-in site search instead, a more accurate and discoverable description of how a reader actually finds a term on the live website. No code changes.

1.68 0.10.1 (2026-07-24)

Docs: `prodockit.index`'s marking syntax and `prodockit.pdf`'s back-of-book index *generation* were split across two pages (`extensions/index-terms.md` and `pdf.md` respectively), even though the marker is useless without turning index generation on and vice versa - `prodockit.bibliography`'s own docs already combine marking and generation into one page. Moved the generation content into `extensions/index-terms.md` as a new "Generating the index" section, merged the per-feature rendered-output examples into their existing marking sections instead of duplicating them, and renamed the page from "Index terms" to "Index (pdf-only)" now that it covers the whole feature. `pdf.md` keeps only a short pointer, matching how `prodockit.bibliography` is treated there. No code changes.

1.69 0.10.0 (2026-07-24)

`prodockit.bibliography`'s `\bibliography` marker now takes two optional, positional parameters - `\bibliography{<file>}{<true|false>}` - so a project can generate both a strict **References** section (only sources actually `\citebib{}`-cited in the text) and a broader **Bibliography** section (every entry, including background reading that's never individually cited) in one build, from the same or different `.bib` files:

```
←!— references.md —→
\bibliography{}{true}
```

```
←!— bibliography.md —→
\bibliography{background.bib}
```

Bare `\bibliography` is completely unchanged - fully backward compatible, no breaking change. A `\citebib{id}` citation now cross-links to whichever marker's page actually defines that entry (via a new, lightweight `.bib` entry-key discovery helper, not a CSL reimplementation) rather than assuming a single global bibliography page - the common single-file case is unaffected. See Multiple sections: References and Bibliography for the full syntax and worked examples.

Fixes #89.

1.70 0.9.0 (2026-07-24)

Breaking: `copyright` / `pdf_copyright` are now a real HTML fragment, rendered as a real DOM element in the PDF's running footer via CSS Paged Media's `position: running()` / `content: element()`, instead of being escaped into a CSS `content: "..."` string. This is what makes a real `<a href="../...">` link inside either value survive as a real, clickable link in the PDF - on every page, not just wherever the source element itself sits - matching how Zensical's own website-side `copyright` setting already works. Use a real `<br>` for a forced line break; the `\A` CSS-escape trick added in 0.8.0 only ever worked for a plain string, not real markup, and no longer applies - update any existing `pdf_copyright` using it to a real `<br>` instead.

`prodockit.pdf.css.build_css()` no longer takes a `copyright_text` parameter (`site_name` is unaffected, still a plain CSS content string) - no formal, versioned public API surface yet for `prodockit.pdf` (see prodockit-extension#7), so this is an acceptable break at this stage.

This project's own cover page (`docs/index.md`'s hero subtitle) no longer hyperlinks the word "Zensical" - it stays as plain text, matching this project's own PDF footer now crediting Zensical/prodockit with real links instead of the cover page doing it via a website-only, PDF-invisible link.

1.71 0.8.1 (2026-07-24)

Docs: this project's own docs site and PDF were missing the "Made with Zensical and prodockit" credit line that `overrides/partials/ copyright.html` / `pdf_copyright` (new in 0.8.0) already give a downstream project - added both here too, via a new `overrides/partials/ copyright.html` for the website and `extra.pdf_copyright` in `zensical.toml` for the PDF, so this site credits itself the same way a project built with it does. No library code changed.

1.72 0.8.0 (2026-07-24)

New `pdf_copyright` setting: `project.copyright` (a plain, native Zensical setting) already feeds the footer of both the website and the PDF by default - `pdf_copyright` is an opt-in override for the PDF's footer only, for a project that wants its PDF footer to say something different from its website's (e.g. adding a "Made with Zensical and prodockit" credit line only to the downloadable PDF, not the live site). Write a forced line break in either setting with a literal `\A` inside a TOML *literal* string (`'''...'''`) - see Copyright text for the full mechanism and why a literal string is required.

Also fixed a real, previously-undocumented rendering gap found while building this: a `\A` forced line break inside a `content` string only actually renders as a line break under `white-space: pre-line` - under WeasyPrint's default `white-space: normal` it silently collapsed to a plain space instead. Both the single-sided and double-sided verso copyright footer boxes now set `white-space: pre-line` so the forced break always works as expected.

1.73 0.7.1 (2026-07-24)

This project's own documentation site now enables every prodockit extension (`prodockit.headings`, `prodockit.refs`, `prodockit.citations`, `prodockit.glossary`, `prodockit.bibliography`, in addition to the `prodockit.tables` / `prodockit.index` already enabled) via `zensical.toml`, dogfooding the full set rather than just the two used to build this site previously.

Doing so surfaced a real bug: Zensical does not render pages in a stable order between builds, so a heading name shared across two or more pages (e.g. "Quick start", "Syntax", "Options") non-deterministically resolves its id collision differently from one `zensical build` to the next - confirmed by running repeated clean builds and observing the reported "keeping the first" winner change between runs. Fixed by giving every colliding heading across the docs an explicit, unique, page-prefixed id via `attr_list` (e.g. `## Quick start {: #refs-quick-start }`), rather than relying on build order at all. No library code changed - this is a docs-content-only fix, and not something a project sharing a heading name across its own pages will normally have to think about, since a one-off name collision is far less likely there than in this project's consciously-parallel per-extension documentation structure.

1.74 0.7.0 (2026-07-24)

Breaking: `prodockit.bibliography` now uses its own `\citebib{id}` syntax instead of `\cite{id}`. Previously it registered the same `\cite{id}` pattern `prodockit.citations` uses, at the same inline-pattern priority - enabling both extensions together left it undefined which one actually resolved a given `\cite{...}` occurrence. Renaming `prodockit.bibliography`'s own syntax removes the conflict entirely: both extensions can now be enabled in the same

build with no interference, each citing its own sources by its own marker. A project still using `prodockit.bibliography` on its own needs to update every `\cite{id}` in its source to `\citebib{id}` - the old syntax no longer resolves.

1.75 0.6.8 (2026-07-21)

`build_pdf_from_zensical_config()` (what `prodockit pdf` runs) now supports cover page markers, so a project no longer needs its own custom Python just to fill in a cover page's word count/repo URL/release tag - found via `prodockit-template`, whose `build_pdf.py` had grown to nearly nothing except this one piece:

- `{WORDCOUNT}` - the site-wide word count (the same value a `54,092` website macro shows).
- `{REPOURL}` - the git-detected repo URL.
- `{RELEASE}` - the latest published GitHub/GitLab release tag - the whole line is dropped instead if there isn't one.
- `prodockit` - substituted literally, since `prodockit pdf` never evaluates Jinja.

All four are opt-in by literally writing the marker in your `nav`'s index page - no new `zensical.toml` setting needed. See Cover page markers.

Also new: `pdf_extra_css`, a stylesheet meant *only* for the PDF (e.g. a rule that would look wrong on the live website), concatenated after `extra_css` - the same `["stylesheets/print.css"]` role a project's own custom PDF-build script might have hardcoded outside `zensical.toml` entirely before, now expressible as ordinary configuration.

Also fixed two real bugs found while building this:

- `extra_css`'s (and now `pdf_extra_css`'s) own relative `url(...)` references (e.g. a light/dark logo swap or a header background image) were passed through unresolved, pointing nowhere once compiled into the PDF's own temporary work directory - now resolved and base64-embedded, matching how a local `<img>` reference already was.
- `copyright` / `site_name` were passed straight into `build_pdf()`'s generated CSS `content: "..."` string with no escaping at all - `project.copyright` is commonly a triple-quoted TOML string spanning multiple lines, and a raw embedded newline (or a literal `"`) silently broke the whole generated rule, dropping the running header/footer entirely with no error. Both are now collapsed to one line and escaped before being passed through.

1.76 0.6.7 (2026-07-21)

Fixed `prodockit.pdf.html.fix_up_page_html()` permanently embedding *both* halves of a `#only-light` / `#only-dark` (or GitHub's `#gh-light-mode-only` / `#gh-dark-mode-only`) image pair in a PDF, stacked one after the other, instead of just

one - found via `prodockit-template`'s own cover page hero graphic showing twice. A PDF has no light/dark toggle to make that convention meaningful, but `to_base64_data_uri()` already strips anything from `#` onward before resolving the file (to find the right one), so the resulting `data:` URI has no trace of the fragment left for any stylesheet to hide either half by. The `#only-dark` / `#gh-dark-mode-only` half is now dropped entirely rather than embedded.

1.77 0.6.6 (2026-07-21)

- Docs: the cover page hero graphic (`docs/assets/cover-hero-*.svg`) used a different colour palette in light mode (blue) than in dark mode (green) - recoloured the light variant to match dark exactly, so the hero reads the same regardless of theme. The "Download PDF" button also picked up this same green, rather than the theme's default primary colour.
- `prodockit.pdf.css`'s back-of-book index letter-group headings (`h2.prodockit-index-letter` - the "A", "B", "C" separators) were hardcoded to the hero graphic's *old* light-theme blue - updated to match the now-green hero, which a PDF always shows regardless of a project's own website light/dark toggle.
- No functional (Python package behaviour) changes beyond the index letter colour.

1.78 0.6.5 (2026-07-21)

Extends the 0.6.4 always-excluded-directory mechanism in `prodockit.pdf.source_bundle` to two more classes of vendored, never student-written content:

- Any directory literally named `styles` - a Vale `StylesPath` (conventionally named this way) holds downloaded rule packs (e.g. the Microsoft, proselint, and Readability style guides), typically tracked for offline/CI builds rather than gitignored.
- Common dependency lockfiles by exact file name - `package-lock.json`, `npm-shrinkwrap.json`, `yarn.lock`, `pnpm-lock.yaml`, `Pipfile.lock`, `poetry.lock`, `Cargo.lock` - machine-generated by a package manager, never hand-written, and often thousands of lines each.

Neither is project-configurable, matching 0.6.4's `.icons` exclusion: a project can't reach for the same knob to narrow what a bundle archives.

Also fixes `source_bundle.pdf`'s running header naming the wrong file: a file's own last page could show the *next* file's name instead of its own, because the invisible marker that sets the header text had no page break of its own - only the following content did - so it rendered on the tail end of the previous file's last page. The break now moves onto the marker itself, so the string-set and the page it applies to always agree.

1.79 0.6.4 (2026-07-21)

`prodockit.pdf.source_bundle` now always excludes any directory literally named `.icons` (e.g. a `custom_icons` directory, per `pymdownx.emoji`'s own convention) from `source_bundle.pdf`, regardless of `.gitignore` - found via `prodockit-template`, whose own vendored icon packs (`overrides/.icons/bootstrap`, `overrides/.icons/gitlab` - together ~2,500 unused SVGs) turned a source bundle meant to hold a student's own report content into a 3,000-page dump of unreferenced vendor assets. `.gitignore` alone can't fix this, since these directories are typically *tracked* (needed for the site/PDF to build at all) - deliberately not a project-configurable setting, so a project can't reach for the same knob to exclude something that should actually be archived.

1.80 0.6.3 (2026-07-20)

Bug fixes found by an in-depth test-coverage review of the extensions and PDF pipeline test suites - each paired with a new regression test.

- Fixed `prodockit pdf`'s CLI command showing a raw, unhandled traceback instead of a clean `Error: ...` message when `pdf_source_bundle` was enabled and the underlying `git`/`weasyprint` invocation failed - `SourceBundleError` wasn't in the CLI's caught exception tuple.
- Fixed `prodockit.headings` numbering a skipped heading level (e.g. h1 followed directly by h3, or a document starting below h1) with a literal "0" segment (e.g. "1.0.1") - a shallower level with no heading of its own yet is now treated as an implicit first one instead.
- Fixed `prodockit.pdf.mermaid` letting an uncaught `OSError`/`PermissionError` (e.g. a non-executable `mmdc` binary) escape instead of failing just that one diagram gracefully.
- Fixed `prodockit.pdf.source_bundle` crashing the whole bundle build on a file that's valid UTF-8 in the first 8 KiB sniffed to decide "is this text?" but not further in - now skipped like any other binary file instead.
- Fixed `prodockit.pdf`'s generated Lua filter producing broken syntax if a configured math/tex2svg path contained a quote or backslash - both are now escaped.
- Fixed `prodockit.pdf.build_pdf()` having no timeout on the underlying `pandoc`/WeasyPrint invocation, so a hang (e.g. a pathological CSS layout) could block the whole build indefinitely - added a `pandoc_timeout` parameter (default 30 minutes).
- Fixed a back-of-book index term nested more than three levels deep rendering with no extra indent at all, since the generated CSS only defines an indent step up to level 3 - now clamped to the deepest available indent instead.
- Substantially expanded test coverage across the extensions and PDF pipeline test suites (shared registries, cross-page linking, malformed input, table/index edge cases, icon/rotation/CSS edge cases) - no other functional changes.

1.81 0.6.2 (2026-07-20)

- Docs: fixed a real bug found by checking the live site after 0.6.1 - four spots in `docs/extensions/index-terms.md` / `docs/pdf.md` (plus two more in this same changelog) tried to show the code-styled `\index{}` syntax as literal example text using *inline* backticks around a hierarchical, code-styled path. Confirmed directly this doesn't work the way it does for the plain syntax - the code-styled pattern has to run before Python-Markdown's own backtick handling (see 0.6.0's own entry below), so inline backticks don't protect it, and the live site was rendering a raw internal Python-Markdown placeholder string instead of the intended literal text. Moved each one to a fenced code block (already documented as the safe way to show this syntax) or reworded to avoid the literal example entirely.
- No functional changes.

1.82 0.6.1 (2026-07-20)

- Docs: `prodockit.index` (new in 0.6.0) was missing from `README.md` - and so from PyPI's own project page - entirely: added it to the "Status" line and the extensions table, and mentioned the generated index alongside `prodockit.pdf`'s other PDF-only features. Also added it to `pyproject.toml`'s own `description` (PyPI's summary line) and `src/prodockit/__init__.py`'s module docstring, both of which had the same gap.
- No functional changes.

1.83 0.6.0 (2026-07-20)

- New `prodockit.index` extension: mark a term inline with `\index{Term}` for a traditional, PDF-only back-of-book index (browser/Ctrl-F search covers this on the live website, so there's no equivalent there) - the term displays inline exactly as written and is marked for indexing in one go, no separate "definition" step. Needed its own extension rather than the usual `attr_list` marker convention every other prodockit extension uses - confirmed directly plain `attr_list` can't wrap arbitrary inline text in a span on its own.
    - **Sub-entries**: `\index{Parent!Child!Grandchild}` (up to three levels deep in practice, matching LaTeX `makeidx`'s own `\index{primary!secondary!tertiary}` convention) nests related entries together instead of listing every term flat.
    - **Code-styled terms**: backticks around the last segment - or, combined with sub-entries, around just the last segment of a hierarchical path - mark a command/code term: it displays inline in a real `<code>` element, and the generated index entry renders the same way.
    - A term can be a markdown link or contain nested emphasis/code - confirmed directly neither needs special handling, since a term isn't exempted from Python-Markdown's own later inline-pattern passes the way `\ref{id}` / `\cite{id}` / `\gls{id}` are.

- New `prodockit.pdf.index`: the two-pass build (a term's own page number can only be known once WeasyPrint has already laid the PDF out once), configured by `prodockit.index`'s `include` and `title` settings - a traditional, two-column, letter-headed index page (matching this project's own cover page hero graphic colour), alphabetised ignoring leading punctuation (so `--set-upstream option` files under "S", not a separate symbols section), with consecutive pages collapsed into an en-dash range (`67–70`). Requires the new optional `pymupdf` dependency - `pip install prodockit[index]`.
- Fixed a real bug found while writing tests: code-styling a non-last segment of a hierarchical term - never a supported combination, but this shouldn't have corrupted anything either - used to leak a raw Python-Markdown internal stash placeholder into the generated index instead of failing gracefully - a real rendered PDF would have shown a nonsense category label instead of "Git".

1.84 0.5.0 (2026-07-19)

- New `prodockit.pdf.source_bundle`: bundles every text/source file `.gitignore` doesn't exclude into a separate `source_bundle.pdf` at a project's own top-level directory - 8pt Courier, wrapped lines, each file starting its own page, a running header (`site_name` on the left, that page's own file path on the right), and a "Page N of M" footer. Off by default; set `pdf_source_bundle = true` under `[project.extra]` to turn it on. Independent of the rest of `prodockit.pdf` - there's no Markdown involved, so it skips Pandoc entirely and hands a small, self-contained HTML document straight to WeasyPrint. File discovery shells out to `git ls-files --cached --others --exclude-standard` rather than reimplementing `.gitignore`'s own matching rules; text/ binary filtering is content-based, not by file extension.
- Docs: this site's own header now shows a PDF download icon next to "view" (an `overrides/partials/actions.html` override, linking to that page's own per-page PDF) instead of a "Download this page as PDF" text link at the top of the page - removed from every page that had one. Since the new icon is template markup rather than Markdown content, it also no longer shows up inside the PDF itself (no `.web-only` CSS trick needed, unlike the link it replaces).

1.85 0.4.2 (2026-07-19)

- Docs: matched more of this site's own theme config to prodockit-userguide's - the header's repo link now shows the actual GitHub logo instead of Zensical's default Git icon; the "View source of this page" button now shows an eye icon instead of a generic file icon; every admonition (e.g. the "tip" callout in `citations.md`) now uses the same custom FontAwesome icon set userguide uses instead of Zensical's own bundled defaults - this also feeds into `prodockit.pdf`'s own admonition icons, so PDF output picks it up too;

added the matching theme features userguide already had (`content.tabs.link` in particular actually affects this project's own tabbed content); and swapped the palette toggle icons to match userguide's own light/dark convention.
- No functional (Python package) changes.

1.86 0.4.1 (2026-07-18)

- Docs: reworked this site's own chrome to match prodockit-userguide's - a new split hero cover page ("Home"), reusing that project's own logo/ favicon/ illustration assets; top-level nav moved to a top tab bar with the right-hand page TOC merged into the left sidebar instead; the previous cover page's own prose moved to a new "Introduction" page.
- Fixed a real bug found along the way: `zensical.toml`'s own `copyright` was a triple-quoted, multi-line TOML string - `prodockit.pdf` substitutes it verbatim into a CSS `content: "..."` string for the PDF's running footer, and the embedded newline silently broke that declaration, dropping the whole footer with no error (this site's own PDF footer had no copyright text at all). Fixed to a single-line string, and switched `&copy;` to a literal `©` character - a CSS content string doesn't decode HTML entities either.
- Fixed the deploy workflow missing a per-page PDF build step for the new Introduction page (its own "Download this page as PDF" link 404'd), and that page's leftover PDF link (still the old cover page's, pointing at the whole-site PDF) to the same per-page convention every other content page already uses.
- No functional (Python package) changes.

1.87 0.4.0 (2026-07-18)

- New `pdf_double_sided` option: a duplex-printing layout for book/ handbook-style documents printed and bound on both sides. Verso (left-hand) and recto (right-hand) pages mirror their header/footer content and page margins (new `pdf_margin_inner`/`pdf_margin_outer`, replacing `pdf_margin_left`/`_right` in this mode) via CSS Paged Media's `@page :left`/`:right` selectors - chapter title and page number always on the outer, fore-edge corner; site name and copyright always on the inner, spine-side corner, whichever physical side that is for a given page. Every numbered heading now starts its own recto page (`break-before: recto`, auto-inserting a blank page as needed - confirmed directly this needs no Python-side page-counting logic at all), and a `prodockit-table-rotated` landscape page's own rotation direction now alternates by its final page position (270 degrees on recto, 90 on verso - the spine sits on the opposite physical side either way).
- New `recto_title` front matter key: overrides a page's own running header text with a shorter title, from the *next* page onward (the heading's own page still shows its full title - confirmed directly this is a consequence of CSS `string()`'s "first value on this page wins" default policy) - useful for a

chapter title too long to fit comfortably in the header, with or without `pdf_double_sided`.
- Off by default: a single-sided build is completely unchanged.

1.88 0.3.1 (2026-07-18)

- Docs: renamed `glossary.md`'s heading to "Acronyms and Glossary" and `citations.md`'s to "Citations or References" (and their matching nav labels); added a flow diagram to `bibliography.md`'s Requirements section (and fixed a real, unrelated gap found along the way - this docs site had no Mermaid `custom_fences` config at all, so a plain ```` ```mermaid ```` fence never rendered as a diagram anywhere on the site); switched the citation-style example to `harvard-cite-them-right.csl`; added an admonition pointing from `prodockit.citations` to `prodockit.bibliography`; and noted `prodockit.bibliography`'s own independent Pandoc invocation in `prodockit.pdf`'s "Limitations and workarounds".
- Docs: updated `README.md` (and so PyPI's own project page description) to include `prodockit.tables` / `prodockit.bibliography`, and to mention sideways tables/ `.web-only` / `.pdf-only` under PDF generation - it had gone stale since both extensions shipped in 0.3.0.
- No functional changes.

1.89 0.3.0 (2026-07-18)

- New `prodockit.bibliography` extension: an alternative to `prodockit.citations` for a `.bib`-backed reference list instead of a hand-authored one. Define sources in a BibTeX/BibLaTeX `.bib` file, cite them with the same `\cite{id}` syntax, and get the resolved citation text and a full, auto-generated reference list formatted in any Citation Style Language (CSL) style (APA, IEEE, Harvard, ...) via Pandoc's own `--citeproc` - confirmed directly against real Pandoc output rather than reimplementing citation formatting, and rejected an actual LaTeX/biblatex toolchain as a new hard dependency along the way. Makes `pandoc` a required dependency for this extension specifically, including for a website-only build with no PDF. New `docs/extensions/bibliography.md` includes a "References and Bibliography" comparison of this, `prodockit.citations`, and what `prodockit-template` / `prodockit-userguide` currently do.
- New sideways (90-degree anticlockwise) tables in the PDF: wrap a table and its own caption in `<div class="prodockit-table-rotated" markdown="1">` to print it on its own landscape-sized page(s), spanning multiple pages with a repeated heading row exactly like any other table. Confirmed directly that a CSS `transform: rotate()` doesn't work for this (clips the table to one page and loses its heading row) - the actual rotation is applied afterwards via a `/Rotate` post-process on the finished PDF (new `prodockit.pdf.rotate` module, new `pypdf` dependency).
- `.web-only` content is now hidden in every PDF build automatically, via `prodockit.pdf.css`'s own always-included stylesheet - no project-side CSS

needed any more. `.pdf-only` is documented as a one-line, centrally-sourced snippet instead (`prodockit` has no way to reach into a project's own website stylesheet), in a new "Web-only / PDF-only content" section in the PDF generation docs.

1.90 0.2.0 (2026-07-18)

- New `prodockit.tables` extension: gives a table column a percentage or fixed width via a `width` attribute already attachable to a header cell with `attr_list` - no new syntax. Column-width distribution beyond what's explicitly given is left to CSS's own `table-layout: fixed` algorithm rather than computed in Python. Ships with the matching CSS in `prodockit.pdf`'s generated stylesheet, and documents the equivalent rule a project's own website theme needs (see the new Tables docs page).
- New `prodockit pdf --markdown-file` / `-m` option: builds a PDF from a single markdown file instead of the whole `nav`, using the same `zensical.toml` settings as a full build.
- `prodockit.pdf`'s generated table CSS now draws a full grey 0.5pt grid - outer border and internal row *and* column lines (there was previously no line between columns at all) - and reads a project's own `extra_css` (from `zensical.toml`), so a project-specific `@media print` rule (e.g. hiding a website-only "Download PDF" link/button) also applies in the PDF.
- `prodockit.citations`: a resolved `\cite{id}` link now always gets `class="prodockit-cite-resolved"` (previously no class at all), matching `prodockit.refs` / `prodockit.glossary`'s existing convention of a stable class for both the resolved and unresolved case.
- Docs: added a "CSS hooks" section to `refs.md` / `citations.md` / `glossary.md` (`headings.md` already had one), documenting every class/attribute each extension itself emits; replaced the docs site's "edit this page" link with "view this page" (a `content.action.view` link to the raw source rather than a GitHub edit form); added a whole-site PDF download button on the front page and a per-page download link on every other page, both built via the new `--markdown-file` option above.
- Fixed `prodockit.__version__` reporting a stale `"0.10.0"` (left over from before the `zendoc` → `prodockit` rename) instead of matching this package's actual, `pyproject.toml`-declared version.

1.91 0.1.1 (2026-07-17)

- Docs: reworded the package intro on the docs site and README (dropped the `pymdown-extensions` comparison, added a mention of the website macros and a one-line "kit for professional documentation" summary) - no functional changes.

1.92 0.10.0 (2026-07-15)

- New `prodockit.zensical_macros`: Jinja variables/macros for Zensical's own macros plugin - `54,092` (site-wide prose word count, excluding the cover page and any page flagged `exclude_from_word_count: true`), `https://github.com/buckwem/prodockit-extensions` (git-detected repository URL), `prodockit`, and `heading_counter_reset(page)` / `reference_style()` / `acronym_style()` / `glossary_style()` macros. Add it alongside a project's own `macros.py` via `zensical.toml`'s `modules = ["prodockit.zensical_macros"]` - or use it alone if the project has no macros of its own.
- New `prodockit.wordcount`: the generic prose word-count utility (`count_words()` / `compute_word_count()`) behind both `prodockit.pdf`'s `{WORDCOUNT}`-style cover-page use and `prodockit.zensical_macros`' `54,092` - previously duplicated independently by each downstream project needing both.
- New `prodockit.settings`: `flatten_nav()`, `heading_numbering_enabled()`, and `reference_style_values()` - the `project.extra.*` reading shared by `prodockit.pdf.config` and `prodockit.zensical_macros`, so the two agree on one set of fallback defaults instead of each hand-maintaining its own copy. `prodockit.pdf.config.build_pdf_from_zensical_config()` now uses these too (previously inlined), and its `pdf_math_dir` setting is now created automatically if configured to a directory that doesn't already exist (matching the auto-detected default's existing behaviour).

1.93 0.9.0 (2026-07-15)

- New `prodockit pdf` command: builds a complete PDF with no Python required, reading everything - nav, docs directory, fonts, page size, and all PDF-specific settings - from the project's own `zensical.toml`, the same way `zensical build` / `zensical serve` do. Installing `prodockit` now registers a `prodockit` console script (`pip install prodockit` is enough - no separate build script to write). See the new `prodockit.pdf.config` module (`build_pdf_from_zensical_config()`) for the config-to-`build_pdf()` orchestration this wraps: nav-tree flattening, per-page `is_appendix` front-matter detection, and auto-detection of a local `mmdc` (Mermaid) binary and MathJax `tex2svg` script, so a typical project needs no extra configuration beyond what it likely already has.
- `build_pdf()` gained `include_table_of_contents` / `table_of_contents_title` parameters (both used automatically by `prodockit pdf`): a generated table of contents is now inserted by default, right after a cover page if one is marked `is_index=True`, or at the very start otherwise.
- Rewrote the PDF generation docs page around the `prodockit pdf` command as the primary, and for most projects only necessary, way to use

`prodockit.pdf` - `build_pdf()` and the individual pipeline pieces are now documented as the advanced, scripting-your-own-pipeline path.

1.94 0.8.0 (2026-07-15)

- New `prodockit.pdf.build_pdf()`: a one-call convenience wrapper around the rest of `prodockit.pdf` - hand it a list of already-rendered pages (`prodockit.pdf.Page`) and where to write the PDF, and it fixes up each page's HTML, generates the Lua filter and CSS, concatenates everything, and runs `pandoc`/WeasyPrint for you. Takes `output_path` (the PDF's own destination path) plus font/page-size/margin/header-footer/reference- style/numbering/math parameters, all with sensible defaults. Raises the new `prodockit.pdf.PdfBuildError` (with the underlying `pandoc` exit code and stderr attached) if the build fails, rather than failing silently. `prodockit.pdf.html`/`.lua`/`.css`/`.icons`/`.mermaid` remain directly importable if you need more control over how the pieces fit together.
- Rewrote the PDF generation docs page around `build_pdf()` as the primary documented way to use `prodockit.pdf`, leading with a short, practical quick-start example rather than the implementation-level detail of how Pandoc/WeasyPrint's own quirks are worked around (that detail is still there, now further down, for anyone who wants it).

1.95 0.7.0 (2026-07-15)

- New `prodockit.pdf`: a Pandoc/WeasyPrint pipeline for building a standalone PDF from Zensical-rendered HTML - not a Python-Markdown extension (no `markdown.extensions` entry point), a plain function library, since a PDF build pipeline isn't a Markdown syntax extension:
  - `prodockit.pdf.html`: `fix_up_page_html()` and link/anchor/image helpers
  - fixes up one page's already-rendered HTML for Pandoc's own reader/writer quirks (attribute loss on `<p>`, raw `<svg>` not surviving the round trip to WeasyPrint, footnote/caption structural mismatches, cross-page link rewriting for a concatenated multi-page PDF, and more).
  - `prodockit.pdf.lua`: `build_lua_filter()` - chapter/appendix numbering, caption chapter-prefix numbering, tabbed-set reconstruction, and MathJax pre-rendering, generated as a parameterized Lua filter.
  - `prodockit.pdf.css`: `build_css()` - the compiled CSS a PDF needs on top of a project's own website stylesheet, including WeasyPrint-specific page-break tuning for headings, paragraphs, tables, code blocks, figures/captions, admonitions, and grid cards.
  - `prodockit.pdf.icons` / `prodockit.pdf.mermaid`: admonition icon resolution and Mermaid diagram pre-rendering, as standalone helpers.
- Fixed a real bug found while writing tests: the iframe→"Watch Video" admonition link builder stripped the video id from every single conversion (a

replace-then-split ordering removed the just-added `?v=...` too) - now produces a working YouTube watch link.
- No formal, versioned public API surface yet (see prodockit-extension#7) - import whatever's needed directly, the same informal way as the rest of this package.
- New dependency: `beautifulsoup4` (>= 4.12).
- Broadened the package's own description: prodockit is now framed as a family of extensions for Zensical needed for professional and academic documentation, rather than "Python-Markdown extensions" specifically - `prodockit.pdf` isn't one, and the framing was due to broaden anyway now that PDF generation is in scope alongside cross-references/citations/glossary.

1.96 0.6.0 (2026-07-14)

- `prodockit.headings`: new `numbering="continuous"` option (Zensical only) - `h1` numbering carries on from wherever the previous nav page left off, instead of restarting at 1 on every page. Fixes `\ref{id}` showing the wrong number for a heading on a different page (it previously always showed that page's own per-document number, not the number actually displayed on the page - see zendoc-template#89).
- New `appendix_attr` option (default `is_appendix`): a page whose front matter sets this flag is numbered with a letter instead - "A", "A.1", "A.1.1" - and doesn't consume a number from the numeric sequence, so later pages aren't left with a gap. Letters are assigned sequentially in nav order.
- New public `prodockit.headings.prescan(appendix_attr="is_appendix")` function: returns the same `(start_counts, appendix_letters)` pre-scan `HeadingsExtension` uses internally, for a consuming project's own build tooling (e.g. a template macro driving a presentational CSS counter-reset) to stay in sync automatically rather than re-deriving the same page-order/heading-count logic independently.

1.97 0.5.1 (2026-07-14)

- `prodockit.glossary`: a resolved `\gls{id}` now always renders with `class="prodockit-gls"` (previously it had no class at all), matching `prodockit.refs`' always-present base class. The unresolved case now renders `class="prodockit-gls prodockit-gls-unresolved"` (previously just `prodockit-gls-unresolved`, missing the base class), so a stylesheet has one stable hook (`.prodockit-gls`) regardless of resolution state, with `.prodockit-gls-unresolved` layered on top only when needed.

1.98 0.5.0 (2026-07-14)

- New `prodockit.glossary` extension: define a term once via `attr_list` (an id plus a `data-term` short display string), then insert it by id from anywhere

with `\gls{id}`, which resolves to the term's own text, linked to its definition - e.g. `\gls{css}` → `CSS`. Unlike `prodockit.citations`' `\cite{id}` (which generates new bracketed citation text), `\gls{id}` inserts the term's own registered text in place - closer to LaTeX's `glossaries` package.
- One shared `GlossaryRegistry` covers both acronym-style and glossary-style entries - they're the same kind of thing (an id with a short display text), so acronym and glossary pages can reference each other, or be referenced from any other page, with no special wiring.
- Supports forward references within a document, an `unresolved` marker (`?` by default) for an unknown id, and the same automatic Zensical cross-page registry sharing and nav pre-scan (for citing/using a term before its defining page has been converted) that `prodockit.citations` got in 0.4.0.
- Refactored the nav pre-scan logic (previously private to `prodockit.citations`) into a shared, generic `prodockit._zensical.preseed_attr_from_nav` helper, since `prodockit.glossary` needed the identical scan.

1.99 0.4.0 (2026-07-14)

Fixes found migrating a real multi-page site's references page to `prodockit.citations` for real - all discovered by actually building a real multi-page site, not just single-document tests:

- **Fixed a real correctness bug**: `prodockit.refs` / `prodockit.citations` were emitting a bare `#id` fragment for *every* resolved link, including a cross-page one - which only works by coincidence in a single concatenated PDF document, but 404s on an actual multi-page website (an `#id` fragment only navigates within the *current* page). Both now emit a real relative link (e.g. `references.md#id`, correctly adjusted for the citing page's own directory depth) when the target is on a different page, which Zensical already knows how to rewrite into the right clean URL - the same way a hand-typed cross-page Markdown link already works.
- New: `prodockit.citations` pre-scans every page in a Zensical build's nav for citation definitions before any page is actually converted, so citing a source *before* it's defined - the common case, since a references page is usually kept at the end of nav as an appendix - resolves correctly in a single `zensical build` pass, rather than only working from `zensical serve`'s live-reload. New `CitationRegistry.preseed()` method backs this; a real registration always supersedes a preseeded stub.
- `RefsExtension` gained a `source` option (mirroring `HeadingsExtension`'s), needed for the same-page-vs-cross-page link decision above.
- Fixed the nav pre-scan matching a citation-definition attr_list example shown literally inside a fenced code block in documentation - it now skips fenced content, the same protection `CitationDefTreeprocessor` already gets for free from the real Python-Markdown parser.

1.100 0.3.0 (2026-07-14)

- New `prodockit.citations` extension: define a source once via `attr_list` (an id plus a `data-cite-text` short display string), then cite it by key from anywhere with `\cite{id}` (or `\cite{id1,id2,...}` for multiple), auto-generating a bracketed, linked citation - `[Skoulikari, 2023]` - instead of hand-typing the link and text at every citation site.
- Supports forward references within a document, an `unresolved` marker (`?` by default) for an unknown key, and the same automatic Zensical cross-page registry sharing (with soft-fail on key collisions) that `prodockit.headings` / `prodockit.refs` got in 0.2.0.
- Auto-generating the references page's own listing from structured bibliographic data isn't built yet - see the extension's docs for the current scope.
- Fixed the `zensical.toml` installation examples in the docs: nested `[project.markdown_extensions.prodockit.headings]` tables don't work (Zensical only hoists the `pymdownx` / `zensical` namespaces that way) - the quoted-key form (`[project.markdown_extensions."prodockit.headings"]`) is required.

1.101 0.2.0 (2026-07-14)

- `prodockit.headings` / `prodockit.refs` now share their registry automatically under Zensical, without any explicit `registry` / `source` configuration: each extension detects Zensical's per-page rendering context and derives a stable `source` from the page's own path, fixing cross-page `\ref{id}` references not resolving.
- A heading id collision across two different sources, when detected via this automatic Zensical sharing, now logs a warning and keeps the first registration instead of raising `DuplicateIdError` - so two unrelated pages that happen to share a heading title (e.g. both have an "Overview" section) no longer break the build. Explicitly-shared registries (the manual multi-page pattern) still raise on a collision, unchanged.
- Fixed an extension-ordering bug: `prodockit.headings` and `prodockit.refs` now find and share each other's registry regardless of which order they're listed in - previously, only `prodockit.headings`-then-`prodockit.refs` worked reliably, and Zensical's own TOML-to-extension-list conversion doesn't preserve list order at all.

1.102 0.1.0 (2026-07-14)

Initial release.

- `prodockit.headings`: heading ids and hierarchical section numbering, backed by a shared `IdRegistry`.

- `prodockit.refs`: `\ref{id}` section cross-references, resolving to the target's current section number, including forward references within a document and across a shared registry.
- Documentation site built with Zensical, published at buckwem.github.io/prodockit-extension.