

How a $\text{\LaTeX}$ Document Becomes a PDF

Alex Rivera Jordan Lee Sam Okafor

Example University • Example Labs • Open Lab Workshop Series

1. From source to typeset boxes

A $\text{\LaTeX}$ file is plain text. The engine reads it character by character, turns it into *tokens*, expands macros and finally produces *boxes*: rectangles with a width, a height and a depth. Paragraphs are broken into lines by an optimal line-breaking algorithm, and pages are built by stacking boxes until the page is full.

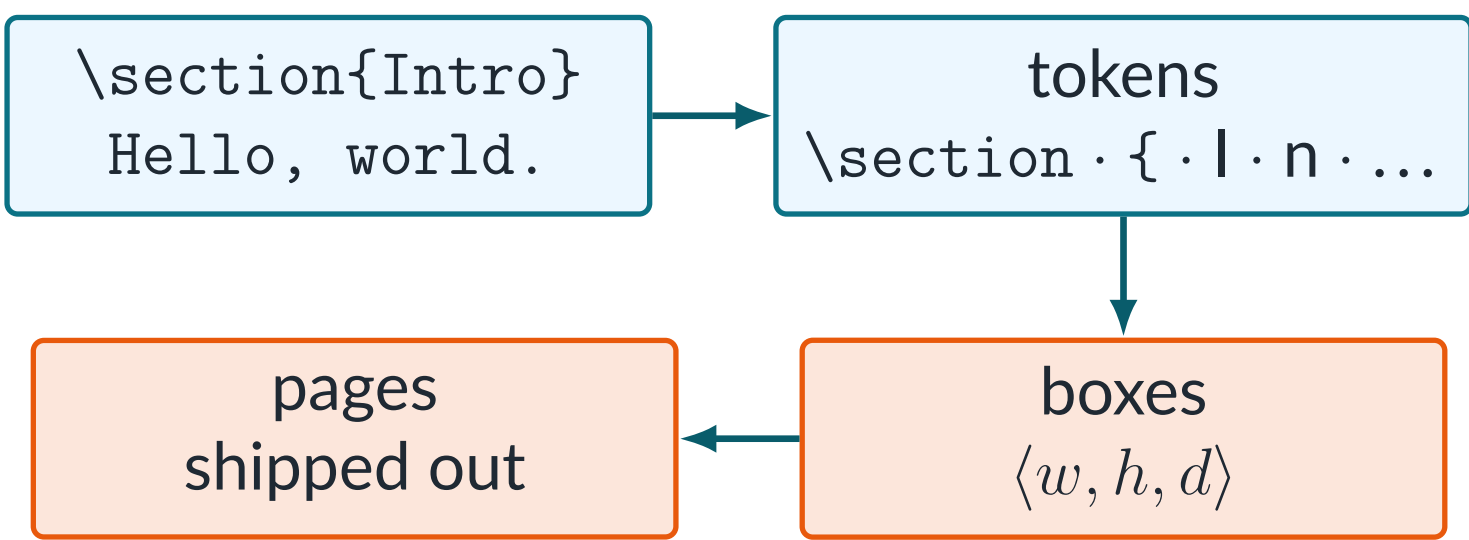


Fig. 1: Tokens become boxes, boxes become pages.

2. Three engines, one language

	pdfTeX	XeTeX	LuaTeX
Output	PDF	XDV → PDF	PDF
Input encoding	inputenc	UTF-8	UTF-8
Fonts	Type 1	OpenType	OpenType
Scripting	–	–	Lua
Start-up	fastest	fast	slowest

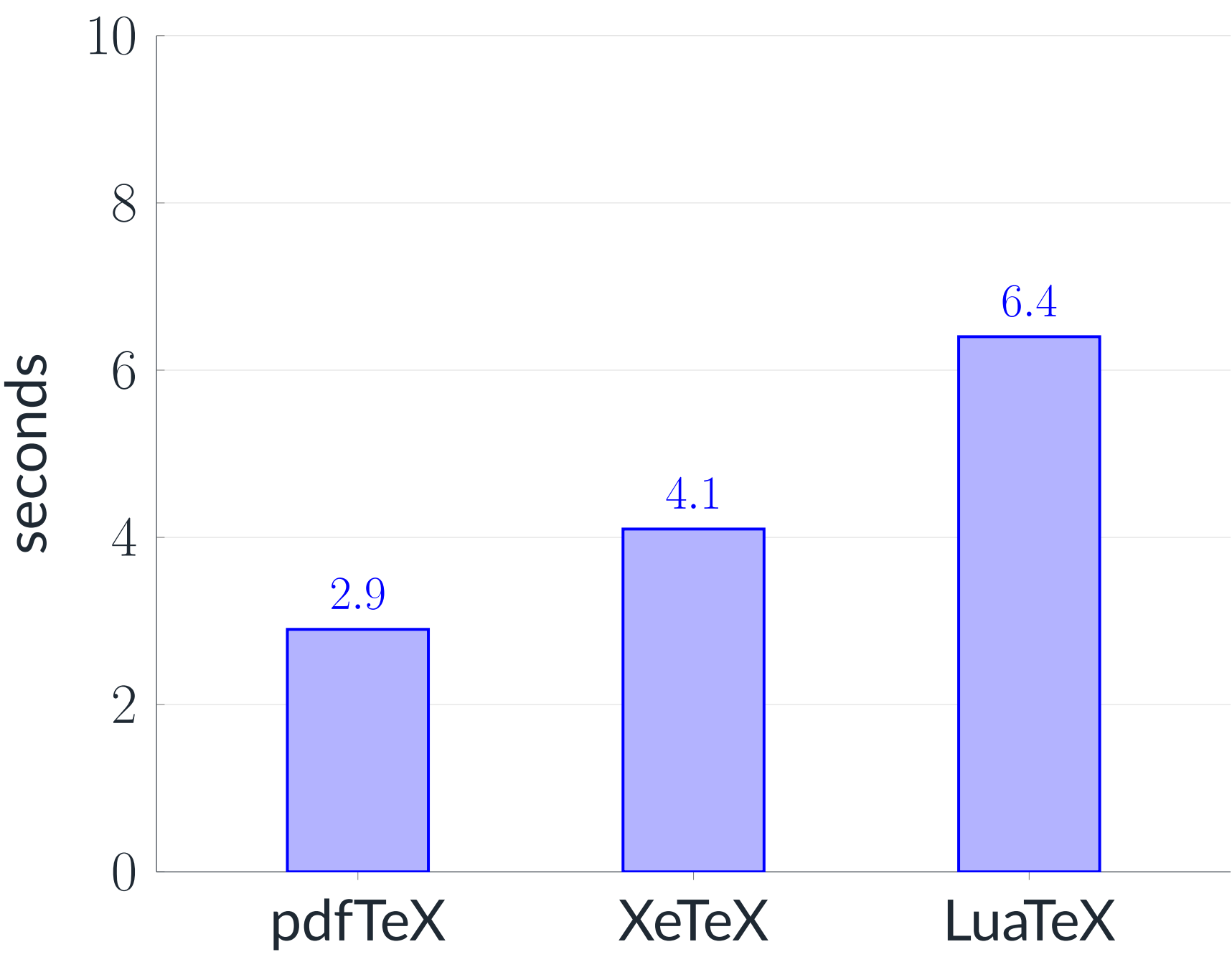


Fig. 2: Median compile time of an 8-page paper (seconds).

3. Fonts



pdfTeX embeds **Type 1** fonts described by metric files (`.tfm`); XeTeX and LuaTeX load **OpenType** fonts directly through fontspec. Every glyph used in the document is embedded in the PDF, which is why a 2-page paper can weigh 150 kB.

Glossary

Format file A pre-digested dump of the $\text{\LaTeX}$ kernel that the engine loads at start-up instead of re-reading thousands of macros.

.aux Auxiliary file where labels, citations and counters are written for the next pass.

.bbl Formatted bibliography produced by `bibtex` or `biber` from your `.bib` entries.

Glyph The drawn shape of a character inside a font; only the glyphs you use are embedded.

4. The build pipeline

Most documents need **more than one pass**: the first run writes labels, citations and the table of contents to auxiliary files, and later runs read them back. `latexmk` automates the loop and decides when to call `bibtex` or `biber`.

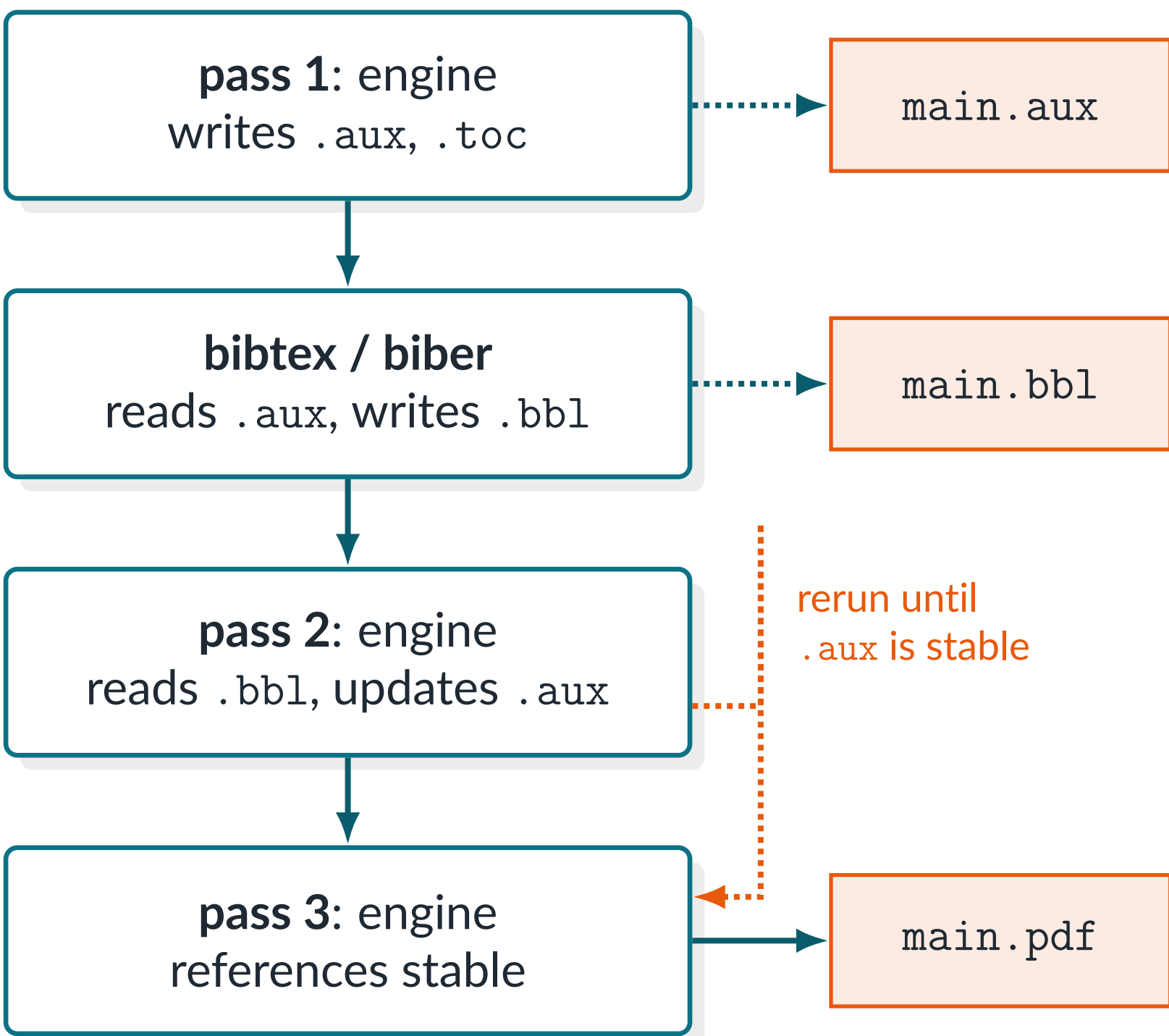


Fig. 3: A typical multi-pass build driven by `latexmk`.

5. Where the time goes

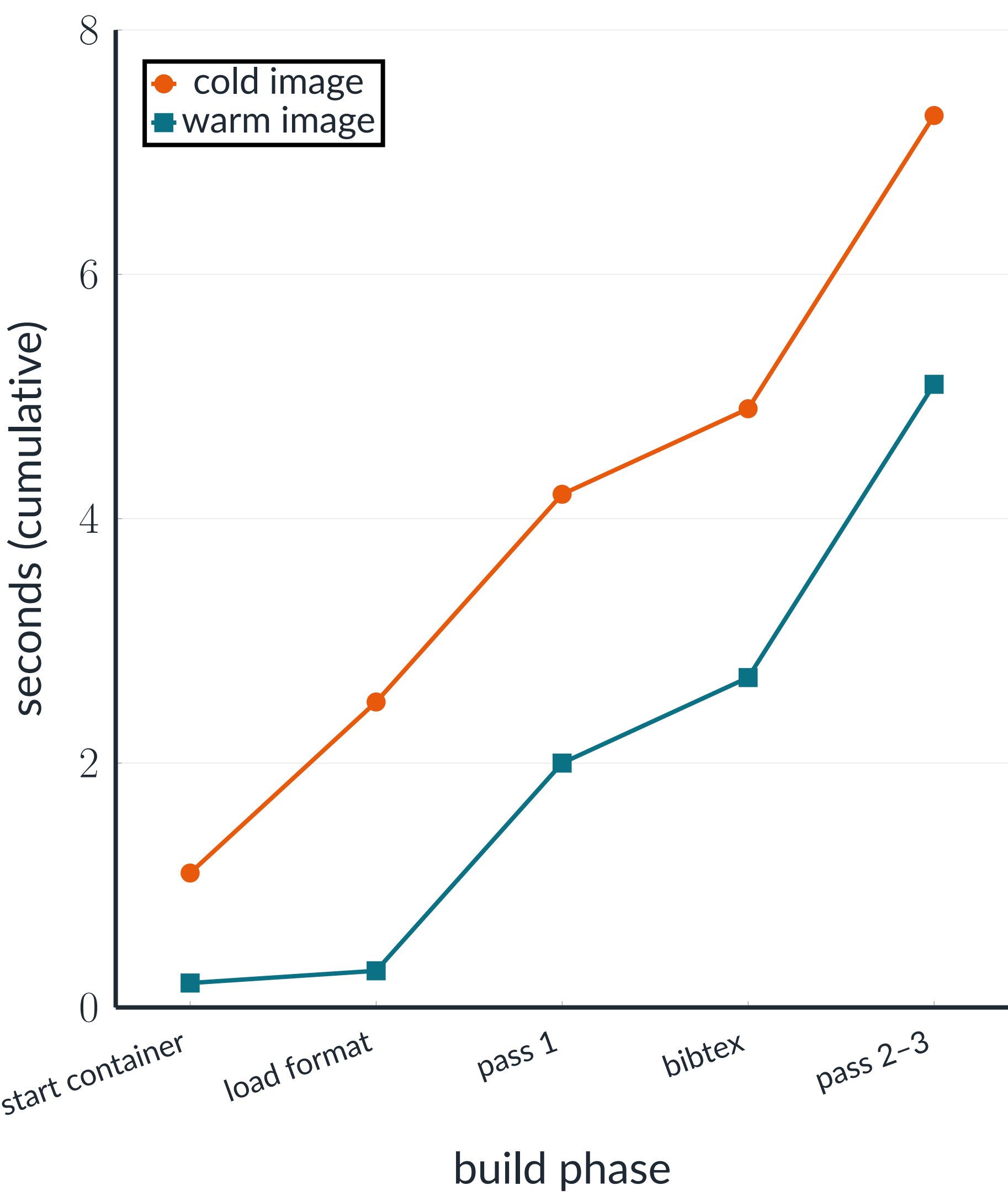


Fig. 4: Cumulative time of a cold build inside a container.

Pre-generating the format files and the font database inside the image removes most of the fixed start-up cost. The engine passes themselves take about the same time either way.

Try it yourself



One command is enough on any machine with $\text{\TeX}$ Live installed. It runs the engine as many times as needed and calls the bibliography tool for you:

```
$ latexmk -pdf -interaction=nonstopmode main.tex
$ latexmk -lualatex cv.tex # OpenType fonts
$ latexmk -c # remove .aux, .log, .bbl
```

6. Inside the PDF



The final file is a tree of **objects**: a catalog, a page tree, one content stream per page, embedded font programs and, optionally, an outline, hyperlinks and metadata. pdfTeX compresses streams and can pack small objects into *object streams*, which is why the page count is not visible in a plain-text search.

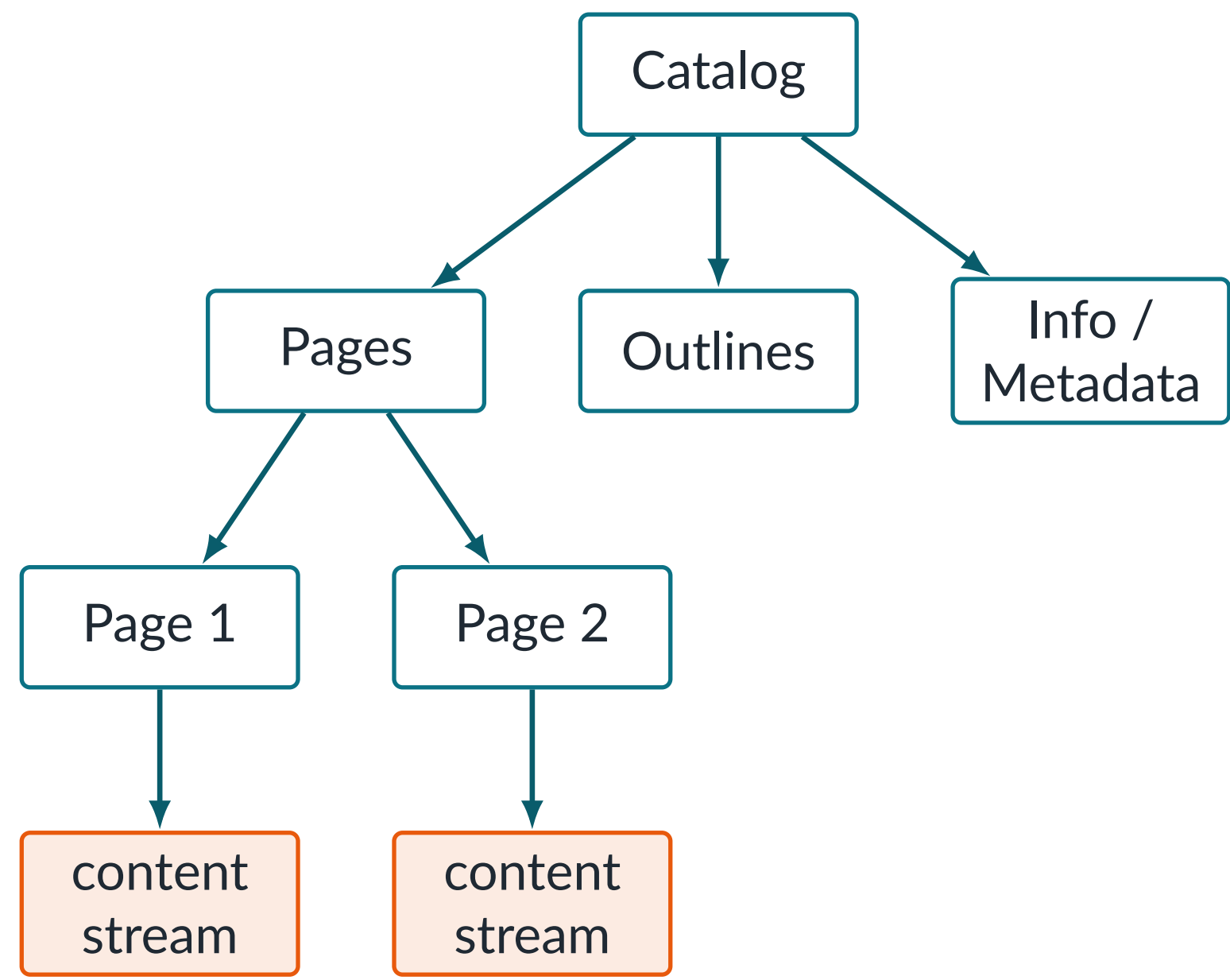
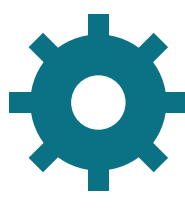


Fig. 5: Object graph of a minimal two-page PDF.

7. Running it in the cloud



A compile service treats the project as an **immutable bundle**: its hash identifies the output, so unchanged documents are never rebuilt. Each build runs in a throw-away container with no network, no shell escape and strict CPU, memory and time limits.

Safety checklist

- Unprivileged user, read-only root file system
- `--network none`, `seccomp allow-list`
- `shell_escape = f`, `openout_any = p`
- 2 GB memory, 2 CPUs, 64 processes, hard time-out

68%
cache hits

2.9 s
median build

0.3%
timeouts

Take-aways

1. $\text{\LaTeX}$ is a programming language whose output is boxes.
2. Multi-pass builds are normal; let `latexmk` drive them.
3. Fonts are embedded, so the PDF is self-contained.
4. In the cloud, start-up cost dominates short builds; cache aggressively and warm the image.

Poster template for the `latex-api` project. Built with `tikzposter` and `pgfplots`; all figures are drawn with TikZ and the data is fictional, for illustration only.