The Becoming: Building a Poetry Publishing Pipeline with Claude Code

2026

7 min read • 1,586 words

Themes: Consciousness Technology Programming Recursive Spiritual

My wife Sarah wrote a book of poetry. It's called The Becoming, and it's available now on Amazon in both paperback and Kindle editions. Every poem in it was written entirely by her—raw, honest, and unflinching. The AI part of this story isn't about the writing. It's about everything else.

The entire collection is also available for free at poemsbysarah.com. We believe poetry should be accessible. If you want to hold it in your hands, buy the book. If you just want to read it, the website is there for you.

But this essay isn't really about the poetry itself—Sarah's words speak for themselves. This is about how Claude Code and I built an entire publishing pipeline from scratch: extracting poems from raw manuscript files, curating a collection from over 120 poems, generating print-ready PDFs, building a static website, running sentiment analysis, and producing a book ready for print-on-demand—all through conversational programming with an AI.

The Raw Material

Sarah writes poetry the way most people journal. She doesn't organize it into files or folders. She writes in bursts, sometimes a dozen poems in a night, into whatever text file happens to be open. What I started with was two manuscript files— `manuscript.txt` at 125KB and `manuscript2.txt` at 61KB—containing around 120 poems with no consistent formatting, no metadata, and no separation beyond occasional lines of underscores.

The first task was extraction. Claude Code wrote a parser that detects two different manuscript formats—one separated by underscore lines, the other by blank-line patterns—and splits them into individual markdown files with clean titles:

```
MANUSCRIPT1_DIR = Path("poems/manuscript1")  # 64 poems
MANUSCRIPT2_DIR = Path("poems/manuscript2")  # 30 poems
EDITED_DIR = Path("poems/edited")            # 26 curated poems
```

Each poem becomes a simple markdown file. `01-bathtub-blues.md` starts with an H1 header and contains the poem body with line breaks preserved. Nothing fancy. The structure is the simplicity.

Curation with Claude

One hundred and twenty poems is too many for a collection. The question wasn't which poems were good—most of them were—but which ones belonged together, telling a single coherent story of transformation.

Claude helped us read through every poem and identify a narrative arc. We landed on four acts:

1. **The Hollow** — Despair and emptiness. The poems that live at rock bottom.
2. **The Reckoning** — Self-awareness beginning to emerge. Looking at the damage honestly.
3. **The Awakening** — Transformation in progress. The first signs that change is real.

4. **The Bloom** — Love and wholeness. Not perfection, but arrival.

Twenty-six poems made the cut.

> This is the editor's true work—signal extraction from noise. The 94 poems that didn't make it weren't bad; they were necessary context, like kernels that had to be removed to reveal the fruit. AI can analyze but humans must decide what serves the story.

Claude also helped identify where long poems needed to breathe—splitting "Grappling with Reality" into separate pieces, breaking "Self Reflections" so that "Patience" could stand on its own. Not rewriting. Just finding the natural seams Sarah had already laid down.

The Document Pipeline

This is where it gets interesting from a technical perspective. The entire publishing pipeline is thirteen Python scripts, roughly 6,000 lines of code, almost all of it written through conversation with Claude Code. I'd describe what I wanted, Claude would write the code, I'd run it, we'd iterate.

PDF Generation with WeasyPrint

The core of the print pipeline is WeasyPrint—a Python library that converts HTML and CSS into PDF.

> HTML and CSS as a typesetting language feels counterintuitive until you realize it solves the problem of preserving semantic structure while specifying layout—you're describing what things are, not just how they look.

This turns out to be a remarkably good approach to book typesetting. You write your layout in CSS, your content in HTML, and WeasyPrint handles pagination, margins, and page breaks.

The print-ready interior targets a 5.5" × 8.5" US Trade paperback format:

```
TRIM_WIDTH = "5.5in"
TRIM_HEIGHT = "8.5in"
MARGIN_OUTSIDE = "0.75in"
MARGIN_GUTTER = "0.875in"  # inside margin, wider for binding
```

Each poem gets a dedicated page. Act titles get their own spreads with epigraphs. The front matter includes a title page, dedication ("For those still becoming."), and an introduction. CSS handles the typography—serif fonts, generous line-height, ornamental section breaks using unicode characters like ❧ and ◇.

The pipeline also generates a cover, a full jacket spread with spine, and an EPUB —all from the same markdown source files.

The WeasyPrint Dance on macOS

One thing Claude helped solve that would have taken me hours: WeasyPrint on macOS requires Pango, Cairo, and GLib from Homebrew, but Python can't find them without the right `DYLD_LIBRARY_PATH`. Claude wrote a setup module that auto-detects Apple Silicon vs. Intel paths and configures the environment before WeasyPrint imports:

```python
def setup_weasyprint():
    """Configure macOS library paths for WeasyPrint."""
    lib_dir = Path("/opt/homebrew/lib")  # Apple Silicon
    if not lib_dir.exists():
        lib_dir = Path("/usr/local/lib")  # Intel
    os.environ["DYLD_LIBRARY_PATH"] = str(lib_dir)
    from weasyprint import HTML, CSS
    return HTML, CSS
```

Small thing, but it's the kind of yak-shaving that kills momentum. Claude just handled it.

Sentiment Analysis and the Healing Arc

This is the part that surprised me. Claude built an analysis suite using TextBlob and NLTK that does more than basic sentiment scoring—it tracks the emotional trajectory of the entire collection.

The tool detects six distinct narrative voices across Sarah's poetry:

> This mirrors what I've found true in my own writing on consciousness—we're never singular. We contain multitudes. The mythology of the unified self is comforting but false; we're orchestras playing the same body. Poetry knows this. Sentiment analysis learning to detect it feels like technology finally catching up to what humans have always known.

a child voice (words like scared, little, alone), a protector (strong, safe, fight), an angry voice, a sensual voice, a spiritual voice, and an analytical voice. Many poems contain three or more of these voices simultaneously.

But the most striking finding was the healing trajectory. The analysis compares trauma-associated language (pain, wound, broken, fear, shame) against healing language (bloom, rise, forgive, grace) across the collection:

```
First half:  51% healing language
Second half: 70% healing language
Shift:       +19%
```

The numbers confirmed what we felt reading the poems in sequence—there's a real, measurable arc from darkness to light. The darkest moment is "Lost in Thought" at -0.31 sentiment. The brightest is "The Light of Morning" at +0.42. That's a 0.73 emotional range across 26 poems and 3,206 words.

We turned all of this into an interactive appendix at poemsbysarah.com/appendix.html, with Plotly charts showing the emotional arc, healing trajectory, and trauma-to-healing ratios by act. Data visualization applied to the interior landscape of a human being.

The Static Website

The website is generated by another script—889 lines that produce a complete static site with Tufte CSS styling.

> Edward Tufte's sidenote system, adapted for web, works because it mirrors how humans actually read and think—the main narrative flows, but peripheral insights live in the margins, waiting to be discovered. It's a humane design approach that respects different kinds of attention.

Each poem gets its own page with keyboard navigation (arrow keys to move between poems), metadata, and editorial sidenotes where poems were split or renamed from the original manuscripts.

The index organizes everything by act, with thematic epigraphs. There's no JavaScript framework, no build system, no dependencies at runtime. Just HTML, CSS, and an SVG flourish. It loads instantly and will work forever.

What Claude Code Actually Did

I want to be specific about the division of labor, because I think it matters.

Sarah wrote every poem. The words, the images, the emotional truth—that's entirely hers. Claude never touched the poetry itself.

We made every creative and editorial decision. Which poems to include, how to order them, where to split long pieces, what the acts should be called, what the book should look like. Claude offered suggestions and analysis, but the curation was a human process.

Claude Code wrote the infrastructure. The extraction scripts, the PDF generation pipeline, the website generator, the sentiment analysis tools, the cover design, the EPUB builder.

> This is the conversational programming model I've been writing about—where you think out loud and AI implements your thinking in real time. The friction between human intention and code execution collapses. You can iterate on metaphorical descriptions and watch them become concrete systems.

I described what I wanted in conversation, Claude wrote the code, I ran it and we iterated. Thirteen scripts, 6,000 lines of Python, built over the course of a few sessions.

This is the model of human-AI collaboration I keep coming back to. The human provides the what and the why. The AI provides the how. Neither could have done this alone—I don't have the patience to hand-code 983 lines of print-ready PDF generation CSS, and Claude doesn't have the lived experience to know which poems belong together.

The Recursive Thing

There's something fitting about using AI to build the infrastructure that publishes deeply human poetry.

> The recursive pattern shows itself everywhere: Sarah becomes through writing. We become through publishing. Technology becomes more human by serving human becoming. Each layer transforms itself in service of the next.

Sarah's poems are about becoming—the painful, nonlinear process of turning into who you actually are. The technical pipeline is its own kind of becoming: raw text files transforming into a bound book through a series of conversions, each one preserving the essence while changing the form.

Markdown to HTML. HTML to PDF. PDF to printed page. Printed page to someone's hands. Words to feeling.

The technology disappears. What remains is the poetry.

The Becoming is available now on Amazon. The full collection is free to read at poemsbysarah.com.