DOCUMENT360

Webinar On

From Writer to Tool Builder: AI Coding Agents for Technical Documentation

Speaker

Rebecca Alves

Senior Technical Writer

Indeed

Overview

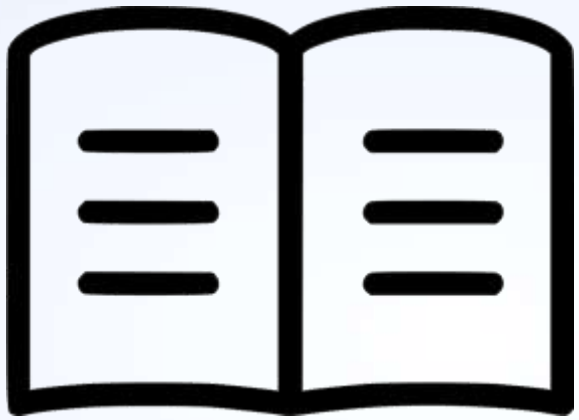
Challenge

- You see automation opportunities
- But you're a technical writer, not a SWE

Solution

- AI coding agents = collaboration partners
- Build automation without deep programming expertise

Overview



1. Why docs-as-code
2. Automation projects
3. Key principles and next steps

Prerequisites

Environment

- Docs-as-code setup (Markdown, GitHub)
- Command line access
- Access to an AI coding tool (free or paid tiers)

Skills

- Git basics
- Edit text files & run commands
- Clear communication skills

Optional

- API access for tools
- Python or JavaScript familiarity (not required)
- Local dev environment

My setup

Cursor

- AI-native code editor
- Real-time AI collab
- Conversations in same interface
- Version control integration

Claude Sonnet

- Writes and explains code
- Understands context & asks questions
- Explains decisions & reasoning
- Iterates on feedback

DOCUMENT360

My setup

Why this combo works

- Like pair programming
- Describe goals, AI implements
- Learn as you build

DOCUMENT360

Docs-as-Code enables automation

Programmable pipelines

- GitHub Actions, webhooks, CI/CD

Structured content

- Markdown = parseable, queryable, transformable

Git/GitHub

- Audit trails, collaborative iteration, rollback safety

Barriers for technical writers

Limited programming expertise

- "I don't know Python/JavaScript"

No dedicated engineering support

- "We don't have developers on our team"

Fear of breaking production

- "What if my code breaks the docs site?"

From writing code to collaborating

- Write production-ready code from natural language descriptions
- Explain existing codebases in plain English
- Debug & fix errors iteratively
- Adapt to feedback and refinement

From writing code to collaborating

Leverage skills you already have

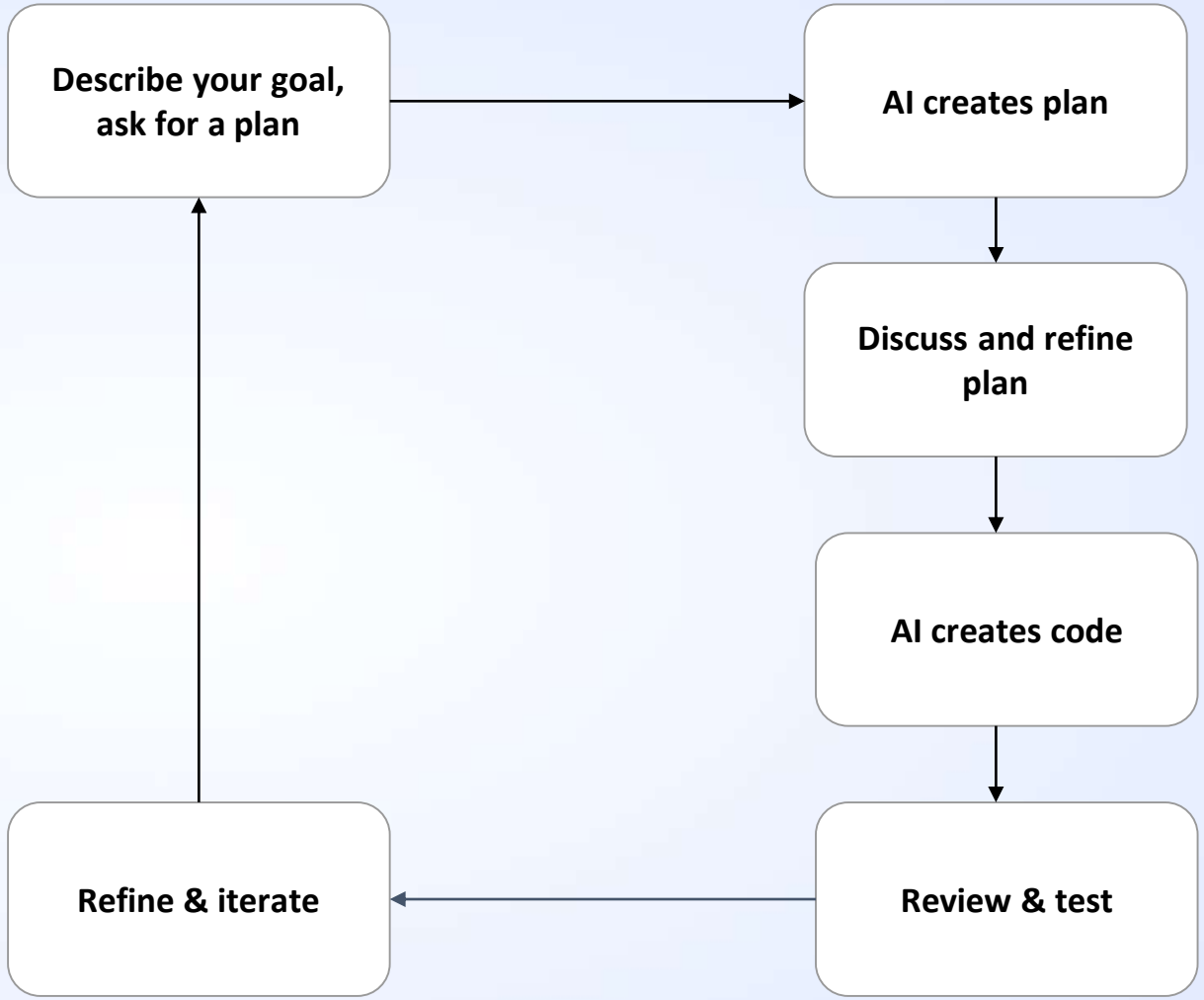
- Clear communication
- Systematic thinking
- Edge case identification
- Iterative refinement

From writing code to collaborating

Best practice -> plan first, then implement

- Plans are faster to iterate than code
- Catch issues before implementation
- More efficient collaboration

From writing code to collaborating



Release notes automation

DOCUMENT360

Release notes automation

Manual process

- Review merged pull requests
- Read commit messages and change descriptions
- Categorize changes (features, fixes, enhancements, docs)
- Format consistently for publication
- Add context and links for readers

Release notes automation

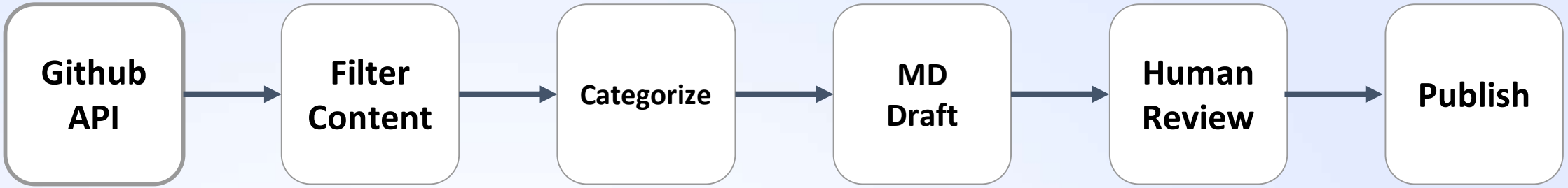
Manual process

- Significant effort for curation and formatting
- Easy to miss important changes buried in commit messages
- Inconsistent categorization across different curators
- Repetitive, structured work

Release notes automation

- **Best practice -> Before automating, write down manual process**
- Automation replicates your workflow
- Dramatically reduces manual curation effort while maintaining quality

Release notes automation



Query PRs by date range

Exclude noise, WIP, internal

Group by type

Format output and add links

Validate & add context

Live docs

DOCUMENT360

Release notes automation

Key lessons

- ✓ Write manual workflow before automating
- ✓ AI augments, humans review for accuracy
- ✓ Iterate prompts based on real results
- ✓ Start simple
- ✗ First attempt included internal changes (criteria unclear)
- ✗ Categories too vague (standards not documented)
- ✗ Formatting inconsistent (format specs missing)

Translation sync

Translation sync

Scenario

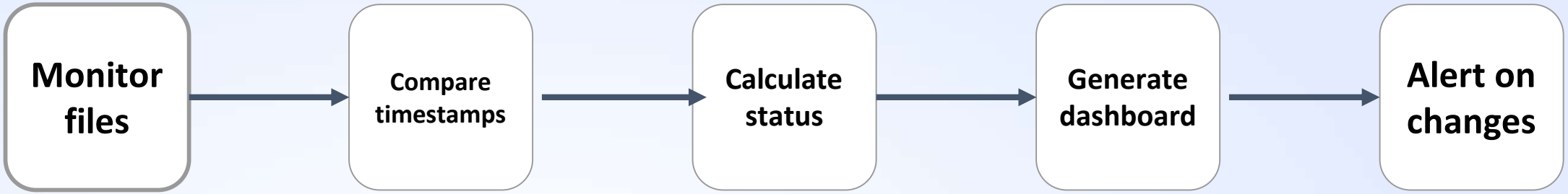
- Docs published in multiple languages
- Source language constantly updated
- What's translated? What's outdated? What's missing?

Translation sync

Manual process

- Open source file, check modification date
- Open corresponding translated file, check its date
- If source is newer, add to translation queue
- Repeat for every single file
- Track in spreadsheets

Translation sync



Script scans all source doc files

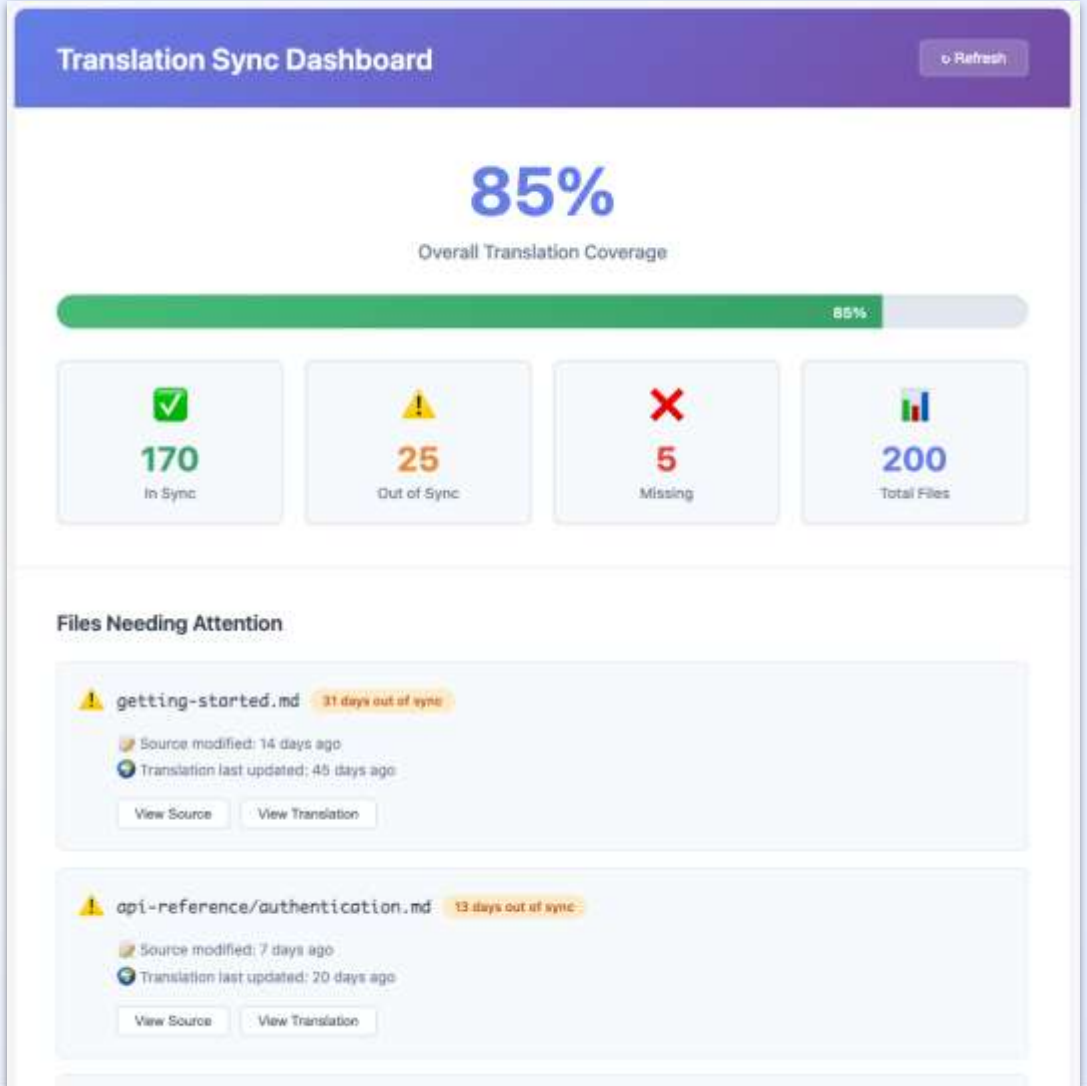
Check source mod date vs. translated version

Determine which files are in sync, out of sync, or missing

Create HTML status page with metrics

Notify team when translation work is needed

Translation sync



DOCUMENT360

Translation sync

Key lessons

- ✓ Dashboards provide instant visibility
- ✓ Separate concerns: status tracking ≠ translation
- ✓ Simple timestamp logic is reliable
- ✓ AI translation prompts need iteration and standards
- ✗ Vague translation prompts
- ✗ No terminology standards

Translation sync

Best practice -> ask AI to help write prompts

- AI knows what info it needs and how to structure it
- Helps clarify ideas
- Learn prompt engineering

Quality assessment

Quality assessment

Manual process

- Review documentation pages individually for quality
- Apply subjective judgment and inconsistent criteria
- Try to remember what you evaluated previously
- Repeat for hundreds of pages without systematic tracking
- Compile findings in spreadsheets

Quality assessment

Manual process

- Inconsistent scoring
- Time-consuming
- Doesn't scale to large doc sites
- No tracking
- Vague feedback

Need systematic, evidence-based quality assessment at scale

DOCUMENTATION QUALITY RUBRIC

🔧 API Reference Page

Purpose: Provides complete technical documentation for developers

Total Points Available **100**

🎓 Teaching Approach

This rubric is built like grading student papers: clear criteria for each content type, with specific point values and detailed descriptions. Each criterion defines exactly what "good" looks like.

CRITERION	DESCRIPTION	POINTS
Technical Completeness	Are all endpoints, methods, and options documented?	25
Code Examples	Are there working, copy-paste code samples?	20
Parameter Documentation	Are all parameters clearly described with types and requirements?	20
Audience Accessibility	Is the documentation understandable for both human developers and AI/machine parsing?	15
Error Handling	Are error codes and troubleshooting steps provided?	10
Response Examples	Are sample responses and data structures shown?	10

⚠️ Evidence-Based Scoring Required

AI must cite specific examples from the page to justify every score. No scores without evidence.

```
Example: "Code Examples: 10/20. Section 2 has authentication example, but Section 4 (Error Handling) lacks code samples showing how to catch and handle API errors."
```

Score Interpretation

90-100	80-89	70-79	Below 70
Excellent	Good	Acceptable	Needs Work

Quality assessment

Script crawls doc site and extracts content from pages

Match each page to rubric based on content type

AI scores (0-100) and must cite specific examples from the text

Provide specific improvement steps, not vague suggestions

Generate visual dashboard and Jira tickets with clear improvement steps

DOCUMENT360

Quality assessment

Key lessons

- ✓ Custom rubrics per content type
- ✓ Require cited evidence
- ✓ Actionable feedback
- ✓ Teaching approach
- ✓ Dashboard visibility

Quality assessment

Key lessons

- ❌ Generic one-size-fits-all rubric
- ❌ Scores without justification ("60/100" with no explanation)
- ❌ Vague feedback ("This could be better")
- ❌ No distinction between content types

Quality assessment

Best practice -> make AI show its work

- Verify accuracy & soundness of reasoning
- Catch hallucinations
- Actionable feedback

Key principles & getting started

Key principles & getting started

What works

1. Plan first
2. Ask AI to help you prompt
3. Clear context
4. Iterate in small steps
5. Use Git/GitHub
6. Ask "why?”
7. Treat as pair programming

Key principles & getting started

What doesn't work

1. ❌ Jump straight to code
2. ❌ Vague requests
3. ❌ Assume AI knows context
4. ❌ Blindly copy code
5. ❌ Everything at once
6. ❌ Skip testing
7. ❌ Direct to production

Choose your first project

Criteria

- High pain (time/frustration)
- Low risk (won't break production)
- Rule-based (clear patterns)

Good starter projects

- Changelog generation
- Broken link checking
- File organization
- Format conversion

Choose your first project

AI tool options

- Cursor + Claude (what I use)
- GitHub Copilot
- ChatGPT
- Choose what you have access to

Choose your first project

Architecture principles

- Single-purpose first - one script, one job
- File-based workflows - leverage Markdown & Git
- Strategic APIs - add integration when needed

Build team trust

- Document what's automated
- Keep humans in review loops
- Measure impact (efficiency, quality, team satisfaction)
- Be transparent about limitations

AI coding agents amplify your skills, they don't replace them. Your systematic thinking, clear communication, and attention to detail are what make automation valuable.

Your action plan this week

1. Identify one automation opportunity
2. Write down the manual process
3. Try describing it to an AI tool
4. Share what you learn

On the job market?
Ask a coding agent to help you build a documentation site with GitHub Pages as your portfolio

Learn more

Connect with me on LinkedIn
https://www.linkedin.com/in/rebeccaalvestechwriter/

Docs-automation-examples on GitHub

- Working release notes automation examples from this talk
- Check out the simple docs site, try the tutorials, contribute improvements
- Real code you can adapt to your workflow

DOCUMENT360

Questions ?

Thank You!