

Technical Documentation Writing Practices

- Quality documentation accelerates onboarding and reduces support load
- Covers audience analysis, IA, style, tooling, review, and maintenance
- Maps to pain points for writers, developers, and doc owners
- Outcomes: design, write, test, and maintain high-quality docs

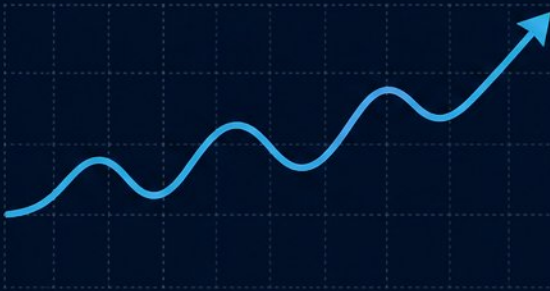


The Business Case for Documentation Quality

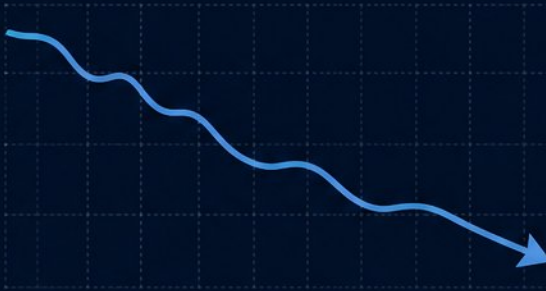
- Poor docs cost 15-25% of engineering capacity
- Developers spend 3-10 hours weekly searching for answers
- 60-70% abandon integrations failing within 30 minutes
- Quality docs drive 4-5x higher productivity metrics
- Use metrics: support load, onboarding time, search failures



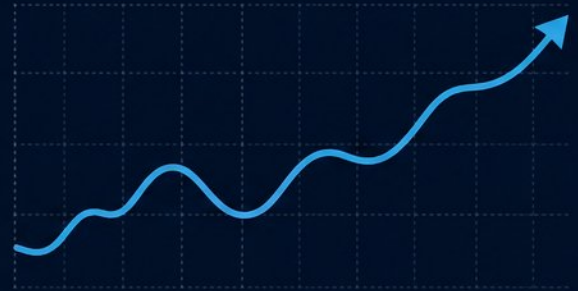
Support Load



Onboarding Time



Search Failures



Audience Analysis and Documentation Use Cases

- Identify readers: new users, developers, engineers, PMs, support
- Map content to intent via Diátaxis: tutorials, how-tos, reference, explanation
- Tailor vocabulary and depth to each audience's expertise
- Apply audience analysis before writing or revising documents



FIELD GUIDE

DOCUMENTATION AUDIENCES AND THEIR INTENT



NEW USERS

Learning the basics and getting started



TUTORIALS



DEVELOPERS

Building, integrating, and implementing solutions



HOW-TOS



ENGINEERS & PMs

Understanding APIs, options, and product capabilities



REFERENCE



SUPPORT

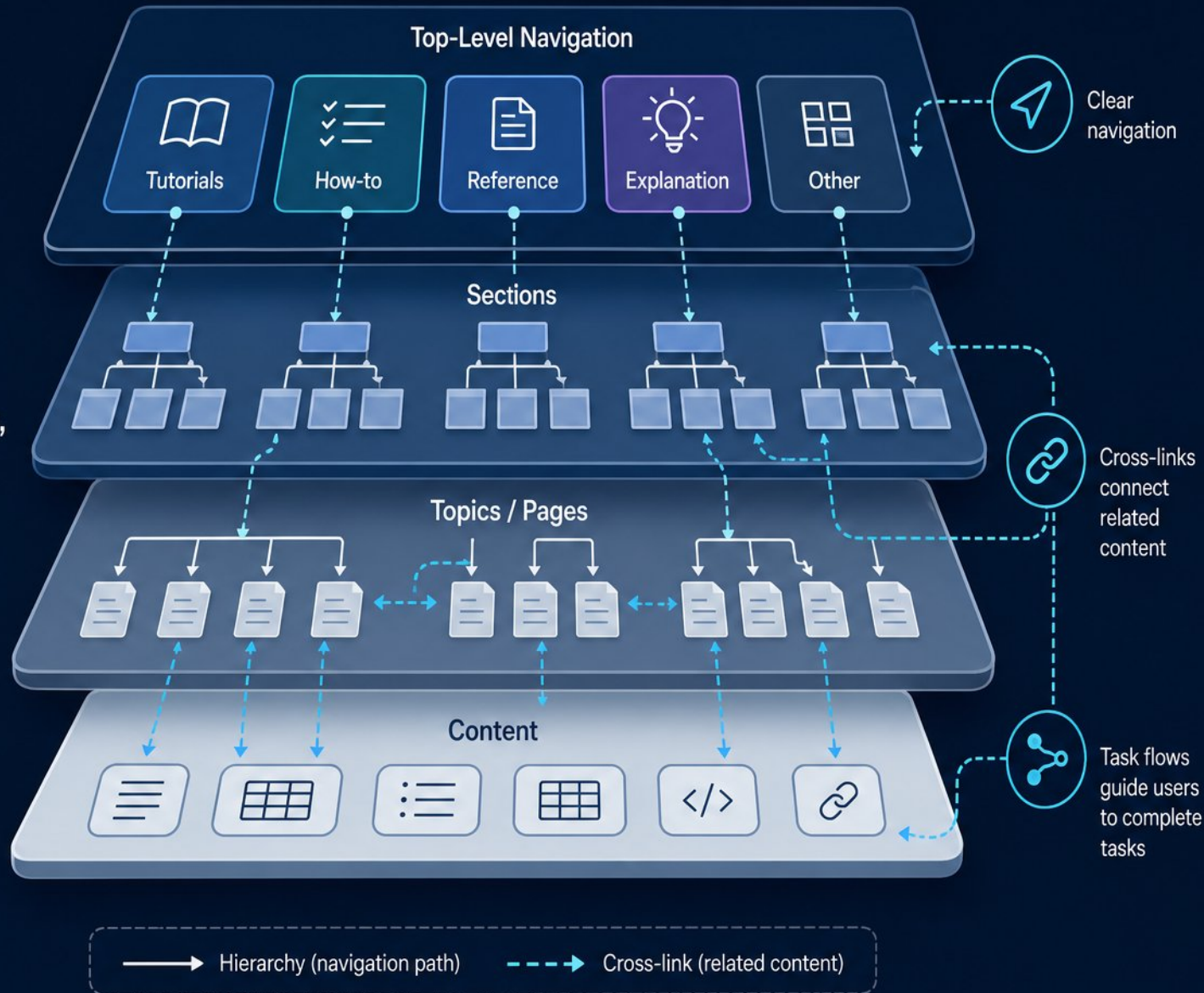
Troubleshooting, diagnosing, and solving issues



EXPLANATION

Information Architecture for Technical Content

- Organize by user tasks, not internal teams or features
- IA framework: Diátaxis — tutorials, how-to, reference, explanation
- Use clear navigation, headings, cross-links, and task flows
- Limit top-level categories to 5–9, keep to three clicks
- Keep pages self-contained with scannable headings and tables



Writing Clear, Accurate, and Maintainable Content



- Use plain language with precise terminology



- Write imperative, numbered steps with expected results



- Avoid vague pronouns, jargon, and inconsistent naming



- Create modular, reusable content to reduce maintenance



- Set context before action; state the goal first





Style Guides, Terminology Management, and Localization Readiness



- ▶ - Choose a style guide that fits your product and audience



- ▶ - Keep terminology and capitalization consistent across teams



- ▶ - Write short, clear sentences to ease translation



- ▶ - Align docs with code style, UI text, and product naming



RECOMMENDED



Terminology

Use the preferred term consistently.



Capitalization

Use sentence case for UI and headings.



Clarity for Translation

Write short, clear sentences.



Alignment

Match code style, UI text, and product naming.



AVOID



Terminology

Mix terms or invent new words.



Capitalization

Mix title Case, ALL CAPS, and sentence case.



Clarity for Translation

Use long, complex sentences.

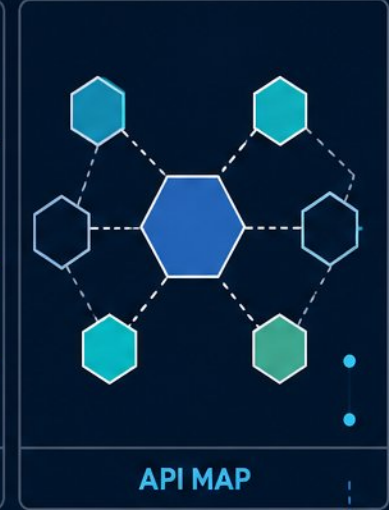
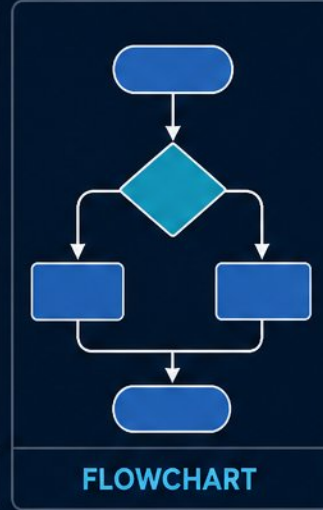
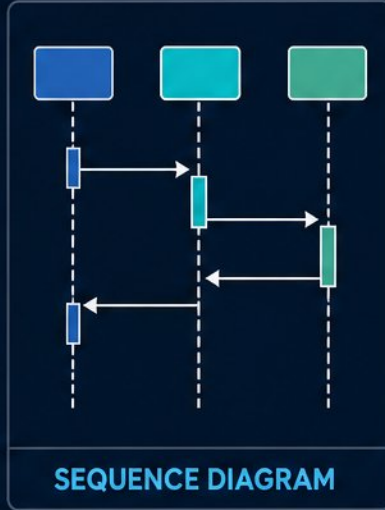
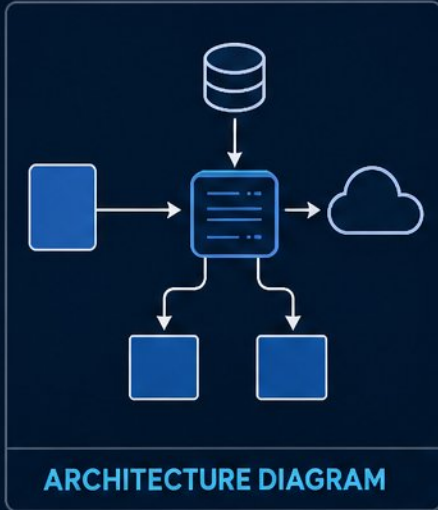


Alignment

Use different naming or ignore code style.



Effective Use of Diagrams, Screenshots, and Visual Documentation



- Use visuals only when clarity beats text; avoid extra maintenance costs



- Key types: architecture, sequence, flowchart, UI callouts, API maps



- Keep visuals reusable, accurate, and updateable as products change

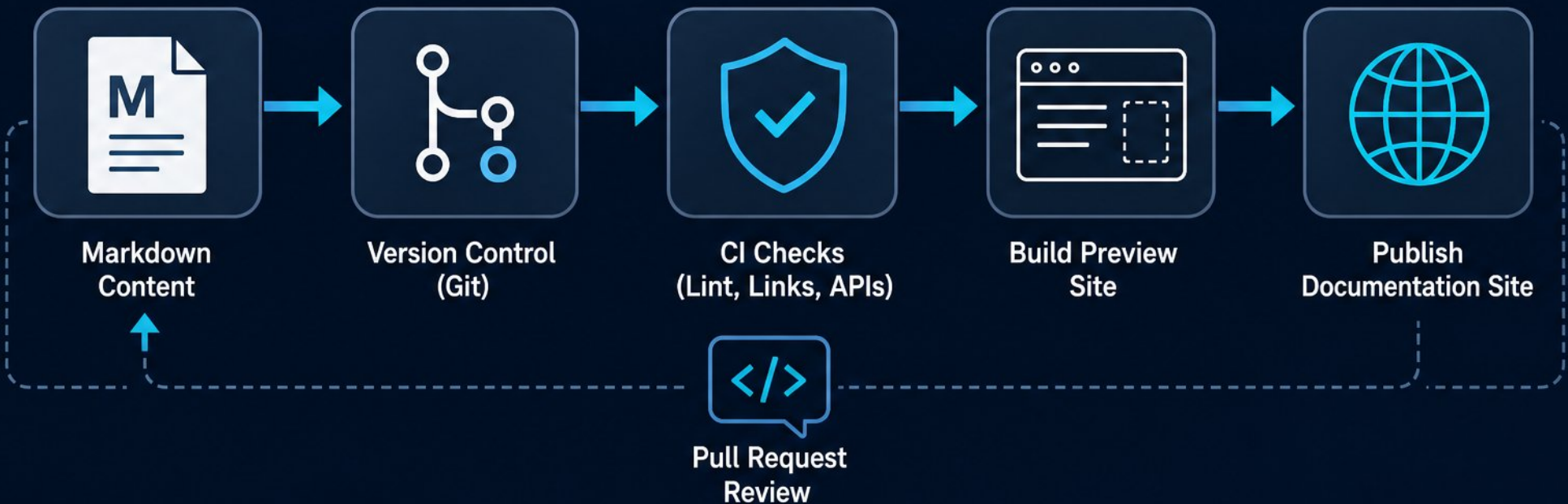


- Ensure accessibility with descriptive alt text and localization-friendly design



Documentation Tooling, Automation, and Docs-as-Code

- Docs live in Git; reviewed via pull requests
- CI/CD builds previews and publishes sites
- Automate linting, link checks, and API generation
- Choose tools fitting your engineering workflow
- AI agents draft; human review keeps it accurate



Review, Testing, and Quality Assurance for Documentation

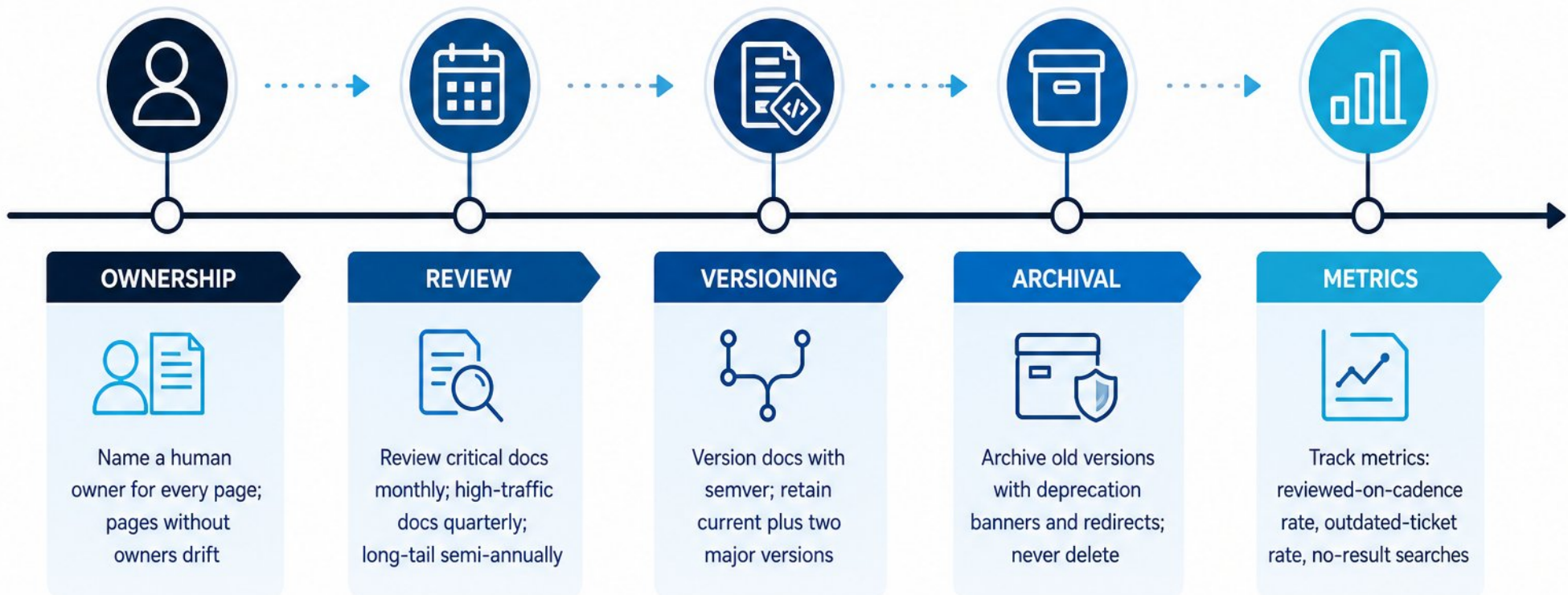
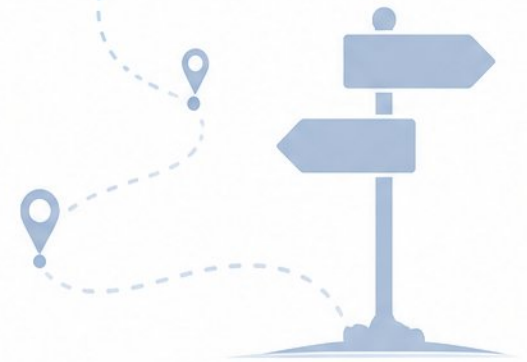
- Structured review stages: doc team, product team, support, legal, beta testers
- Technical accuracy review verifies code, commands, and API parameters
- Editorial review enforces style guide, voice, terminology, and consistency
- Testing docs alongside code catches drift and keeps content current
- Feedback loops and audits drive continuous documentation improvement





Governance, Maintenance, and Continuous Improvement

- - Name a human owner for every page; pages without owners drift
- - Review critical docs monthly; high-traffic docs quarterly; long-tail semi-annually
- - Version docs with semver; retain current plus two major versions
- - Archive old versions with deprecation banners and redirects; never delete
- - Track metrics: reviewed-on-cadence rate, outdated-ticket rate, no-result searches



Designing the Documentation Workflow

- ✓ - Plan documentation before writing: define scope, audience, success criteria
- ✓ - Templates and checklists enforce consistency across page types
- ✓ - Definition of Done: what must be complete before a page ships
- ✓ - Align deliverables with release cycles and engineering workflows



Writing Procedures and Task-Oriented Content



- Start each step with an imperative verb and active voice



- List prerequisites first, then numbered steps with expected results



- Add verification and recovery guidance for failure-prone steps



- Match step format to task complexity: single, multi, or multi-action



API Reference and Code Sample Best Practices



- Generate API reference from OpenAPI specs to eliminate drift



- Keep code samples runnable, realistic, and testable



- Document constraints and error responses with each endpoint



- Separate concepts, guides, and reference content by reading mode



GET /items

Retrieve a collection of items

GET



Parameters

query	status	Filter by status
query	limit	Maximum number of results
query	offset	Pagination offset



Description

Returns a list of items that match the optional query parameters.



Constraints

- limit must be a positive integer
- limit has a maximum value
- offset must be a non-negative integer



Error Responses

400	Bad Request	Invalid query parameter or value
401	Unauthorized	Missing or invalid authentication
404	Not Found	Requested resource was not found
500	Internal Server Error	Unexpected server error



Code Sample

cURL ▾

```
curl -X GET 'https://api.example.com/items' \
-H 'Accept: application/json' \
-H 'Authorization: Bearer YOUR_TOKEN' \
-G --data-urlencode 'status=active' \
--data-urlencode 'limit=50' \
--data-urlencode 'offset=0'
```

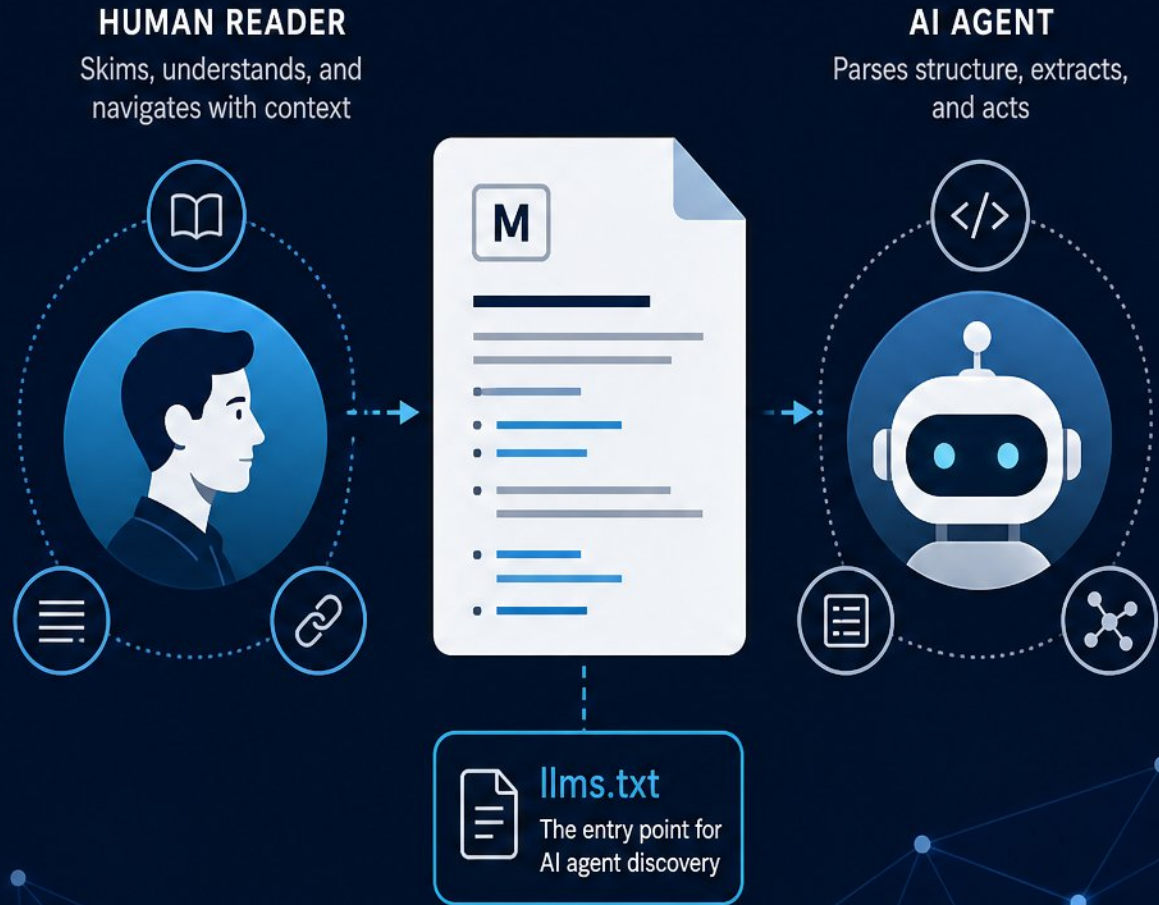
▶ Run Sample



Writing for Both Humans and AI Agents

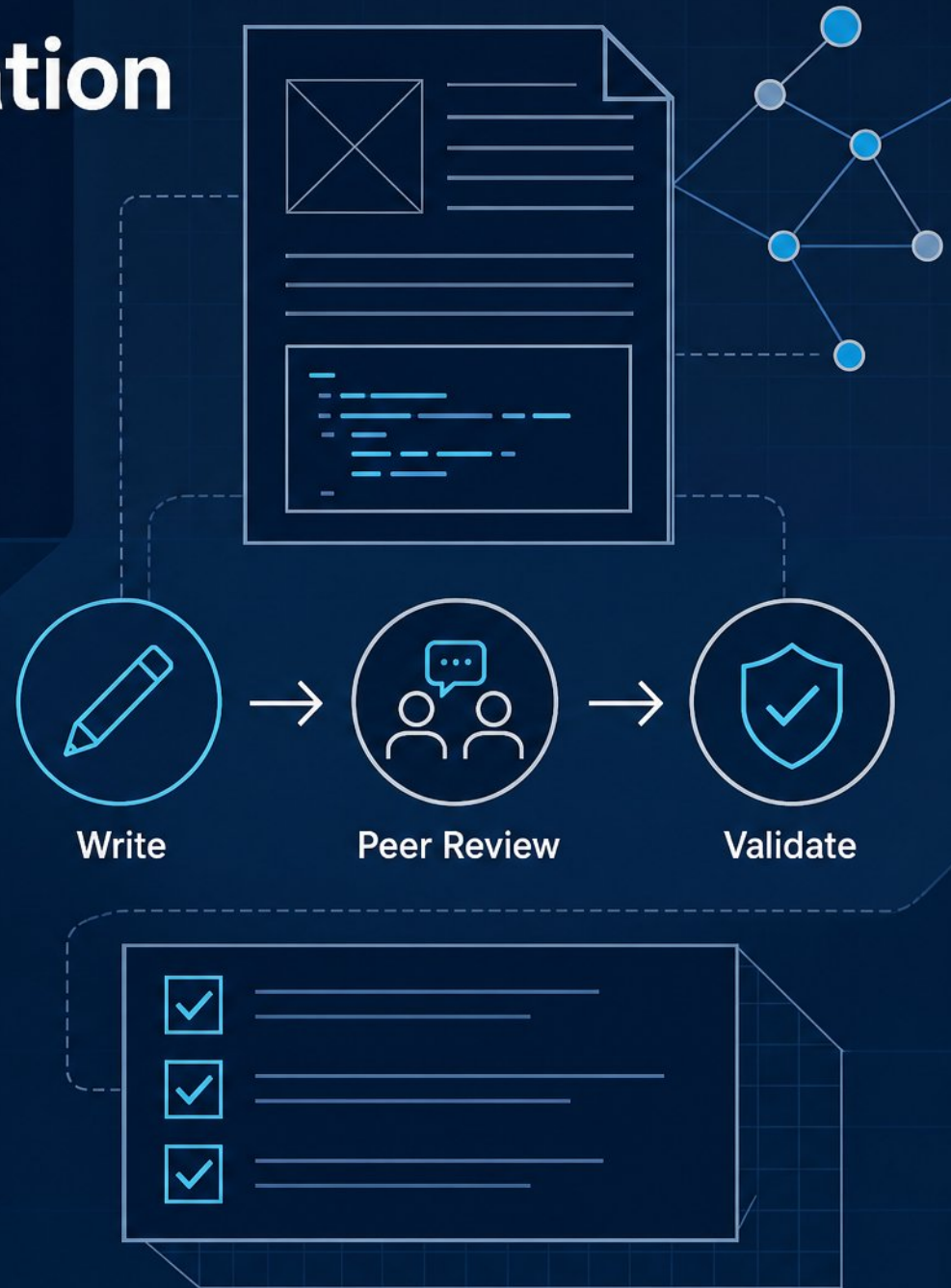
- AI agents consume docs differently: they parse structured formats like `llms.txt` and Markdown endpoints
- Use clear headings, descriptive link text, and LLM-friendly formats for machine readability
- Expose core docs via `llms.txt` to help agents find quickstarts, references, and SDK guides first
- Avoid patterns that confuse AI retrieval: hidden scripts, images-only instructions, and vague link text

SAME DOCUMENTATION, TWO READERS



Practical Documentation Project

- Apply audience analysis, information architecture, and writing principles
- Produce a tutorial, how-to guide, or reference page with a review checklist
- Simulate a docs-as-code workflow: write, peer review, and validate
- Collect feedback and define next steps for continuous improvement
- Share results and discuss takeaways with the group



PersonWise.

PersonWise • AI digital-human course platform



Scan the code or click anywhere to watch this course live, with voice Q&A

personwise.ai/t/d91bfed8/public