Knowledge Orchestration with Copilot

Keegan, Fiona

Exported on 2026-02-10 03:32:16

Table of Contents

1 Goal

Make docs generation & validation system actions by combining GitHub Copilot customizations (Instructions, Prompts, Agents, Skills, Collections) with golden-path docs structures and traditional CI gating.

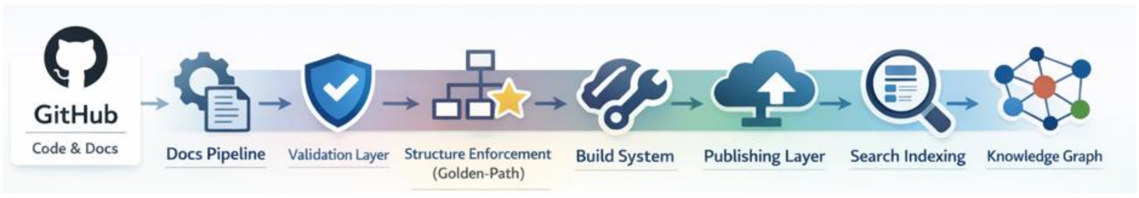
We aim to transform our docs processes from manual to an **end-2-end AI-augmented knowledge orchestration workflow** that's reliable.

To do this, we need to:

1. Standardize docs structure (golden path), schema-like `.md` layout (per patterns in GitHub > awesome-copilot).
2. Use semantic logic (prompts) in `.github/copilot-instructions.md` file in the repo.
3. Use Copilot Instructions to keep authoring behavior consistent.
4. Use Copilot Prompts/Agents to accelerate generation and remediation.
5. Use deterministic CI checks (linting, link checks, schema validation, policy) to gate what reaches customers.

1.1 E2E docs orchestration

GitHub (code + docs) > docs pipeline > validation layer > structure enforcement (golden path) > build system > publishing layer > search indexing > knowledge graph.



2 Overview

2.1 Why now

Today, docs accuracy depends heavily on manual review. We can reduce drift and increase consistency by making documentation a product of three things:

1. **Predictable structure** (golden-path file layout + templates) that humans and tools can reason about.
2. **Copilot customizations** to guide how people (and agent workflows) generate and update content (Instructions + Prompts + Agents).
3. **Deterministic validation** in CI to enforce correctness signals (structure, metadata, links, lint rules, policy) before publishing.

This turns docs quality into a **system property**. Humans optimize for clarity and UX, while automation enforces structure and gating. Copilot accelerates creation and remediation inside those guardrails.

2.2 Principles

Modular > pipeline-driven > automation-first > governance-aware > human-in-the-loop > **AI as executor, not owner.**

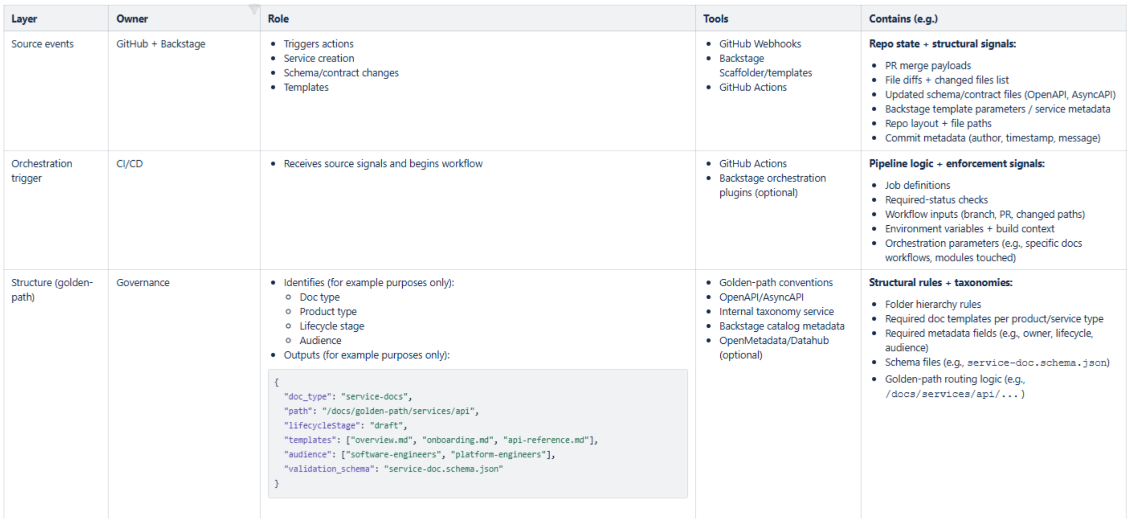
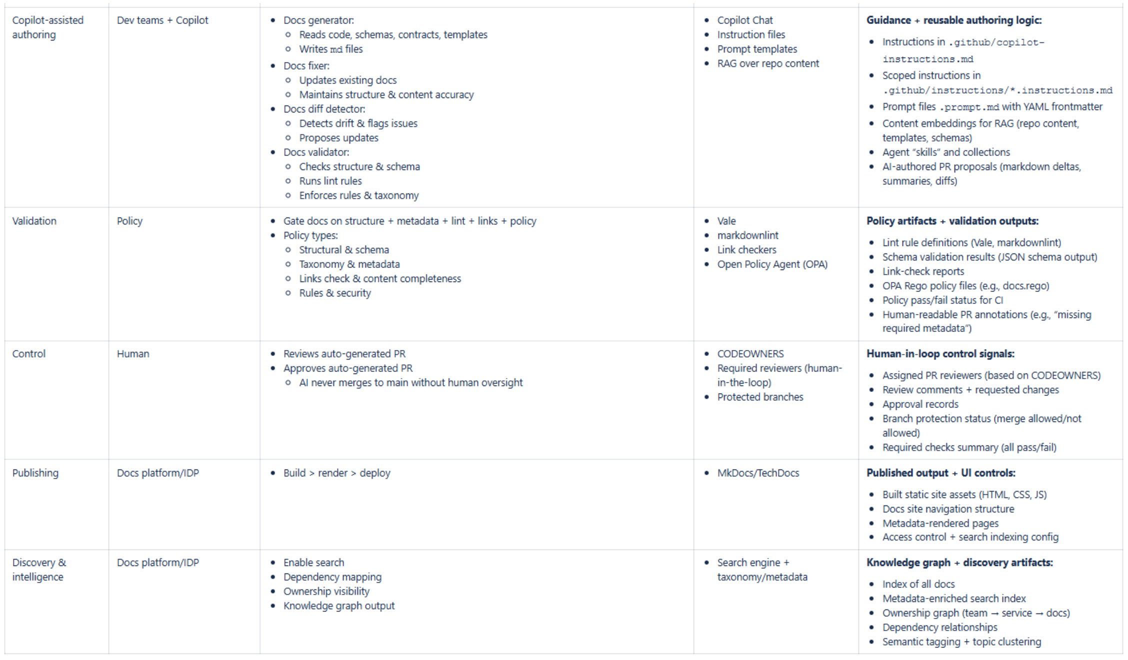
2.3 Governance

We'll implement governance using similar customization surfaces described in awesome-copilot:

1. **Repo-wide Instructions** in `.github/copilot-instructions.md` for global conventions and review guidance.
2. **Path-specific Instructions** in `.github/instructions/*.instructions.md` with YAML frontmatter like `applyTo` (and `description`) to scope guidance by file patterns (including docs paths).
3. **Reusable Prompts** in `.prompt.md` for repeatable doc tasks (generate overview, update API reference, create changelog entry, etc.), triggered via commands in Copilot Chat.
4. **Custom Agents** in `.agent.md` for specialized workflows (e.g., “Docs Maintainer”, “TechDocs Builder”), selected explicitly when using agent experiences.
5. **Skills + Collections** to bundle repeatable capabilities and curated sets of prompts/instructions/agents.

Copilot behavior is non-deterministic, so we'll use instructions to improve consistency, not to replace CI enforcement.

3 Mechanism

Layer	Owner	Role	Tools	Contains (e.g.)
Source events	GitHub + Backstage	• Triggers actions • Service creation • Schema/contract changes • Templates	• GitHub Webhooks • Backstage Scaffolder/templates • GitHub Actions	**Repo state + structural signals:** • PR merge payloads • File diffs + changed files list • Updated schema/contract files (OpenAPI, AsyncAPI) • Backstage template parameters / service metadata • Repo layout + file paths • Commit metadata (author, timestamp, message)
Orchestration trigger	CI/CD	• Receives source signals and begins workflow	• GitHub Actions • Backstage orchestration plugins (optional)	**Pipeline logic + enforcement signals:** • Job definitions • Required-status checks • Workflow inputs (branch, PR, changed paths) • Environment variables + build context • Orchestration parameters (e.g., specific docs workflows, modules touched)

Structure (golden-path)	Governance	• Identifies (for example purposes only):
		  ○ Doc type
		  ○ Product type
		  ○ Lifecycle stage
		  ○ Audience
		• Outputs (for example purposes only):
		`{`
		`  "doc_type": "service-docs",`
		`  "path": "/docs/golden-path/services/api",`
		`  "lifecycleStage": "draft",`
		`  "templates": ["overview.md", "onboarding.md", "api-reference.md"],`
		`  "audience": ["software-engineers", "platform-engineers"],`
		`  "validation_schema": "service-doc.schema.json"`
		`}`
			• Golden-path conventions
			• OpenAPI/AsyncAPI
			• Internal taxonomy service
			• Backstage catalog metadata
			• OpenMetadata/Datahub (optional)
				Structural rules + taxonomies:
				• Folder hierarchy rules
				• Required doc templates per product/service type
				• Required metadata fields (e.g., owner, lifecycle, audience)
				• Schema files (e.g., `service-doc.schema.json`)
				• Golden-path routing logic (e.g., `/docs/services/api/...`)

Copilot-assisted authoring	Dev teams + Copilot	• Docs generator: ○ Reads code, schemas, contracts, templates ○ Writes `md` files • Docs fixer: ○ Updates existing docs ○ Maintains structure & content accuracy • Docs diff detector: ○ Detects drift & flags issues ○ Proposes updates • Docs validator: ○ Checks structure & schema ○ Runs lint rules ○ Enforces rules & taxonomy	• Copilot Chat • Instruction files • Prompt templates • RAG over repo content	**Guidance + reusable authoring logic:** • Instructions in `.github/copilot-instructions.md` • Scoped instructions in `.github/instructions/*.instructions.md` • Prompt files `.prompt.md` with YAML frontmatter • Content embeddings for RAG (repo content, templates, schemas) • Agent "skills" and collections • AI-authored PR proposals (markdown deltas, summaries, diffs)
Validation	Policy	• Gate docs on structure + metadata + lint + links + policy • Policy types: ○ Structural & schema ○ Taxonomy & metadata ○ Links check & content completeness ○ Rules & security	• Vale • markdownlint • Link checkers • Open Policy Agent (OPA)	**Policy artifacts + validation outputs:** • Lint rule definitions (Vale, markdownlint) • Schema validation results (JSON schema output) • Link-check reports • OPA Rego policy files (e.g., docs.rego) • Policy pass/fail status for CI • Human-readable PR annotations (e.g., "missing required metadata")
Control	Human	• Reviews auto-generated PR • Approves auto-generated PR ○ AI never merges to main without human oversight	• CODEOWNERS • Required reviewers (human-in-the-loop) • Protected branches	**Human-in-loop control signals:** • Assigned PR reviewers (based on CODEOWNERS) • Review comments + requested changes • Approval records • Branch protection status (merge allowed/not allowed) • Required checks summary (all pass/fail)
Publishing	Docs platform/IDP	• Build > render > deploy	• MkDocs/TechDocs	**Published output + UI controls:** • Built static site assets (HTML, CSS, JS) • Docs site navigation structure • Metadata-rendered pages • Access control + search indexing config
Discovery & intelligence	Docs platform/IDP	• Enable search • Dependency mapping • Ownership visibility • Knowledge graph output	• Search engine + taxonomy/metadata	**Knowledge graph + discovery artifacts:** • Index of all docs • Metadata-enriched search index • Ownership graph (team → service → docs) • Dependency relationships • Semantic tagging + topic clustering

3.1 Why this works

AI-augmented authoring performs best when there is:

- **Clear structure** and headings to make instructions easier for Copilot to apply consistently.
- **Scoped instructions** (`applyTo`) reduce noise and improve relevance.
- **Reusable prompts** create repeatable workflows for common tasks.
- **Agents** are best used when you want a consistent "persona + workflow" for multi-step tasks.
- **Concise, directive instructions** tend to work better than long, sprawling guidance.

End-to-end, this makes docs quality enforceable.

Humans focus on clarity, UX, and strategy while Copilot accelerates creation and remediation, and deterministic checks ensure publishable integrity.

4 Outcome

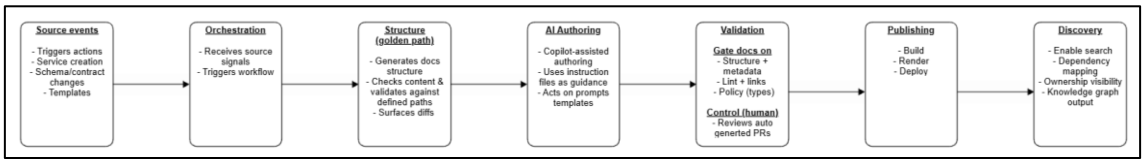
This creates an end-to-end pipeline where:

- **Correctness is enforced deterministically** for structure, metadata, lint, links, and policy gates.
- **Copilot handles acceleration** for docs generation, updates, and drift remediation (through Instructions + Prompts + optional Agents, always inside review and CI guardrails).
- **Humans focus on clarity, intent, UX, and strategy**, people remain accountable for what ships.
- **Backstage surfaces** validated, source-aligned docs, because publishing is downstream of the gates.

Docs shift from a manual editorial workflow to a self-maintaining, AI-assisted system.

5 Workflow

Here is the simplified workflow derived from the Mechanism table:



6 References

- GitHub > awesome-copilot
- GitHub > awesome-copilot > prompts/documentation-writer
- DocAider > Docs & DocAider > Updating docs