AI agents inside the engineering department

Who prepares, who decides, and what never merges, deploys or publishes on an agent's word

	IN BRIEF
01	Software engineering is where agents have gone furthest. The work is text, and the tools are scriptable.
02	Two things are on record. First, in the public cases where a coding agent removed live data, the agent held more access than its task needed. In one, a report says, it found a token "sitting in an unrelated file".
03	Second, what an agent reads can carry instructions. An issue, a pull request, a file or a package description can hold text written to steer it. Public vulnerability records describe this for several coding tools.
04	No vendor page we opened says an agent merges to the main branch or deploys with no person. Several products have settings that remove the person from each action. The public templates are looser. Of 14 we opened, none requires a named person or a branch rule before a change reaches the main branch or production.
05	Hosting is not the difference. More than one vendor offers coding agents on the customer's own infrastructure.
06	This paper is for the department that builds and maintains the company's own software. It is about the work around the code: issues, specifications, review, tests, documentation, dependencies, releases and incidents. Writing code in the editor is not the subject. It ends with seven questions for any vendor and a place to start.
07	The position throughout is simple, and it has five parts. Agents prepare and a named engineer decides. An agent is given read operations. Anything that writes is a separate tool that waits for a person. Every call is recorded. And review by a second person stays with a person. No agent merges to the main branch, approves a change, deploys, changes infrastructure or a database, opens a secrets store, publishes a package, or writes to a repository.

IN THIS PAPER

01	Where engineering departments are today	2
02	What the public templates do	3
03	Three doors, one gate	4
04	What an agent may do	5
05	What the rules already ask	7
06	A worked example: one payments company's engineering department	9
07	Seven questions before an agent touches a repository	10
08	Where to start	10

PART 01

Where engineering departments are today

6

published vulnerability records we opened for AI coding tools from four vendors, 2025 and 2026. Each describes code running or data leaving without the user's approval

The failures on record share one cause. We found two public cases in which a coding agent removed live data.

July 2025: a production database deleted during a freeze. Two press reports describe the same event. A founder testing Replit's agent wrote that it deleted his live database although he had told it to change nothing. Fortune reports that the system was in a designated "code and action freeze" [1] [2].

Fortune quotes the company's chief executive: "Replit agent in development deleted data from the production database. Unacceptable and should never be possible". The same report says the company then brought in "automatic separation between development and production databases" [1]. The agent built for development could reach production. That was the access it did not need.

April 2026: a database and its backups, in one call. We opened one press report, which rests on the founder's own account, so we name no company. A coding agent met a credential problem and deleted the storage volume that held the production data. The backups sat on the same volume. "It took 9 seconds," the founder wrote. The report says the agent found a token "sitting in an unrelated file". The token had been made for managing custom domains, but "its permissions were not limited to those specific actions" [3].

July 2025: a token with too wide a scope. This was not an agent's mistake, but the cause is the same. Amazon Web Services' own bulletin says an extension for its coding assistant "had an inappropriately scoped GitHub token" in its build configuration. With it, an outsider "was able to commit malicious code into the extension's open-source repository that was automatically included in a release". The bulletin says the code failed to run, and that the credentials were replaced [4].

What the three have in common. Something held a credential wider than its task, and nothing stood between that credential and a live system. An agent that holds only read operations cannot repeat the first two cases.

What an agent reads can carry instructions. GitHub's documentation for its own cloud agent says: "Users can include hidden messages in issues assigned to Copilot cloud agent or comments left for Copilot cloud agent as a form of prompt injection" [6].

The public vulnerability records show the same route. We opened six, for coding tools from four vendors. Each gives a fixed version or the vendor's advisory.

- One record for Cursor says an attacker could "chain a indirect prompt injection vulnerability to hijack the context to write to the settings file" and run code "without user approval".
- Two records for Claude Code describe code run, and data sent, from a project's own files before the user confirmed trust in the folder.
- One record for the Kiro IDE describes "maliciously crafted project directory files that bypass workspace trust protections".
- One record for GitHub Copilot and Visual Studio says a flaw "allows an unauthorized attacker to execute code locally" [5].

No vendor is alone in this. The lesson is about design. Treat everything an agent reads as material to report on, and give it no tool that could act on that text without a person.

What the vendors say. We opened no vendor page that says an agent merges to the main branch or deploys to production with no person.

- GitHub states the strongest review design we found. Its cloud agent "cannot approve or merge a pull request", and by default "workflows are not triggered until Copilot cloud agent's code is reviewed" [6].
- Cognition describes its agent's output as "draft PRs waiting for review" [10].
- Google's documentation for Jules says of the agent's plan: "You can review and approve it before any code changes are made" [11].
- Cline's documentation says: "Every action requires your explicit approval" [12].
- CodeRabbit says its coding agent delivers work "as a commit or a stacked pull request for you to approve" [13].
- Cursor says its cloud agents "produce merge-ready PRs with artifacts to demo their changes". The page we read describes no approval step [9].

Several products have settings that remove the person from each action.

- GitHub: "By default, Copilot's reviews do not count toward required approvals". When the setting is on, "Copilot can submit an approving review that satisfies your repository's required-approval rule" [7].
- Qodo: "Qodo approves the PRs that meet your conditions based on severity, blast radius and your rules" [14].
- GitLab lists a flow that can "push the requested changes, and resolve the thread". It also lists a control to "gate sensitive agent actions with human approval at execution time" [8].
- Anthropic documents a mode for Claude Code that "disables permission prompts and safety checks so tool calls execute immediately", and advises: "Only use this mode in isolated environments" [15].
- OpenAI documents a setting for Codex described as "No sandbox; no approvals" and marks it "not recommended" [16].

PART 01 · CONTINUED

These are settings, and each vendor explains them. A buyer should know which are on.

Hosting is not the difference. Qodo lists "single-tenant SaaS, on-prem, or air-gapped" deployment [14]. One open-source coding agent describes itself as "deployable in your infrastructure" [19]. Where an agent runs is a question to ask, and more than one vendor can answer it.

What the studies do not settle. One study generated 576,000 code samples with 16 models. Its authors report that the share of package names that do not exist was "at least 5.2% for commercial models and 21.7% for open-source models". It was last revised in March 2025, and models have changed since [17]. Hence one agent on our list: a check that each new package exists.

On speed, a trial published in July 2025 found that experienced open-source developers took 19% longer with AI tools. In February 2026 its authors wrote that developers are probably faster now, but that their new data is "only very weak evidence for the size of this increase" [18]. We print no productivity figure and promise none.

PART 02

What the public templates do

0 of 14

public engineering agent templates and example configurations we opened require a named person or a branch rule before a change reaches the main branch or production

We opened 14 public templates and example configurations for engineering work, and one examples page we could not judge. They come from one code-hosting platform's documentation and preview, three vendors' workflow actions, one open-source agent, two workflow platforms and one cookbook. We read each for one thing: must a named person, or a branch rule, approve before a change reaches the main branch or production?

WHAT THE TEMPLATE DOES	DOES A PERSON APPROVE FIRST?
Fixes style findings on a pull request and opens a new one with the fixes	**No step. Developers "review and merge these fixes with a single click"**
Reviews infrastructure pull requests and "auto-approves clean changes or routes flagged ones to Slack for human sign-off"	**Only for the flagged ones**
Approves every pull request opened by a dependency update bot, with one command	**No. The script approves**
Turns on merging for a patch update of one dependency	**Only if the repository's own rules ask for "tests and approvals"**
Fixes a failed build and opens a pull request	**The page says "Check to see if everything looks good and then merge it". Nothing enforces it**
Runs agents that read, and stages each write for a separate job	**In part. Reviewers on write jobs are an option the user sets**

PART 02 · CONTINUED

The count. None of the 14 requires it. 8 do so in part: on some paths, as an option, or through the repository's ordinary pull request. 6 have no step at all. Two of the 14 are examples for a dependency update bot, not an AI agent; they are the standard pattern for automated updates [19] [20].

Two findings matter more than the count. First, most of these agents stop at a branch or a pull request. The gate is then the repository's own branch rule, which the template does not set up. One vendor's notes say as much: "Branch protection on your default branch is still worth having" [19].

Second, the credentials are wide. One template asks for a token with the scopes "repo, workflow, admin:repo_hook". Another gives an agent tools to "Create, Edit, Delete, Get, and List files" [20]. That is the pattern behind the cases in Part 1.

One design stands out. A code-hosting platform's preview states: "Agents read by default. Every write is checked before it lands." The agent, it says, "does not hold the authority to change" the repository [19]. That is the separation this paper recommends.

PART 03

Three doors, one gate

An agent is only as useful as what it can reach. In an engineering department that means three kinds of thing.

Documents.

The coding standard, the review checklist, design documents, decision records, runbooks and past incident reviews. These are indexed once as knowledge sources, with each reader's permissions kept, so a colleague does not reach through an agent a document they could not open themselves.

Systems of record.

Source control, the issue tracker, the build system, the package registry. Most are reached through an interface the company already runs. The company gives the platform the description of that interface and picks which operations become tools.

Files and outside sources.

A diff, a test report, a build log or a coverage report arrives as a file. Repositories, issues, on-call alerts, metrics, logs and chat channels are reached through MCP servers deployed in the company's own cluster.

All three pass one gate: every tool call is checked against the company's policy, which can allow it, record it, hold it for a person's approval, or block it.

FOUR WAYS INTO AN ENGINEERING DEPARTMENT'S SYSTEMS

	ROUTE	WHAT IT REACHES	WHAT TO KNOW
01	**A file the user uploads**	A diff, a test report, a build log, a design document, an export from the issue tracker	Nothing is connected. The safest start
02	**A knowledge source**	The coding standard, the review checklist, design documents, runbooks, past incident reviews	Permissions are kept per reader
03	**An imported description of an interface**	Source control, the issue tracker, the build system, the package registry	The company picks the operations. Changes wait for approval
04	**An MCP server, deployed in the company's cluster**	Repositories, issues, alerts, metrics and logs, chat channels	It runs inside. Check what each tool does

PART 03 · CONTINUED

12,931

operations sent as changes, out of 24,103, in the public interface descriptions we imported (135 files read)

What is published. We ran Katonic's importer on 135 public interface descriptions from 64 publishers of systems an engineering department uses. 115 import as they are and 20 are over the 5 MB import limit. None was refused. The 115 give 24,103 operations. 11,172 read, and 12,931 are sent as changes. The method alone cannot tell which of those only read. The counts are per file, and some publishers split one system into several files or list several versions [21].

Who the publishers are. Source control, issue tracking, build and release, registries, feature flags, error tracking and metrics, on-call, cloud and container platforms, gateways, documentation and chat.

What that exercise taught us. More than half of the operations change something. That is the reason for the approval rule, and for giving an agent the read operations and nothing else.

The largest publishes only large files. The largest code-hosting platform in the set publishes its description only in files over the import limit. A description over the limit is imported in the part the company needs.

Several common systems publish nothing. One large code-hosting and pipeline suite, two widely used build servers, two artefact stores, a code quality server, a secrets and provisioning vendor and the test management tools publish no description a file address reaches. For those, the description comes from the company's own installation, or the agent reads a report or an uploaded file.

So pick the operations at import, read what each one does, and start with the ones that only read.

PART 04

What an agent may do

One rule runs through every agent in this paper: it prepares, and a named engineer decides.

01	**It reads** the issues, changes, logs, documents and records it has been given, through read tools only.
02	**It drafts** the summary, the checklist result, the test cases, the release notes, with the file and the line behind each statement. A draft is a file for the engineer. It is not a commit.
03	**It lists** what a report is missing, what a change touches, and what a release has no evidence for.
04	**It says when it found nothing.** It is built not to fill a gap from memory.
05	**It stops** before anything that changes a repository, a build, a database or a running system. A tool that changes something waits for approval unless the company has decided otherwise for that tool.
06	**It scores no engineer.**

PART 04 · CONTINUED

That gives 35 agents in nine areas. The full list is a separate document.

AREA	AGENTS	WHAT THEY PREPARE	WHO DECIDES
Backlog and issues	4	The kind of issue and what is missing; earlier issues that match; a backlog summary	An engineer and the product owner
Specifications and design	4	A draft specification; what a design leaves out; a draft decision record; requirements no test refers to	The author, the reviewers, the team lead
Code review	4	A summary of a change; each checklist point with its line; what else a change touches; a check on each new package	The reviewing engineer
Tests	4	Draft test cases as a file; failures grouped by likely cause; unstable tests; code no test exercises	The engineer on duty and the reviewer
Documentation	4	Answers with the passage cited; draft module documentation; what changed between two interface versions; a draft runbook	The module owner and the service owner
Dependencies	3	Newer versions and what changed; licences against the policy; services an advisory affects	An engineer, the engineering lead, legal
Build and release	4	Where a build failed; draft release notes; a draft change record; release checklist items with no evidence	The release manager and the change manager
Incidents and reliability	4	A draft incident summary; a draft review; errors grouped by cause; the service level report	The engineer on call and the service owner
Running the department	4	The technical debt list; an onboarding guide; the delivery report; change control evidence	The head of engineering and the control owner

Four lines are drawn tighter than the rest.

The review.

An agent may read a change against the team's own checklist and list each point for the reviewer, with the line. It approves nothing and merges nothing. An engineer reads the change and gives the review.

What it reads.

An issue, a pull request, a file or a package description is material to report on, never an order. The agents hold no tool that could act on such text without a person.

The release.

An agent may draft the change record and list what the change policy asks for that is missing. The change manager approves the change, and an engineer deploys it. No agent deploys, changes infrastructure or a database, opens a secrets store, or publishes a package.

The report.

The delivery report counts work by team and service, with the records behind each figure. It scores no engineer. After an incident, the review draft names no person as the cause.

PART 05

What the rules already ask

This part needs a read by the company's own lawyers and auditors before it is relied on. It reports what the published texts say; it is a summary, not legal advice. Nothing in it says that a platform makes a company compliant. Whether a rule applies depends on the company's sector, size and country, and on what it sells.

Review before release

The card payment standard. We could not open the official text of PCI DSS, so we quote nothing from it. A third-party copy of version 4.0.1 reads as follows. Custom software is reviewed before release to find coding flaws. Where the review is manual, it is done by someone other than the author of the code, and management approves it before release [22].

The supply chain framework. SLSA's source track says, at its highest level: "The SCS requires two trusted persons to review all changes to protected branches" [23].

The secure development framework. The United States standards institute's framework describes code review as one where "a person looks directly at the code to find issues". It leaves the choice between review and automated analysis "as defined by the organization" [24].

The open question. Each text speaks of persons. None that we read settles whether an agent can be the second reviewer. This paper does not claim it can. An agent prepares the review, and an engineer gives it.

Products with software, sold in the European Union

The Cyber Resilience Act. "This Regulation shall apply from 11 December 2027. However, Article 14 shall apply from 11 September 2026". Article 14 is the reporting duty: an early warning of an actively exploited vulnerability "within 24 hours of the manufacturer becoming aware of it".

Manufacturers must also "exercise due diligence when integrating components sourced from third parties", open-source components included, and draw up "a software bill of materials in a commonly used and machine-readable format" [25].

The Product Liability Directive. Its definition of a product "includes electricity, digital manufacturing files, raw materials and software". It applies "to products placed on the market or put into service after 9 December 2026". It "does not apply to free and open-source software that is developed or supplied outside the course of a commercial activity" [26].

Neither text, in the parts we read, mentions code written by AI. Who wrote a line does not change who answers for the product.

Secure development duties

The network security directive. NIS2 lists, among the measures an entity must take, "security in network and information systems acquisition, development and maintenance, including vulnerability handling and disclosure". Entities must weigh their suppliers' practices, "including their secure development procedures" [27]. The duty reaches a company through national law.

For financial firms, the Digital Operational Resilience Act asks for change management that ensures "that all changes to ICT systems are recorded, tested, assessed, approved, implemented and verified in a controlled manner" [28].

Too much agency

A widely used list of risks in AI applications names this one. "The root cause of Excessive Agency is typically one or more of: excessive functionality; excessive permissions; excessive autonomy." Its advice: "require a human to approve high-impact actions before they are taken" [29].

An open-source security foundation's guide says: "AI-generated code isn't a shortcut around engineering processes such as code reviews, testing, static analysis, documentation, and version control discipline" [30].

PART 05 · CONTINUED

Licences and generated code

The licence texts ask for notices. One common licence says: “You must cause any modified files to carry prominent notices stating that You changed the files”. Another: “The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software”. A third: “You must license the entire work, as a whole, under this License” [31]. None mentions AI. Whether a generated passage is a copy is for counsel.

The main court case is not finished. On 16 September 2026 a United States appeals court “affirmed the district court’s dismissal of part of an action” over a coding assistant. It did not decide whether the assistant’s output infringes: “We express no view on whether that similarity would allow plaintiffs to assert a claim for copyright infringement” [32]. We checked no later filing.

The AI Act

There is no entry called engineering. Two points still matter. A company that makes “a substantial modification to a high-risk AI system” can become its provider. And the Act’s list covers systems meant “to allocate tasks based on individual behaviour or personal traits or characteristics or to monitor and evaluate the performance and behaviour of persons” at work [33]. That is one reason no agent here scores an engineer. Whether an entry catches a particular use is for counsel.

The dates have moved. Regulation (EU) 2026/1744 says the high-risk rules apply from “2 December 2027 as regards AI systems classified as high-risk pursuant to Article 6(2) and Annex III” and from 2 August 2028 for those under Article 6(1) and Annex I [34].

PLACE	WHAT WE READ	WHAT IT ASKS
Card payments, worldwide	PCI DSS 4.0.1, a third-party copy	Review before release, by someone other than the author
Any software supply chain	SLSA 1.2, source track	“Two trusted persons” on protected branches [23]
European Union	Cyber Resilience Act	Reporting “within 24 hours”; a bill of materials [25]
European Union, financial firms	Digital Operational Resilience Act	Changes “recorded, tested, assessed, approved” [28]

We did not open the official PCI DSS text, the audit standards behind financial reporting controls, the text of ISO/IEC 27001, the SOC 2 criteria, the joint guidance of six national agencies on agentic AI, or any export control rule. A company should check each with its own advisers.

What the rules have in common

01	**A change is reviewed before release** by someone other than its author.
02	**The texts speak of persons.** None we read says an agent can be the reviewer.
03	**The company answers for what it ships,** whoever or whatever wrote it, and for the components inside it.
04	**Access is kept to what the task needs,** and a person approves the actions that matter.

PART 06

A worked example: one payments company's engineering department

Take a made-up payments company: 1,400 staff, an engineering department of 220, and about 600 changes a week. Nothing here is a customer's story, and every figure is illustrative; it is section 4 put to work.

01

At triage, 90 new issues arrive in a day. The agent proposes a label and a component for each, and lists what is missing from 31. An engineer confirms or corrects each one.

02

Before a review, an engineer opens a change of 640 lines. The agent's note lists 11 checklist points with the line of each, and two database migrations the description does not mention. The reviewer reads the change herself, asks for one fix, and approves.

03

On a new package, a change adds three dependencies. The agent reports that one name does not exist in the company's registry and that one licence is outside the policy. The engineer removes the first and asks the engineering lead about the second.

04

After a failed test run, 48 tests fail overnight. The agent groups them into four likely causes, with the log lines and the six changes since the last passing run. The engineer on duty reverts one change.

05

Before a release, the agent drafts the change record and lists two checklist items with no evidence on file. The change manager holds the release until both are filed. An engineer deploys.

Every one of these changes nothing by itself, writes to no repository, and leaves a record of who asked, which tool ran and what was decided.

PART 07

Seven questions before an agent touches a repository

01 **Where does it run?**
On our infrastructure, in our private cloud, or in the vendor's service? Where is the model, and does our source code leave? More than one vendor can run on your side. Ask the next six of each.

02 **What can it read?**
Does it keep each reader's permissions, or does it read every repository as one all-seeing account?

03 **What can it change?**
List every tool that pushes, merges, approves, deploys, deletes or publishes. Is each one separate from the read tools, and does each wait for a person?

04 **What credentials does it hold?**
Show us every token, its scope and where it is kept. Can the agent find a wider one in a file?

05 **What happens when it reads an instruction?**
In an issue, a pull request, a file or a package description. Which tool could act on that text with no person?

06 **What is recorded?**
Who asked, which tool ran, what was decided, for how long is that kept, and can an auditor read it?

07 **Can we stop it?**
One agent, one tool or one server, at once, by us. And which settings remove the person, and who may turn them on?

PART 08

Where to start

Pick work where reading is slow, the decision stays with an engineer, and nothing changes on the agent's word.

Questions and answers on the engineering documents
Engineers ask how a service works and get the answer with the passage cited. No repository is touched.

Test failure summary
The agent groups the failures of a test run by likely cause. The engineer on duty decides what to fix or revert.

Code review preparation
The agent lists the points of the team's own checklist for the reviewer. An engineer reviews and approves.

One team, one repository, invented data first. Then beside the manual process, measured against it.

About Katonic AI

Katonic AI builds the platform that organisations use to build, run and govern AI agents on their own infrastructure: in their own data centre, in a private or sovereign cloud, or with no connection to the internet at all. The platform is in production with governments, banks, telecom operators and utilities. Katonic is certified to ISO 27001.

KATONIC.AI/SOLUTIONS/ENGINEERING

SOURCES — Read on 23 November 2026. Figures are the publishers' own and were not audited by us.

01 Fortune, report of 23 July 2025 on a coding agent that deleted a live database during a code freeze, with the chief executive's statement. https://fortune.com/2025/07/23/ai-coding-tool-replit-wiped-database-called-it-a-catastrophic-failure/

02 The Register, report of 21 July 2025 on the same event. https://www.theregister.com/2025/07/21/replit_saastr_vibe_coding_incident/

03 TechRadar Pro, report of 2 May 2026 on a founder's account of a coding agent that deleted a production volume and its backups. https://www.techradar.com/pro/it-took-9-seconds-tech-founder-outlines-how-rogue-claude-powered-ai-tool-wiped-entire-company-database-and-backups-but-says-theres-no-such-thing-as-bad-publicity

04 Amazon Web Services, security bulletin AWS-2025-015, published 23 July 2025, updated 25 July 2025. https://aws.amazon.com/security/security-bulletins/AWS-2025-015/ ; and the CVE record CVE-2025-8217. https://cveawg.mitre.org/api/cve/CVE-2025-8217

05 CVE records, read from the CVE programme's own service: CVE-2025-54135 and CVE-2025-54136 (Cursor, published 5 and 1 August 2025); CVE-2025-59536 and CVE-2026-21852 (Claude Code, published 3 October 2025 and 21 January 2026); CVE-2026-4295 (Kiro IDE, published 17 March 2026), with the vendor's bulletin 2026-009-AWS; CVE-2025-53773 (GitHub Copilot and Visual Studio, published 12 August 2025). https://cveawg.mitre.org/api/cve/CVE-2025-54135 ; https://cveawg.mitre.org/api/cve/CVE-2025-54136 ; https://cveawg.mitre.org/api/cve/CVE-2025-59536 ; https://cveawg.mitre.org/api/cve/CVE-2026-21852 ; https://cveawg.mitre.org/api/cve/CVE-2026-4295 ; https://aws.amazon.com/security/security-bulletins/2026-009-AWS/ ; https://cveawg.mitre.org/api/cve/CVE-2025-53773

06 GitHub Docs, "Risks and mitigations for GitHub Copilot cloud agent", undated, read 23 November 2026. https://docs.github.com/en/copilot/concepts/agents/coding-agent/risks-and-mitigations

07 GitHub Docs, page on Copilot code review (the part headed "Copilot approvals"), undated. https://docs.github.com/en/copilot/concepts/agents/code-review

08 GitLab Docs, "GitLab Duo Agent Platform" (the feature table), undated. https://docs.gitlab.com/user/duo_agent_platform/

09 Cursor, documentation page on cloud agents, undated. https://cursor.com/docs/cloud-agent

10 Cognition, Devin documentation, introduction page, undated. https://docs.devin.ai/get-started/devin-intro

11 Google, Jules documentation, getting started page, undated. https://jules.google/docs

12 Cline, documentation home page, undated. https://docs.cline.bot/

13 CodeRabbit, documentation home page, undated. https://docs.coderabbit.ai/

14 Qodo, home page, undated, read 23 November 2026. https://www.qodo.ai/

15 Anthropic, Claude Code documentation, "Permission modes", undated. https://code.claude.com/docs/en/permission-modes

16 OpenAI, Codex documentation, "Agent approvals and security", undated. https://developers.openai.com/codex/agent-approvals-security

17 Spracklen and others, "We Have a Package for You! A Comprehensive Analysis of Package Hallucinations by Code Generating LLMs", arXiv 2406.10279, submitted 12 June 2024, last revised 2 March 2025. https://arxiv.org/abs/2406.10279

SOURCES continued

18 METR, “Measuring the Impact of Early-2025 AI on Experienced Open-Source Developer Productivity”, 10 July 2025. https://metr.org/blog/2025-07-10-early-2025-ai-experienced-os-dev-study/ ; and the authors’ update of 24 February 2026. https://metr.org/blog/2026-02-24-uplift-update/

19 Eight templates and example configurations on or for code-hosting platforms: Anthropic’s workflow action for its coding product, with its security notes. https://github.com/anthropics/claude-code-action ; OpenAI’s workflow action. https://github.com/openai/codex-action ; Google’s workflow action. https://github.com/google-github-actions/run-gemini-cli ; GitHub Agentic Workflows (technical preview). https://github.github.com/gh-aw/ ; OpenAI cookbook example that fixes a failed build (marked archived on the page). https://developers.openai.com/cookbook/examples/codex/autofix-github-actions ; GitHub Docs, two example workflows for a dependency update bot (approve, and enable merging). https://docs.github.com/en/code-security/dependabot/working-with-dependabot/automating-dependabot-with-github-actions ; LangChain, Open SWE. https://github.com/langchain-ai/open-swe

20 Six templates and one examples page: n8n workflow templates 4073 (style fixes on a pull request), 19232 (deployment manifest review), 19175 (infrastructure pull request review), 4013 (test drafts as a comment) and 4629 (repository tools for an agent). https://n8n.io/workflows/4073 ; https://n8n.io/workflows/19232 ; https://n8n.io/workflows/19175 ; https://n8n.io/workflows/4013 ; https://n8n.io/workflows/4629 ; Zapier, template that sorts new issues and creates tickets. https://zapier.com/templates/details/monitor-triage-github-issues-severity-route-to-jira ; CrewAI documentation, examples page (the one not judged). https://docs.crewai.com/en/examples/example

21 135 public interface descriptions from 64 publishers, run through the platform’s importer on 23 November 2026 (Katonic research); and the search for publishers with no downloadable description (Katonic research).

22 PCI DSS version 4.0.1, requirements 6.2.3 and 6.2.3.1, read from a third-party copy (WithPCI). https://withpci.pages.dev/requirements/6/6.2/6.2.3 ; https://withpci.pages.dev/requirements/6/6.2/6.2.3.1

23 SLSA, version 1.2, “Source: Requirements for producing source” (Level 4, two-party review), undated. https://slsa.dev/spec/v1.2/source-requirements

24 National Institute of Standards and Technology, Special Publication 800-218, “Secure Software Development Framework (SSDF) Version 1.1”, February 2022, task PW.7.1. https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-218.pdf

25 Regulation (EU) 2024/2847 (Cyber Resilience Act), Article 71(2), Article 14(2), Article 13(5) and Annex I Part II. http://publications.europa.eu/resource/celex/32024R2847

26 Directive (EU) 2024/2853 (Product Liability Directive), Article 4(1) and Article 2(1) and (2). http://publications.europa.eu/resource/celex/32024L2853

27 Directive (EU) 2022/2555 (NIS2), Article 21(2)(e) and 21(3). http://publications.europa.eu/resource/celex/32022L2555

28 Regulation (EU) 2022/2554 (Digital Operational Resilience Act), Article 9(4)(e). http://publications.europa.eu/resource/celex/32022R2554

29 OWASP, Top 10 for LLM Applications 2025, “LLM06:2025 Excessive Agency”. https://genai.owasp.org/llmrisk/llm062025-excessive-agency/

30 Open Source Security Foundation, “Security-Focused Guide for AI Code Assistant Instructions”, 1 August 2025. https://best.openssf.org/Security-Focused-Guide-for-AI-Code-Assistant-Instructions

31 Licence texts: Apache License 2.0, section 4. https://www.apache.org/licenses/LICENSE-2.0.txt ; MIT License. https://opensource.org/license/mit ; GNU General Public License version 3, section 5, from the SPDX copy. https://raw.githubusercontent.com/spdx/license-list-data/main/text/GPL-3.0-only.txt

32 United States Court of Appeals for the Ninth Circuit, Doe v. GitHub, Inc., No. 24-7700, opinion filed 16 September 2026. https://cdn.ca9.uscourts.gov/datastore/opinions/2026/09/16/24-7700.pdf

33 Regulation (EU) 2024/1689 (AI Act), Article 25(1)(b) and Annex III point 4(b). http://publications.europa.eu/resource/celex/32024R1689

34 Regulation (EU) 2026/1744 of 8 July 2026, the replacement of Article 113(c) of Regulation (EU) 2024/1689. http://publications.europa.eu/resource/celex/32026R1744