/thoughtworks

Design. Engineering. AI.

Technology Radar

An opinionated guide to
today's technology landscape

Volume 33
Nov. 2025

About the Radar

Thoughtworkers are passionate about technology. We build it, research it, test it, open source it, write about it and constantly aim to improve it — for everyone. Our mission is to champion software excellence and revolutionize IT. We create and share the Thoughtworks Technology Radar in support of that mission. The Thoughtworks Technology Advisory Board, a group of senior technology leaders at Thoughtworks, creates the Radar. They meet regularly to discuss the global technology strategy for Thoughtworks and the technology trends that significantly impact our industry.

The Radar captures the output of the Technology Advisory Board's discussions in a format that provides value to a wide range of stakeholders, from developers to CTOs. The content is intended as a concise summary.

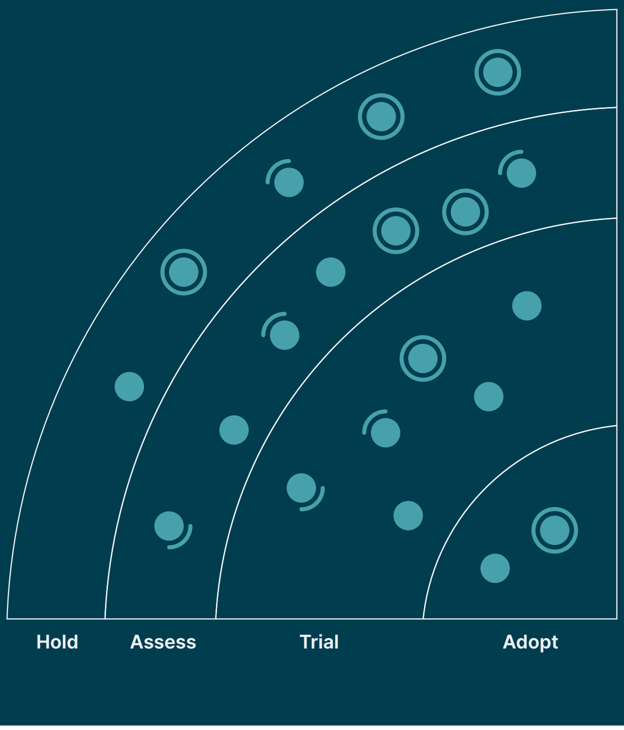
We encourage you to explore these technologies. The Radar is graphical in nature, grouping items into techniques, tools, platforms and languages and frameworks. When Radar items could appear in multiple quadrants, we chose the one that seemed most appropriate. We further group these items in four rings to reflect our current position on them.

For more background on the Radar, see thoughtworks.com/radar/faq.

Radar at a glance

The Radar is all about tracking interesting things, which we refer to as blips. We organize the blips in the Radar using two categorizing elements: quadrants and rings. The quadrants represent different kinds of blips. The rings indicate our recommendation for using that technology.

A blip is a technology or technique that plays a role in software development. Blips are “in motion” — their position in the Radar often changes — usually indicating our increasing confidence in recommending them as they move through the rings.



Adopt: We feel strongly that the industry should be adopting these items. We use them when appropriate in our projects.

Trial: Worth pursuing. It’s important to understand how to build up this capability. Enterprises can try this technology on a project that can handle the risk.

Assess: Worth exploring with the goal of understanding how it will affect your enterprise.

Hold: Proceed with caution.

◉ **New** ◔ **Moved in/out** ● **No change**

Our Radar is forward-looking. To make room for new items, we fade items that haven’t moved recently, which isn’t a reflection on their value but rather on our limited Radar real estate.

Contributors

The Technology Advisory Board (TAB) is a group of 22 senior technologists at Thoughtworks. The TAB meets twice a year face-to-face and biweekly virtually. Its primary role is to be an advisory group for Thoughtworks CTO Rachel Laycock.

The TAB acts as a broad body that can look at topics that affect technology and technologists at Thoughtworks. This edition of the Thoughtworks Technology Radar is based on a meeting of the TAB in Bucharest in September 2025.

Rachel Laycock (CTO)

Martin Fowler (Chief Scientist)

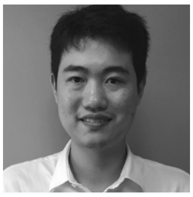
Alessio Ferri

Bharani Subramaniam

Birgitta Böckeler

Bryan Oliver

Camilla Falconi Crispim


Chris Chakrit Riddhagni

Effy Elden

James Lewis

Kief Morris

Ken Mugrage

Maya Ormaza

Nati Rivera

Neal Ford

Ni Wang

Nimisha Asthagiri

Pawan Shah

Selvakumar Natesan

Shangqi Liu

Vanya Seth

Will Amaral

Production Credits

Editorial Team

- **Willian Amaral**
  Product Owner
- **Nati Rivera**
  Product Owner
- **Preeti Mishra**
  Project and Campaign Manager
- **Richard Gall**
  Content Editor
- **Michael Koch**
  Copy Editor
- **Gareth Morgan**
  Head of Content and Thought Leadership

Design and Multimedia

- **Leticia Nunes**
  Lead Designer
- **Sruba Deb**
  Visual Designer
- **Ryan Cambage**
  Multimedia Specialist
- **Anish Thomas**
  Multimedia Designer

Digital and Web Experience

- **Rashmi Naganur**
  Business Analyst
- **Brigitte Britten-Kelly**
  Digital Content Strategist
- **Vandita Kamboj**
  UX Designer
- **Lohith Amruthappa**
  Analytics Specialist
- **Neeti Thakur**
  Marketing Automation Specialist

Communications

- **Shalini Jagadish**
  Internal Communications Specialist
- **Hiral Shah**
  Social Media Specialist
- **Abhishek Kasegaonkar**
  Social Media Specialist
- **Michelle Surendran**
  Public Relations Specialist
- **Anushree Tapuriah**
  Campaigns and Advertisement Specialist
- **Prakhar Nigam**
  Campaigns and Advertisement Specialist

Themes

Infrastructure orchestration arrives for AI

AI workloads are driving organizations to orchestrate large fleets of GPUs for both training and inference. Teams increasingly work with model sizes that exceed a single accelerator's capacity (even with 80 GB of HBM), pushing them toward distributed training and multi-GPU inference. As a result, platform teams are building complex, multi-stage pipelines and continuously tuning for throughput and latency. Discussions in this space included Nvidia DCGM Exporter for fleet telemetry and topology-aware scheduling to place jobs where interconnect bandwidth is highest.

Before this surge in GPU demand, Kubernetes had already solidified itself as the de facto container orchestrator — and it remains a strong substrate for managing AI workloads at scale, even as we also explored alternatives like micro and Uncloud. We're tracking emerging GPU-aware scheduling patterns — such as queueing and quota via Kueue, coupled with topology-aware placement and gang scheduling — to co-locate multi-GPU jobs on fast GPU-to-GPU links (e.g., NVLink/NVSwitch) and within contiguous data center "islands" (e.g., racks or pods with RDMA). Recent multi-GPU and NUMA-aware API improvements in Kubernetes further strengthen these capabilities, improving cross-device bandwidth, reducing tail latency and increasing effective utilization.

We expect rapid innovation in AI infrastructure as platform teams race to support the growing demand for AI coding workflows and the rise of agents elevated by MCP. In our view, GPU-aware orchestration is becoming table stakes — topology is now a first-class scheduling concern.

The rise of agents elevated by MCP

The dual rise of MCP and agents — and the expanding ecosystem of protocols and tools built around them — dominates this edition of the Radar. Virtually every major vendor is adding MCP awareness to their tools, which makes sense: In many ways, MCP has become the ultimate integration protocol for powering agents and enabling them to work efficiently and semi-autonomously. These capabilities are central to making agentic workflows productive.

We observed continued innovation in agentic workflows, where context engineering has proven critical to optimizing both behavior and resource consumption. New protocols such as A2A and AG-UI are reducing the boilerplate required to build and scale user-facing multi-agent applications. In the software development space, we compared different ways of supplying context to coding agents — from AGENTS.md files to patterns like anchoring coding agents to a reference application. As expected in the AI ecosystem, each Radar brings a new burst of innovation — last time it was RAG; this time, it's agentic workflows and the growing constellation of tools, techniques and platforms that support them, along with a few emerging AI antipatterns worth watching.

AI coding workflows

It's evident AI is transforming how we build and maintain software, and it continues to dominate our recent conversations. As AI becomes strategically embedded across the software value chain — from using AI to understand legacy codebases to GenAI for forward engineering — we're learning how to better supply knowledge to coding agents. Teams are experimenting with new practices such as defining custom instructions via AGENTS.md files and integrating with MCP servers like Context7 to fetch up-to-date dependency documentation.

There's also a growing realization that AI must amplify the entire team, not just individual contributors. Techniques like curated shared instructions and custom commands are emerging to ensure equitable knowledge diffusion. The tool landscape is vibrant: Designers explore UX Pilot and AI Design Reviewer, while developers prototype rapidly with v0 and Bolt for self-serve UI prototyping.

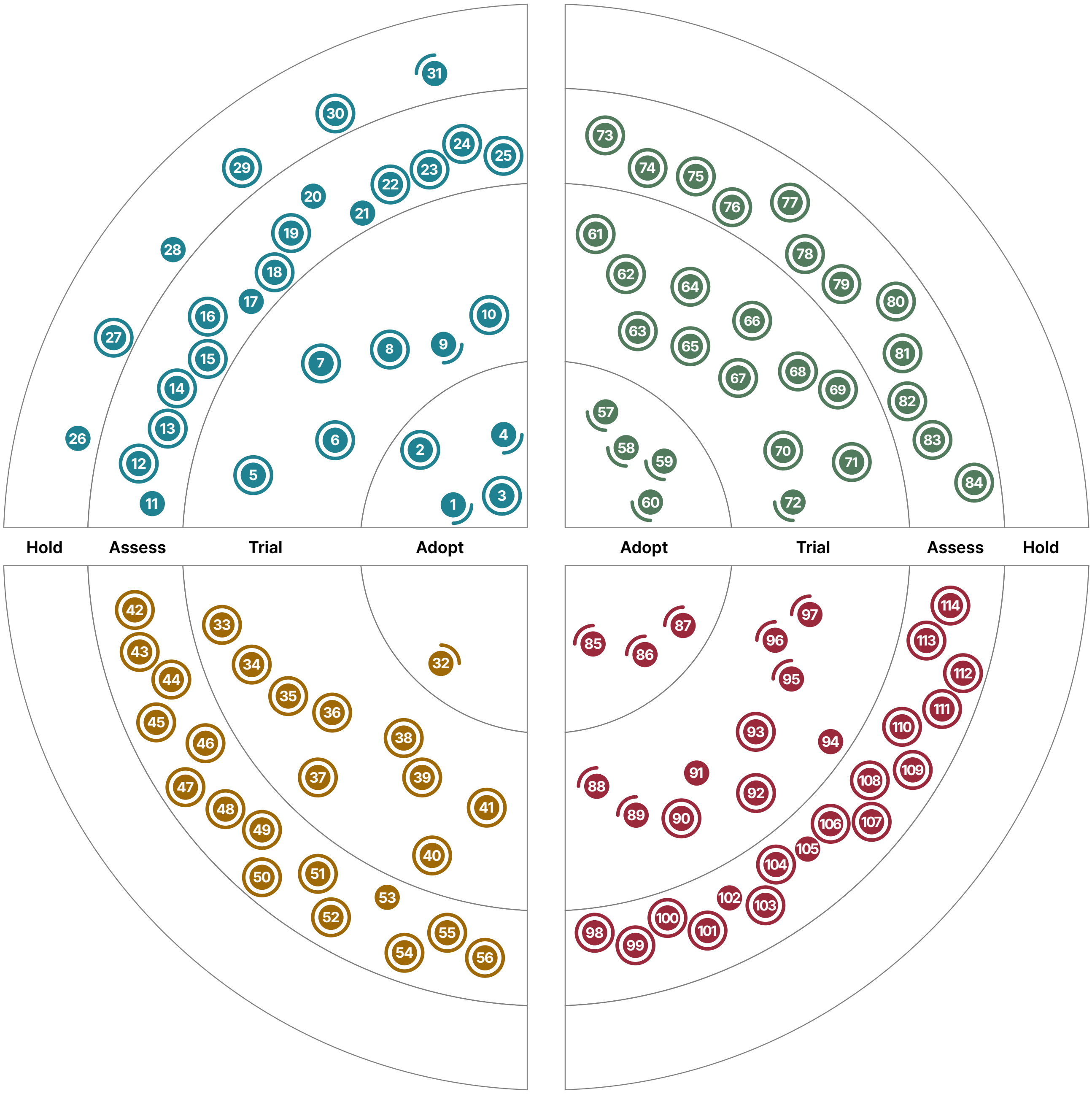
We also continue to debate spec-driven development — its scope, granularity and potential to serve as a single source of truth for incremental delivery. Yet amid the excitement, complacency with AI-generated code remains a shared concern, reminding us that while AI can accelerate engineering, human judgment is still indispensable.

Emerging AI antipatterns

The accelerating adoption of AI across industries has surfaced both effective practices and emergent antipatterns. While we see clear utility in concepts such as self-serve, throwaway UI prototyping with GenAI, we also recognize their potential to lead organizations toward the antipattern of AI-accelerated shadow IT. Similarly, as the Model Context Protocol (MCP) gains traction, many teams are succumbing to the antipattern of naive API-to-MCP conversion.

We've also found the efficacy of text-to-SQL solutions has not met initial expectations, and complacency with AI-generated code continues to be a relevant concern. Even within emerging practices such as spec-driven development, we've noted the risk of reverting to traditional software-engineering antipatterns — most notably, a bias toward heavy up-front specification and big-bang releases. Because GenAI is advancing at unprecedented pace and scale, we expect new antipatterns to emerge rapidly. Teams should stay vigilant for patterns that appear effective at first but degrade over time and slow feedback, undermine adaptability or obscure accountability.

The Radar



◎ New ◔ Moved in/out ● No change

The Radar

Techniques

Adopt

1. Continuous compliance
2. Curated shared instructions for software teams
3. Pre-commit hooks
4. Using GenAI to understand legacy codebases

Trial

5. AGENTS.md
6. AI for code migrations
7. Delta Lake liquid clustering
8. Self-serve UI prototyping with GenAI
9. Structured output from LLMs
10. TCR (Test && Commit || Revert)

Assess

11. AI-powered UI testing
12. Anchoring coding agents to a reference application
13. Context engineering
14. GenAI for forward engineering
15. GraphQL as data access pattern for LLMs
16. Knowledge flows over knowledge stocks
17. LLM as a judge
18. On-device information retrieval
19. SAIF
20. Service mesh without sidecar
21. Small language models
22. Spec-driven development
23. Team of coding agents
24. Topology-aware scheduling
25. Toxic flow analysis for AI

Hold

26. AI-accelerated shadow IT
27. Capacity-driven development
28. Complacency with AI-generated code
29. Naive API-to-MCP conversion
30. Standalone data engineering teams
31. Text to SQL

Platforms

Adopt

32. Arm in the cloud

Trial

33. Apache Paimon
34. DataDog LLM Observability
35. Delta Sharing
36. Dovetail
37. Langdock
38. LangSmith
39. Model Context Protocol (MCP)
40. n8n
41. OpenThread

Assess

42. AG-UI Protocol
43. Agent-to-Agent (A2A) Protocol
44. Amazon S3 Vectors
45. Ardoq
46. CloudNativePG
47. Coder
48. Graft
49. groundcover
50. Karmada
51. OpenFeature
52. Oxide
53. Restate
54. SkyPilot
55. StarRocks
56. Uncloud

Hold

—

The Radar

Tools

Adopt

57. ClickHouse
58. NeMo Guardrails
59. pnpm
60. Pydantic

Trial

61. AI Design reviewer
62. Barman
63. Claude Code
64. Cleanlab
65. Context7
66. Data Contract CLI
67. Databricks Assistant
68. Hoppscotch
69. NVIDIA DCGM Exporter
70. Relational AI
71. UX Pilot
72. v0

Assess

73. Augment Code
74. Azure AI Document Intelligence
75. Docling
76. E2B
77. Helix editor
78. Kueue
79. MCP-Scan
80. oRPC
81. Power user for dbt
82. Serena
83. SweetPad
84. Tape/Z Tools for Assembly Program Exploration for Z/OS

Hold

—

Languages and Frameworks

Adopt

85. Fastify
86. LangGraph
87. vLLM

Trial

88. Crossplane
89. DeepEval
90. FastMCP
91. LiteLLM
92. MLForecast
93. Nuxt
94. Phoenix
95. Presidio
96. Pydantic AI
97. Tauri

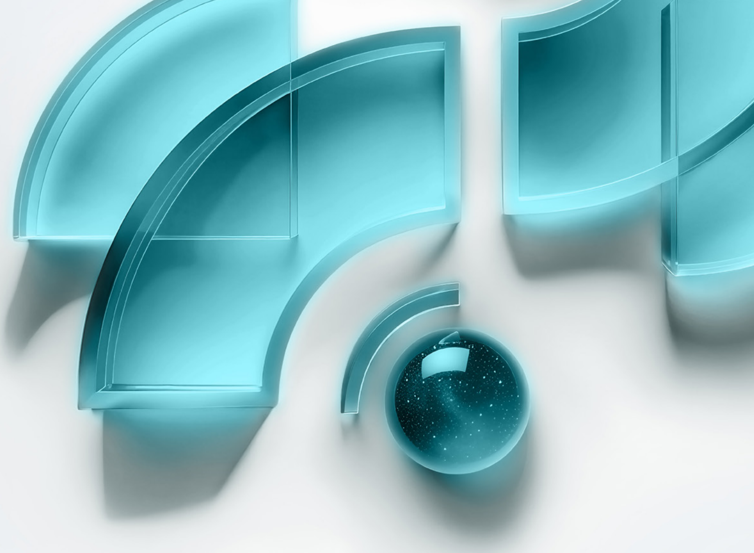
Assess

98. Agent Development Kit (ADK)
99. Agno
100. assistant-ui
101. AutoRound
102. Browser Use
103. DeepSpeed
104. Drizzle
105. Java post-quantum cryptography
106. kagent
107. LangExtract
108. Langflow
109. LMCache
110. Mem0
111. Open Security Control Assessment Language (OSCAL)
112. OpenInference
113. Valibot
114. Vercel AI SDK

Hold

—

Techniques



Adopt

1. Continuous compliance
2. Curated shared instructions for software teams
3. Pre-commit hooks
4. Using GenAI to understand legacy codebases

Trial

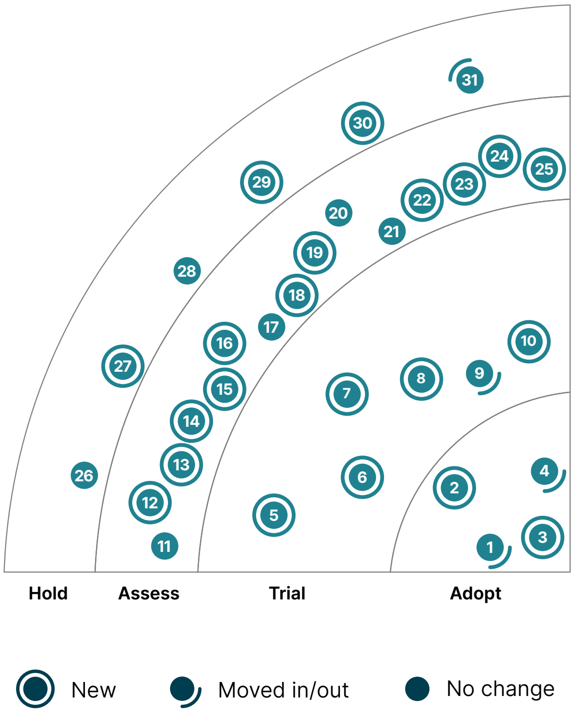
5. AGENTS.md
6. AI for code migrations
7. Delta Lake liquid clustering
8. Self-serve UI prototyping with GenAI
9. Structured output from LLMs
10. TCR (Test && Commit || Revert)

Assess

11. AI-powered UI testing
12. Anchoring coding agents to a reference application
13. Context engineering
14. GenAI for forward engineering
15. GraphQL as data access pattern for LLMs
16. Knowledge flows over knowledge stocks
17. LLM as a judge
18. On-device information retrieval
19. SAIF
20. Service mesh without sidecar
21. Small language models
22. Spec-driven development
23. Team of coding agents
24. Topology-aware scheduling
25. Toxic flow analysis for AI

Hold

26. AI-accelerated shadow IT
27. Capacity-driven development
28. Complacency with AI-generated code
29. Naive API-to-MCP conversion
30. Standalone data engineering teams
31. Text to SQL



1. Continuous compliance

Adopt

Continuous compliance is the practice of ensuring that software development processes and technologies meet regulatory and security standards on an ongoing basis through automation. Manual compliance checks can slow development and introduce human error, whereas automated checks and audits provide faster feedback, clearer evidence and simplified reporting.

By integrating policy-as-code tools such as Open Policy Agent and generating SBOMs within CD pipelines — aligned with SLSA guidance — teams can detect and address compliance issues early. Codifying rules and best practices enforces standards consistently across teams without creating bottlenecks. OSCAL also shows promise as a framework for automating compliance at scale.

Practices and tooling for continuous compliance are now mature enough that it should be treated as a sensible default, which is why we've moved our recommendation to Adopt. The increasing use of AI in coding — and the accompanying risk of complacency with AI-generated code — makes embedding compliance into the development process more critical than ever.

2. Curated shared instructions for software teams

Adopt

For teams actively using AI in software delivery, the next step is moving beyond individual prompting toward curated instructions for software teams. This practice helps you apply AI effectively across all delivery tasks — not just coding — by sharing proven, high-quality instructions. The most straightforward way to implement this is by committing instruction files, such as an AGENTS.md, directly to your project repository. Most AI-coding tools — including Cursor, Windsurf and Claude Code — support sharing instructions through custom slash commands or workflows. For noncoding tasks, you can set up organization-wide prompt libraries ready to use. This systematic approach allows for continuous improvement: As soon as a prompt is refined, the entire team benefits, ensuring consistent access to the best AI instructions.

3. Pre-commit hooks

Adopt

Git hooks have been around for a long time, but we feel they're still underused. The rise of AI-assisted and agentic coding has increased the risk of accidentally committing secrets or problematic code. While there are many mechanisms for code validation, such as continuous integration, pre-commit hooks are a simple and effective safeguard that more teams should adopt. However, overloading hooks with slow-running checks can discourage developers from using them, so it's best to keep them minimal and focused on risks that are most effectively caught at this stage of the workflow, such as secret scanning.

4. Using GenAI to understand legacy codebases

Adopt

In recent months, we've seen clear evidence that using GenAI to understand legacy codebases can significantly accelerate comprehension of large and complex systems. Tools such as Cursor, Claude Code, Copilot, Windsurf, Aider, Cody, Swimm, Unblocked and PocketFlow-Tutorial-Codebase-Knowledge help developers surface business rules, summarize logic and identify dependencies. Used alongside open frameworks and direct LLM prompting, they dramatically reduce the time needed to understand legacy codebases.

Our experience across multiple clients shows that GenAI-assisted understanding of legacy systems is now a practical default rather than an experiment. Setup effort varies, particularly for advanced approaches such as GraphRAG, and tends to scale with the size and complexity of the codebase being analyzed. Despite this, the impact on productivity is consistent and substantial. GenAI has become an essential part of how we explore and understand legacy systems.

5. AGENTS.md

Trial

AGENTS.md is a common format for providing instructions to AI coding agents working on a project. Essentially a README file for agents, it has no required fields or formatting other than Markdown, relying on the ability of LLM-based coding agents to interpret human-written, human-readable guidance. Typical uses include tips on using tools in the coding environment, testing instructions and preferred practices for managing commits. While AI tools support various methods for context engineering, the value of AGENTS.md lies in creating a simple convention for a file that acts as a starting point.

6. AI for code migrations

Trial

Code migrations take many forms — from language rewrites to dependency or framework upgrades — and are rarely trivial, often requiring months of manual effort. One of our teams, when upgrading their .NET framework version, experimented with using AI to shorten the process. In the past, we blipped OpenRewrite, a deterministic, rule-based refactoring tool. Using AI alone for such upgrades has often proven costly and prone to meandering conversations. Instead, the team combined traditional upgrade pipelines with agentic coding assistants to manage complex transitions. Rather than delegating a full upgrade, they broke the process into smaller, verifiable steps: analyzing compilation errors, generating migration diffs and validating tests iteratively. This hybrid approach positions AI coding agents as pragmatic collaborators in software maintenance. Industry examples, such as Google's large-scale int32-to-int64 migration, reflect a similar trend. While our results are mixed in measurable time savings, the potential to reduce developer toil is clear and worth continued exploration.

7. Delta Lake liquid clustering

Trial

Liquid clustering is a technique for Delta Lake tables that serves as an alternative to partitioning and Z-ordering. Historically, optimizing Delta tables for read performance required defining partition and Z-order keys at table creation based on anticipated query patterns. Modifying these keys later necessitates a full data rewrite. In contrast, liquid clustering employs a tree-based algorithm to cluster data based on designated keys, which can be incrementally changed without rewriting all data. This provides greater flexibility to support diverse query patterns, thereby reducing compute costs and enhancing read performance. Furthermore, the Databricks Runtime for Delta Lake supports automatic liquid clustering by analyzing historical query workloads, identifying optimal columns, and clustering data accordingly. Both standalone Delta Lake and Databricks Runtime users can leverage liquid clustering to optimize read performance.

8. Self-serve UI prototyping with GenAI

Trial

We use the phrase self-serve UI prototyping with GenAI to describe an emerging technique where tools such as Claude Code, Figma Make, Miro AI and v0 enable product managers to generate interactive, user-testable prototypes directly from text prompts. Instead of manually crafting wireframes, teams can generate functional HTML, CSS and JS artifacts in minutes — offering the speed of a sketch, but with real interactivity and higher fidelity. These “throwaway” prototypes trade polish for rapid learning, making them ideal for early validation during design sprints. However, higher fidelity can lead to misplaced focus on small details or unrealistic expectations about production effort — making clear framing and expectation management essential.

Used alongside user research, this technique accelerates discovery by turning abstract ideas into tangible experiences for users to react to. However, teams should take care not to let these tools replace the research process itself. Done well, self-serve prototyping shortens feedback loops, lowers barriers for non-designers and helps teams iterate quickly while maintaining a healthy balance between speed and quality.

9. Structured output from LLMs

Trial

Structured output from LLMs is the practice of constraining a large language model to produce responses in a predefined format — such as JSON or a specific programming class. This technique is essential for building reliable, production-grade applications, transforming the LLM’s typically unpredictable text into a machine-readable, deterministic data contract. Based on successful production use, we’re moving this technique from Assess to Trial.

Approaches range from simple prompt-based formatting and model-native structured outputs to more robust constrained decoding methods using tools like Outlines and Instructor, which apply finite-state machines to guarantee valid output. We’ve successfully used this technique to extract complex, unstructured data from diverse document types and convert it into structured JSON for downstream business logic.

10. TCR (Test && Commit || Revert)

Trial

Test && commit || revert (TCR) is a programming workflow derived from test-driven development that promotes very small, continuous steps through a simple rule: After each change, if the tests pass, the changes are committed; if they fail, the changes are reverted. Implementing TCR is straightforward: You only need to define a script that automates this cycle within your codebase. Originally introduced in a canonical article by Kent Beck, we’ve found that TCR reinforces positive coding practices such as YAGNI and KISS. It’s worth evaluating as we experiment with new workflows for building software using GenAI.

11. AI-powered UI testing

Assess

In the previous Radar, AI-powered UI testing primarily focused on exploratory testing, where we noted that the non-determinism of LLMs could introduce flakiness. With the rise of MCP, we’re now seeing major UI testing frameworks like Playwright and Selenium introduce their own MCP servers (playwright-mcp, mcp-selenium). These provide reliable browser automation through their native technologies, enabling coding assistants to generate reliable UI tests in Playwright or Selenium. While AI-powered UI testing remains a fast-evolving space — the latest Playwright release, for example, introduced Playwright Agents — we’re excited about those developments and look forward to seeing more practical guidance and field experience emerge.

12. Anchoring coding agents to a reference application

Assess

In the past we blipped the tailored service templates pattern, which helped organizations adopting microservices by providing sensible defaults to bootstrap new services and integrate them seamlessly with existing infrastructure. Over time, however, code drift between these templates and existing services tends to grow as new dependencies, frameworks and architectural patterns emerge. To maintain good practices and architectural consistency — especially in the age of coding agents — we've been experimenting with anchoring coding agents to a reference application. This pattern guides generative code agents by providing a live, compilable reference application instead of static prompt examples. A Model Context Protocol (MCP) server exposes both reference template code and commit diffs, enabling agents to detect drift and propose repairs. This approach transforms static templates into living, adaptable blueprints that AI can reference intelligently — maintaining consistency, reducing divergence and improving control over AI-driven scaffolding as systems evolve.

13. Context engineering

Assess

Context engineering is the systematic design and optimization of the information provided to a large language model during inference to reliably produce the desired output. It involves structuring, selecting and sequencing contextual elements — such as prompts, retrieved data, memory, instructions and environmental signals — so the model's internal layers operate in an optimal state. Unlike prompt engineering, which focuses only on the wording of prompts, context engineering considers the entire configuration of context: how relevant knowledge, instructions and prior context are organized and delivered to achieve the most effective results.

Today, engineers use a range of discrete techniques that can be grouped into three areas: *Context setup* covers curation tactics such as using minimal system prompts, canonical few-shot examples and token-efficient tools for decisive action. *Context management for long-horizon tasks* addresses finite context windows through context summarization, structured note-taking to persist external memories and sub-agent architectures to isolate and summarize complex sub-tasks. *Dynamic information retrieval* relies on just-in-time (JIT) context retrieval, where agents autonomously load external data only when immediately relevant, maximizing efficiency and precision.

14. GenAI for forward engineering

Assess

GenAI for forward engineering is an emerging technique for modernizing legacy systems through AI-generated descriptions of legacy codebases. It introduces an explicit step focused on what the legacy code does (its specification) while deliberately hiding how it's currently implemented. This relates to spec-driven development but is specifically applied to legacy modernization.

By generating and iterating on functional descriptions before rewriting the code, teams can use GenAI to surface hidden logic, dependencies and edge cases that might otherwise be overlooked. Emphasizing the problem space rather than the existing system also allows GenAI models to explore more creative and forward-looking solutions. The workflow follows a reverse-engineering → design/solutioning → forward-engineering loop, which enables both humans and AI agents to reason at a higher level before committing to an implementation.

At Thoughtworks, we're seeing multiple teams apply this approach successfully to accelerate the rewrite of legacy systems. The goal isn't to obscure implementation details entirely, but to introduce a temporary abstraction that helps teams and agents explore alternatives without being constrained by the current structure. This technique is showing promising results in producing cleaner, more maintainable and future-ready code while also reducing the time spent understanding existing implementations.

15. GraphQL as data access pattern for LLMs

Assess

GraphQL as a data access pattern for LLMs is an emerging approach for creating a uniform, model-friendly data access layer that enhances context engineering. It enables teams to expose structured, queryable data without granting models direct database access. Unlike REST APIs, which tend to over-fetch data or require new endpoints or filters for each use case, GraphQL lets the model retrieve only the data it needs — reducing noise, improving context relevance and cutting token usage.

A well-defined GraphQL schema also provides metadata that LLMs can use to reason about available entities and relationships, enabling dynamic, schema-aware querying for agentic use cases. This pattern offers a secure middle ground between REST and SQL, balancing governance controls with flexible access.

However, the approach relies on well-structured schemas and meaningful field names. Interpreting schema semantics and navigating complex structures remain challenging — and what is difficult for people to reason about is often equally difficult for LLMs. It's also important to be cognizant of the increased vectors for DoS attacks as well as typical GraphQL challenges, such as caching and versioning.

16. Knowledge flows over knowledge stocks

Assess

One question we get asked a lot is "how do we improve the way we share information amongst our teams?" Techniques for managing organizational knowledge continue to evolve, and one perspective we've found valuable borrows from systems thinking: the concepts of knowledge flows and knowledge stocks. Originally from economics, this framing encourages teams to view their organizational knowledge as a system — stocks representing accumulated knowledge and flows representing how knowledge moves and evolves through the organization. Increasing the flow of external knowledge into an organization tends to boost innovation. A time-tested way to improve flow is to establish communities of practice, which consistently show measurable benefits. Another is by deliberately seeking diverse, external sources of insight. As GenAI tools make existing knowledge stocks more accessible, it's worth remembering that nurturing fresh ideas and external perspectives is just as critical as adopting new technologies.

17. LLM as a judge

Assess

Using an LLM as a judge — to evaluate the output of another system, usually an LLM-based generator — has garnered attention for its potential to deliver scalable, automated evaluation in generative AI. However, we're moving this blip from Trial to Assess to reflect newly recognized complexities and risks.

While this technique offers speed and scale, it often fails as a reliable proxy for human judgment. Evaluations are prone to position bias, verbosity bias and low robustness. A more serious issue is scaling contamination: When LLM as a judge is used in training pipelines for reward modeling, it can introduce self-enhancement bias — where a model family favors its own outputs — and preference leakage, blurring the boundary between training and testing. These flaws have led to overfitted results that inflate performance metrics without real-world validity. There have been research studies that conduct more rigorous investigations into this pattern. To counter these flaws, we are exploring improved techniques, such as using LLMs as a jury (employing multiple models for consensus) or chain-of-thought reasoning during evaluation. While these methods aim to increase reliability, they also increase cost and complexity. We advise teams to treat this technique with caution — ensuring human verification, transparency and ethical oversight before incorporating LLM judges into critical workflows. The approach remains powerful but less mature than once believed.

18. On-device information retrieval

Assess

On-device information retrieval is a technique that enables search, context-awareness and retrieval-augmented generation (RAG) to run entirely on user devices — mobile, desktop or edge devices — prioritizing privacy and computational efficiency. It combines a lightweight local database with a model optimized for on-device inference. A promising implementation pairs sqlite-vec, a SQLite extension that supports vector search within the embedded database, with EmbeddingGemma, a 300 million parameter embedding model built on the Gemma 3 architecture. Optimized for efficiency and resource-constrained environments, this combination keeps data close to the edge, reducing dependence on cloud APIs and improving latency and privacy. We recommend teams assess this technique for local-first applications and other use cases where data sovereignty, low-latency and privacy are critical.

19. SAIF

Assess

SAIF (Secure AI Framework) is a framework developed by Google to provide a practical guide for managing AI security risks. It systematically addresses common threats such as data poisoning and prompt injection through a clear risk map, component analysis and practical mitigation strategies. We find its focus on the evolving risks of building agentic systems especially timely and valuable. SAIF offers a concise, actionable playbook that teams can use to strengthen security practices for LLM usage and AI-driven applications.

20. Service mesh without sidecar

Assess

As the cost and operational complexity of sidecar-based service meshes persists, we're excited to see another option for service mesh without sidecar emerge: Istio ambient mode. Ambient mode introduces a layered architecture that separates concerns between two key components: the per-node L4 proxy (ztunnel) and the per-namespace L7 proxy (Waypoint proxy). ztunnel ensures that L3 and L4 traffic is transported efficiently and securely. It powers the ambient data plane by fetching certificates for all node identities and handles traffic redirection to and from ambient-enabled workloads. The Waypoints proxy, an optional ambient mode component, enables richer Istio features such as traffic management, security and observability. We've had good experience with ambient mode in small-scale clusters and look forward to gaining more large-scale insights and best practices as adoption grows.

21. Small language models

Assess

We've observed steady progress in the development of small language models (SLMs) across several volumes of the Technology Radar. With growing interest in building agentic solutions, we're seeing increasing evidence that SLMs can power agentic AI efficiently. Most current agentic workflows are focused on narrow, repetitive tasks that don't require advanced reasoning, making them a good match for SLMs. Continued advancements in SLMs such as Phi-3, SmolLM2 and DeepSeek suggest that SLMs offer sufficient capability for these tasks — with the added benefits of lower cost, reduced latency and lower resource consumption compared to LLMs. It's worth considering SLMs as the default choice for agentic workflows, reserving larger, more resource-intensive LLMs only when necessary.

22. Spec-driven development

Assess

Spec-driven development is an emerging approach to AI-assisted coding workflows. While the term's definition is still evolving, it generally refers to workflows that begin with a structured functional specification, then proceed through multiple steps to break it down into smaller pieces, solutions and tasks. The specification can take many forms: a single document, a set of documents or structured artifacts that capture different functional aspects.

We've seen many developers adopt this style (and have one of our own that we're sharing internally at Thoughtworks). Three tools in particular have recently explored distinct interpretations of spec-driven development. Amazon's Kiro guides users through three workflow stages — requirements, design and tasks creation. GitHub's spec-kit follows a similar three-step process but adds richer orchestration, configurable prompts and a "constitution" defining immutable principles that must always be followed. Tessl Framework (still in private beta as of September 2025) takes a more radical approach in which the specification itself becomes the maintained artifact, rather than the code.

We find this space fascinating, though the workflows remain elaborate and opinionated. These tools behave very differently depending on task size and type; some generate lengthy spec files that are hard to review, and when they produce PRDs or user stories, it's sometimes unclear who their intended user is. We may be relearning a bitter lesson — that handcrafting detailed rules for AI ultimately doesn't scale.

23. Team of coding agents

Assess

Team of coding agents refers to a technique in which a developer orchestrates multiple AI coding agents, each with a distinct role — for example, architect, back-end specialist, tester — to collaborate on a development task. This practice is supported by tools such as Claude Code, Roo Code and Kilo Code which enable subagents and multiple operating modes. Building on the proven principle that assigning LLMs specific roles and personas improves output quality, the idea is to achieve better results by coordinating multiple role-specific agents rather than relying on a single general-purpose one. The optimal level of agent is still being explored, but it can extend beyond simple one-to-one mappings with traditional team roles. This approach marks a shift toward orchestrated, multi-step AI-assisted development pipelines.

24. Topology-aware scheduling

Assess

GPUs and LPUs are no longer standalone devices but tightly coupled networks of accelerators whose performance depends on placement and topology. In rack-scale systems like NVIDIA's NVL72, 72 GPUs share over 13 TB of VRAM and act as a single accelerator — until workloads cross-switch islands, turning collective operations into bottlenecks. Similarly, Groq's compile-time, software-scheduled architecture assumes deterministic data movement; random scheduling breaks those assumptions and predictability. Even within the same data center, GPU performance can vary significantly, creating demand for topology-aware scheduling that accounts for both hardware layout and variability when placing jobs.

Naive schedulers that ignore NVLink, PCIe or NIC topology often scatter multi-GPU workloads arbitrarily, resulting in degraded step time and efficiency. Training workloads, which are synchronous and bandwidth-bound, favor contiguous NVLink islands with uniform, high-bandwidth paths for all-reduce and pipeline stages. These jobs should co-schedule based on fabric bandwidth, avoid cross-switch hops and treat link, switch and node boundaries as failure domains. Inference workloads, by contrast, are latency and SLO-bound and typically balance replication for high availability across domains with sharding to keep mixture of experts (MoE) and KV-cache locality on the shortest paths. Optimizing placement for prefill versus decode phases, micro-batching and tenant isolation further improves efficiency. We believe topology-aware scheduling will become essential as accelerator performance grows increasingly dependent on network and data center topology. Our teams are already assessing Kueue and related projects to improve placement precision, boost performance and ensure reliable scaling for our clients.

25. Toxic Flow analysis for AI

Assess

The now-familiar joke that the S in MCP stands for "security" hides a very real problem. When agents communicate with one another — through tool invocation or API calls — they can quickly encounter what's become known as the lethal trifecta: access to private data, exposure to untrusted content and the ability to communicate externally. Agents with all three are highly vulnerable. Because LLMs tend to follow instructions in their input, content that includes a directive to exfiltrate data to an untrusted source can easily lead to data leaks. One emerging technique to mitigate this risk is toxic flow analysis, which examines the flow graph of an agentic system to identify potentially unsafe data paths for further investigation. While still in its early stages, toxic flow analysis represents one of several promising approaches to detecting the new attack vectors that agentic systems and MCP servers are increasingly exposed to.

26. AI-accelerated shadow IT

Hold

AI is lowering the barriers for noncoders to build and integrate software themselves, instead of waiting for the IT department to get around to their requirements. While we're excited about the potential this unlocks, we're also wary of the first signs of AI-accelerated shadow IT. No-code workflow automation platforms now support AI API integration (e.g., OpenAI or Anthropic), making it tempting to use AI as duct tape — stitching together integrations that previously weren't possible, such as turning chat messages in one system into ERP API calls via AI. At the same time, AI coding assistants are becoming more agentic, enabling noncoders with basic training to build internal utility applications.

This has all the hallmarks of the next evolution of the spreadsheets that still power critical processes in some enterprises — but with a much bigger footprint. Left unchecked, this new shadow IT could lead to a proliferation of ungoverned, potentially insecure applications, scattering data across more and more systems. Organizations should be aware of these risks and carefully weigh the trade-offs between rapid problem-solving and long-term stability.

27. Capacity-driven development

Hold

Key to the success of modern software development practices is maintaining a focus on the flow of work. Stream-aligned teams concentrate on a single, valuable flow — such as a user journey or product — which allows them to deliver end-to-end value efficiently. However, we're seeing a concerning trend toward capacity-driven development, where teams aligned in this way take on features from other products or streams when they have spare capacity. While this may seem efficient in the short term, it's a local optimization best suited to handling sudden spikes in demand. When normalized it increases cognitive load and technical debt, and in the worst case can lead to congestion collapse as the cost of context switching across products compounds. Teams with spare capacity are better served by focusing on improving system health. To manage capacity effectively, use WIP limits to control work across adjacent streams, consider cross-training to balance periods of high demand and apply dynamic reteaming techniques when needed.

28. Complacency with AI-generated code

Hold

As AI coding assistants and agents gain traction, so does the body of data and research highlighting concerns about complacency with AI-generated code. While there's ample evidence these tools can accelerate development — especially for prototyping and greenfield projects — studies show that code quality can decline over time.

GitClear's 2024 research found that duplicate code and code churn have risen more than expected, while refactoring activity in commit histories has dropped. Reflecting a similar trend, Microsoft research on knowledge workers shows that AI-driven confidence often comes at the expense of critical thinking — a pattern we've observed as complacency sets in with prolonged use of coding assistants.

The rise of coding agents further amplifies these risks, since AI now generates larger change sets that are harder to review. As with any system, speeding up one part of the workflow increases pressure on the others. Our teams are finding that using AI effectively in production requires renewed focus on code quality. We recommend reinforcing established practices such as TDD and static analysis, and embedding them directly into coding workflows, for example through curated shared instructions for software teams.

29. Naive API-to-MCP conversion

Hold

Organizations are eager to let AI agents interact with existing systems, often by attempting a seamless, direct conversion of internal APIs to the Model Context Protocol (MCP). A growing number of tools, such as MCP link and FastAPI-MCP, aim to support this conversion.

We advise against this naive API-to-MCP conversion. APIs are typically designed for human developers and often consist of granular, atomic actions that, when chained together by an AI, can lead to excessive token usage, context pollution, and poor agent performance. In addition, these APIs — especially internal ones — frequently expose sensitive data or allow destructive operations. For human developers, such risks are mitigated through architecture patterns and code reviews, but when APIs are naively exposed to agents via MCP, there's no reliable, deterministic way to prevent an autonomous AI agent from misusing such endpoints. We recommend architecting a dedicated, secure MCP server specifically tailored for agentic workflows, built on top of your existing APIs.

30. Standalone data engineering teams

Hold

Organizing separate data engineering teams to develop and own data pipelines and products — separate from the stream-aligned business domains they serve — is an antipattern that leads to inefficiencies and weak business outcomes. This structure repeats past mistakes of isolating DevOps, testing or deployment functions, creating knowledge silos, bottlenecks and wasted effort. Without close collaboration, data engineers often lack the business and domain context needed to design meaningful data products, limiting both adoption and value. In contrast, data platform teams should focus on maintaining the shared infrastructure, while cross-functional business teams build and own their data products, following data mesh principles. We put this practice in Hold to strongly discourage siloed organizational patterns — especially as the need for domain-rich, AI-ready data continues to grow.

31. Text to SQL

Hold

Text to SQL uses LLMs to translate natural language into executable SQL, but its reliability often falls short of expectations. We've moved this blip to Hold to discourage its use in unsupervised workflows — for example, dynamically converting user-generated queries where the output is hidden or automated. In these cases, LLMs frequently hallucinate due to limited schema or domain understanding, risking incorrect data retrieval or unintended data modification. The nondeterministic nature of LLM outputs also makes debugging and auditing errors challenging.

We advise treating Text to SQL with caution, human review is required for all generated queries. For agentic business intelligence, avoid direct database access and instead use a governed data abstraction semantic layer — such as Cube or dbt's semantic layer — or a semantically rich access layer like GraphQL or MCP.

Platforms

Adopt

32. Arm in the cloud

Trial

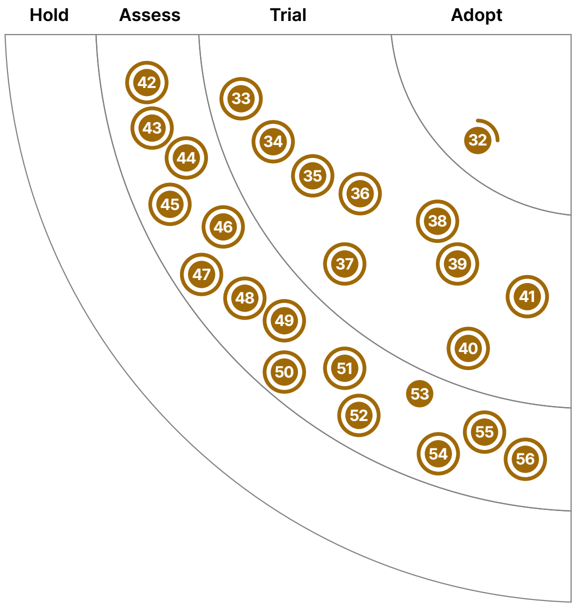
33. Apache Paimon
34. DataDog LLM Observability
35. Delta Sharing
36. Dovetail
37. Langdock
38. LangSmith
39. Model Context Protocol (MCP)
40. n8n
41. OpenThread

Assess

42. AG-UI Protocol
43. Agent-to-Agent (A2A) Protocol
44. Amazon S3 Vectors
45. Ardoq
46. CloudNativePG
47. Coder
48. Graft
49. groundcover
50. Karmada
51. OpenFeature
52. Oxide
53. Restate
54. SkyPilot
55. StarRocks
56. Uncloud

Hold

—



◉ New ◔ Moved in/out ● No change

32. Arm in the cloud

Adopt

Arm compute instances in the cloud have become increasingly popular in recent years for their cost and energy efficiency compared to traditional x86-based instances. Major cloud providers — including AWS, Azure and GCP — now offer robust Arm options. These instances are especially attractive for large-scale or cost-sensitive workloads. Many of our teams have successfully migrated workloads such as microservices, open-source databases and even high-performance computing to Arm with minimal code changes and only minor build-script adjustments. New cloud-based applications and systems increasingly default to Arm in the cloud. Based on our experience, we recommend Arm compute instances for most workloads unless specific architecture dependencies exist. Modern tooling, such as multi-arch Docker images, further simplifies building and deploying across both Arm and x86 environments.

33. Apache Paimon

Trial

Apache Paimon is an open-source data lake format designed to enable lakehouse architecture. It integrates seamlessly with processing engines like Flink and Spark, supporting both streaming and batch operations. A key advantage of Paimon's architecture lies in its fusion of a standard data lake format with an LSM (log-structured merge tree) structure. This combination addresses the traditional challenges of high-performance updates and low-latency reads in data lakes. Paimon supports primary key tables for high-throughput, real-time updates and includes a customizable merge engine for deduplication, partial updates and aggregations. This design enables efficient streaming data ingestion and management of mutable state directly within the lake. Paimon also provides mature data lake capabilities such as scalable metadata, ACID transactions, time travel, schema evolution and optimized data layouts through compression and Z-ordering. We recommend evaluating Paimon for projects that need a unified storage layer capable of efficiently handling large-scale append-only data and complex, real-time streaming updates.

34. DataDog LLM Observability

Trial

Datadog LLM Observability provides end-to-end tracing, monitoring and diagnostics for large language models and agentic application workflows. It maps each prompt, tool call and intermediate step into spans and traces; tracks latency, token usage, errors and quality metrics; and integrates with Datadog's broader APM and observability suite.

Organizations already using Datadog — and familiar with its cost structure — may find the LLM observability feature a straightforward way to gain visibility into AI workloads, assuming those workloads can be instrumented. However, configuring and using LLM instrumentation requires care and a solid understanding of both the workloads and their implementation. We recommend data engineers and operations staff collaborate closely when deploying it. See also our advice on avoiding standalone data engineering teams.

35. Delta Sharing

Trial

Delta Sharing is an open standard and protocol for secure, cross-platform data sharing, developed by Databricks and the Linux Foundation. It's cloud-agnostic, enabling organizations to share live data across cloud providers and on-prem locations without copying or replicating the data — preserving data freshness and eliminating duplication costs. We've seen an e-commerce company successfully use Delta Sharing to replace a fragmented partner data-sharing system with a centralized, real-time and secure platform, significantly improving collaboration. The protocol uses a simple REST API to issue short-lived pre-signed URLs, allowing recipients to retrieve large datasets using tools such as pandas, Spark or Power BI. It supports sharing data tables, views, AI models and notebooks. While it provides strong centralized governance and auditing, users should remain mindful of cloud egress costs, which can become a significant operational risk if unmanaged.

36. Dovetail

Trial

Dovetail platform addresses the persistent challenge of managing fragmented qualitative research data. It provides a centralized repository for user interviews, transcripts and insights, turning raw data into a structured, analyzable asset. We've found it invaluable in product discovery workflows, particularly for creating an evidence trail that links customer quotes and synthesized themes directly to product hypotheses and estimated ROI. In doing so, Dovetail strengthens the role of qualitative data in product decision-making.

37. Langdock

Trial

Langdock is a platform for organizations to develop and run generative AI agents and workflows for internal operations. It provides a unified environment with internal chat assistants, an API layer for connecting to multiple LLMs and tools for building agentic workflows that integrate with systems such as Slack, Confluence and Google Drive. The platform emphasizes data sovereignty, offering on-premise and EU-hosted options with enterprise compliance standards.

Organizations deploying Langdock should still pay close attention to data governance, and use techniques such as toxic flow analysis to avoid the lethal trifecta. Adopters should also consider the platform's maturity, evaluate the specific integrations they require and plan for any custom development that may be necessary.

38. LangSmith

Trial

LangSmith is a hosted platform from the LangChain team that provides observability, tracing and evaluation for LLM applications. It captures detailed traces of chains, tools and prompts, enabling teams to debug and measure model behavior, track performance regressions and manage evaluation data sets. LangSmith is a proprietary SaaS platform with limited support for non-LangChain workflows, making it most appealing to teams already invested in that ecosystem. Its integrated support for prompt evaluation and experimentation is notably more polished than open-source alternatives like Langfuse.

39. Model Context Protocol (MCP)

Trial

The Model Context Protocol (MCP) is an open standard that defines how LLM applications and agents integrate with external data sources and tools, significantly improving the quality of AI-generated outputs. MCP focuses on context and tool access, distinguishing it from the Agent2Agent (A2A) protocol, which governs inter-agent communication. It specifies servers (for data and tools such as databases, wikis and services) and clients (agents, applications and coding assistants). Since our last blip, MCP adoption has surged, with major companies such as JetBrains (IntelliJ) and Apple joining the ecosystem, alongside emerging frameworks like FastMCP. A preview MCP Registry standard now supports public and proprietary tool discovery. However, MCP's rapid evolution has also introduced architectural gaps, drawing criticism for overlooking established RPC best practices. For production applications, teams should look beyond the hype and apply additional scrutiny by mitigating toxic flows using tools like MCP-Scan and closely monitoring the draft authorization module for security.

40. n8n

Trial

n8n is a fair-code–licensed workflow automation platform, similar to Zapier or Make (formerly Integromat), but built for developers who want a self-hosted, extensible and code-controllable option. It offers a lower-code, visual approach to workflow creation than Apache Airflow, while still supporting custom code in JavaScript or Python.

Its primary use case is integrating multiple services into automated workflows, but it can also connect LLMs with configurable data sources, memory and tools. Many of our teams use n8n to rapidly prototype agentic workflows triggered by chat applications or webhooks, often leveraging its import and export capabilities to generate workflows with AI assistance. As always, we advise caution when using low-code platforms in production. However, n8n's self-hosting and code-defined workflows can mitigate some of those risks.

41. OpenThread

Trial

OpenThread is an open-source implementation of the Thread networking protocol developed by Google. It supports all key features of the Thread specification — including networking layers such as IPv6, 6LoWPAN and LR-WPAN — as well as mesh network capabilities that allow a device to function as both a node and a border router. OpenThread runs on a wide range of hardware platforms, leveraging a flexible abstraction layer and integration hooks that enable vendors to incorporate their own radio and cryptographic capabilities. This mature protocol is widely used in commercial products and, in our experience, has proven reliable for building diverse IoT solutions — from battery-operated, low-power devices to large-scale mesh sensor networks.

42. AG-UI Protocol

Assess

AG-UI is an open protocol and library designed to standardize communication between rich user interfaces and agents. Focused on direct user-facing agents, it uses middleware and client integrations to generalize across any frontend and backend. The protocol defines a consistent way for back-end agents to communicate with front-end applications, enabling real-time, stateful collaboration between AI and human users. It supports multiple transport protocols, including SSE and WebSockets, and provides standardized event types to represent different states of agent execution. Built-in support is available for popular agentic frameworks such as LangGraph and Pydantic AI, with community integrations for others.

43. Agent-to-Agent (A2A) Protocol

Assess

Agent2Agent (A2A) is a protocol that defines a standard for communication and interaction among agents in complex, multi-agent workflows. It uses Agent Cards to describe the key elements of inter-agent communication, including skill discovery and the specification of transport and security schemes. A2A complements the Model Context Protocol (MCP) by focusing on agent-to-agent communication without exposing internal details such as an agent's state, memory or internal.

The protocol promotes best practices such as an asynchronous-first approach for long-running tasks, streaming responses for incremental updates and secure transport with HTTPS, authentication and authorization. SDKs are available in Python, JavaScript, Java and C# to facilitate rapid adoption. Although relatively new, A2A enables teams to build domain-specific agents that can collaborate to form complex workflows, making it a strong option for such scenarios.

44. Amazon S3 Vectors

Assess

Amazon S3 Vectors extends the S3 object store with native vector capabilities, offering built-in vector storage and similarity search functionality. It integrates seamlessly with the AWS ecosystem, including Amazon Bedrock and OpenSearch, and provides additional features such as metadata filtering and governance via IAM. While still in preview and subject to restrictions and limitations, we find its value proposition compelling. This cost-effective, accessible approach to vector storage could enable a range of applications that involve large data volumes and where low latency is not the primary concern.

45. Ardoq

Assess

Ardoq is an enterprise architecture (EA) platform that enables organizations to build, manage and scale their architecture knowledge bases so they can plan more effectively for the future. Unlike traditional static documentation, which is prone to drift and siloing, Ardoq's data-driven approach pulls information from existing systems to create a dynamic knowledge graph that stays up to date as the landscape evolves. One feature we've found particularly useful is Ardoq Scenarios, which allows you to visually model and define what-if future states using a branching and merging approach similar to Git. Organizations pursuing architectural transformation should assess dedicated EA platforms like Ardoq for their potential to streamline and accelerate this process.

46. CloudNativePG

Assess

CloudNativePG is a Kubernetes Operator that simplifies hosting and managing highly available PostgreSQL clusters in Kubernetes. Running a stateful service like PostgreSQL on Kubernetes can be complex, requiring deep knowledge of both Kubernetes and PostgreSQL replication. CloudNativePG abstracts much of this complexity by treating the entire PostgreSQL cluster as a single, configurable declarative resource. It provides seamless primary/standby architecture using native streaming replication and includes high-availability features out of the box, including self-healing capabilities, automated failover that promotes the most aligned replica and automatic recreation of failed replicas. If you're looking to host PostgreSQL on Kubernetes, CloudNativePG is a solid place to start.

47. Coder

Assess

Coder is a platform for quickly provisioning standardized coding environments, following the development environments in the cloud practice we've described before. Compared with similar tools such as Gitpod (now rebranded as Ona) and GitHub Codespaces, Coder offers greater control over workstation customization through Terraform. It hosts workstations on your own infrastructure, whether in the cloud or in a data center, rather than on a vendor's servers. This approach provides more flexibility, including the ability to run AI coding agents and access internal organizational systems. However, this flexibility comes with tradeoffs: More effort to set up and maintain workstation templates and greater responsibility for managing data security risks in agentic workflows.

48. Graft

Assess

Graft is a transactional storage engine that enables strongly consistent and efficient data synchronization across edge and distributed environments. It achieves this by using lazy replication to sync data only on demand, partial replication to minimize bandwidth consumption and serializable snapshot isolation to guarantee data integrity. We've mentioned Electric in the Radar for a similar use case, but we see Graft as unique in turning object storage into a transactional system that supports consistent page-level updates without imposing a data format. This makes it well-suited to powering local-first mobile applications, managing complex cross-platform synchronization and serving as the backbone for stateless replicas in serverless or embedded systems.

49. groundcover

Assess

groundcover is a cloud-native observability platform that unifies logs, traces, metrics and Kubernetes events in a single pane of glass. It leverages eBPF to capture granular observability data with zero code instrumentation — that is, without inserting agents or SDKs into application code. groundcover's eBPF sensor runs on a dedicated node in each monitored cluster, operating independently from the applications it observes. Key features include deep kernel-level visibility, a bring-your-own-cloud (BYOC) architecture for data privacy and a data volume–agnostic pricing model that keeps costs predictable.

50. Karmada

Assess

Karmada ("Kubernetes Armada") is a platform for orchestrating workloads across multiple Kubernetes clusters, clouds and data centers. Many teams currently deploy across clusters using GitOps tools like Flux or ArgoCD combined with custom scripts, so a purpose-built solution is welcome. Karmada leverages Kubernetes-native APIs, requiring no changes to applications already built for cloud-native environments. It offers advanced scheduling capabilities for multi-cloud management, high availability, failure recovery and traffic scheduling.

Karmada is still relatively new, so it's important to assess the maturity of the features your team depends on. As a CNCF project, however, it has strong momentum, and several of our teams are already using it successfully. Note that certain areas — such as networking, state and storage management across clusters — are outside Karmada's scope. Most teams will still need a service mesh like Istio or Linkerd for traffic handling and should plan how to manage stateful workloads and distributed data.

51. OpenFeature

Assess

As businesses scale, feature flag management often becomes increasingly complex; teams need an abstraction layer that goes beyond the simplest possible feature toggle. OpenFeature provides this layer through a vendor-agnostic, community-driven API specification that standardizes how feature flags are defined and consumed, decoupling application code from the management solution. This flexibility allows teams to switch providers easily — from basic setups using environment variables or in-memory configurations up to mature platforms like ConfigCat or LaunchDarkly. However, one critical caution remains: teams must manage different categories of flags separately and with discipline to avoid flag proliferation, application complexity and excessive testing overhead.

52. Oxide

Assess

Building and operating private infrastructure is complex. That's one of the main reasons public cloud is the default for most organizations. However, for those that need it, Oxide offers an alternative to assembling and integrating hardware and software from scratch. It provides prebuilt racks with compute, networking and storage, running fully integrated system software. Teams can manage resources through Oxide's IaaS APIs using Terraform and other automation tools — what Oxide calls on-premises elastic infrastructure.

Dell and VMware's VxRail, Nutanix and HPE SimpliVity also provide hyper-converged infrastructure (HCI) solutions, but what distinguishes Oxide is its purpose-built approach. It designs the entire stack — from circuit boards and power supplies to firmware — instead of assembling components from different vendors. Oxide has also developed and open-sourced Hubris, a lightweight, memory-protected, message-passing kernel written in Rust for embedded systems, along with other Rust-based infrastructure projects. We also appreciate that Oxide sells their equipment and software without license fees.

53. Restate

Assess

Restate is a durable execution platform designed to address complex distributed system challenges when building stateful, fault-tolerant applications. It logs every step via execution journaling, ensuring fault-tolerance, reliable recovery and exactly-once communication across services. The platform's key architectural advantage lies in separating application logic into three durable service types: Basic Services for stateless functions; Virtual Objects to model concurrent, stateful entities; and Workflows to orchestrate complex, multi-step processes. We've been carefully assessing Restate in a large insurance system and are quite happy with its performance so far.

54. SkyPilot

Assess

SkyPilot is an open-source platform for running and scaling AI workloads on-premises or in the cloud. Developed by the Sky Computing Lab at UC Berkeley, SkyPilot acts as an intelligent broker, automatically finding and provisioning the cheapest, most available GPUs across major clouds and Kubernetes clusters, often cutting compute costs. For infrastructure teams, it simplifies running AI on Kubernetes by offering Slurm-like ease of use, cloud-native robustness, direct SSH access to pods and features such as gang scheduling and multi-cluster support for seamless scaling of training or inference workloads.

55. StarRocks

Assess

StarRocks is an analytical database that redefines real-time business intelligence by combining the speed of traditional OLAP systems with the flexibility of a modern data lakehouse. It achieves sub-second query latency at massive scale through a SIMD-optimized execution engine, columnar storage and a sophisticated cost-based optimizer. This high-performance architecture allows users to run complex analytics directly on open data formats such as Apache Iceberg, without pre-computation or data copying. While there are many platforms in this space, we see StarRocks as a strong candidate for cost-effective solutions that require both extreme concurrency and consistent, up-to-the-second data freshness.

56. Uncloud

Assess

Uncloud is a lightweight container orchestration and clustering tool that enables developers to take Docker Compose applications to production, offering a simple, cloud-like experience without the operational overhead of Kubernetes. It achieves cross-machine scaling and zero-downtime deployments by automatically configuring a secure WireGuard mesh network for communication and using the Caddy reverse proxy to provide automatic HTTPS and load balancing. Uncloud's main architectural advantage is its fully decentralized design, which eliminates the need for a central control plane and ensures cluster operations remain functional even if individual machines go offline. With Uncloud, you can freely mix and match cloud VMs and bare-metal servers into a unified and cost-effective computing environment.

Tools

Adopt

57. ClickHouse
58. NeMo Guardrails
59. pnpm
60. Pydantic

Trial

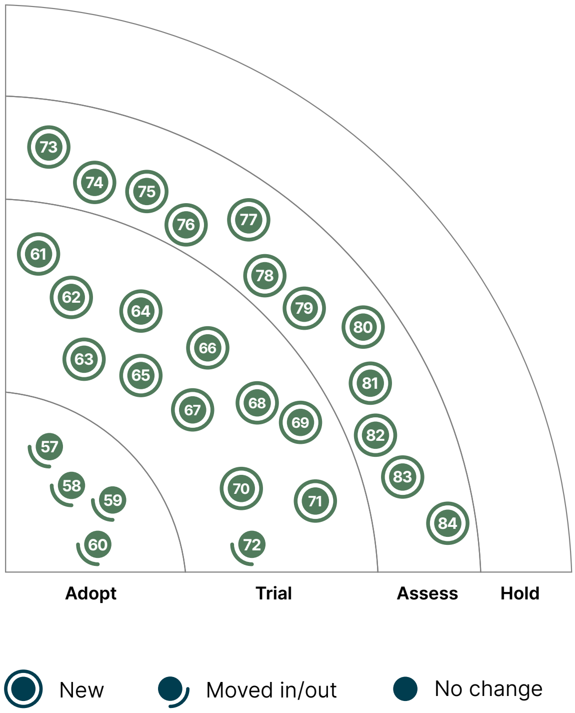
61. AI Design Reviewer
62. Barman
63. Claude Code
64. Cleanlab
65. Context7
66. Data Contract CLI
67. Databricks Assistant
68. Hoppscotch
69. NVIDIA DCGM Exporter
70. RelationalAI
71. UX Pilot
72. v0

Assess

73. Augment Code
74. Azure AI Document Intelligence
75. Docling
76. E2B
77. Helix editor
78. Kueue
79. MCP-Scan
80. oRPC
81. Power user for dbt
82. Serena
83. SweetPad
84. Tape/Z (Tools for Assembly Program Exploration for Z/OS)

Hold

—



57. ClickHouse

Adopt

ClickHouse is an open-source, distributed columnar online analytical processing (OLAP) database for real-time analytics. It has matured into a highly performant and scalable engine capable of handling large-scale data analytics. Its incremental materialized view, efficient query engine and strong data compression make it ideal for interactive queries. Built-in support for approximate aggregate functions enables trade-offs between accuracy and performance, which is especially useful for high-cardinality analytics. The addition of the S3 storage engine and MergeTree allows separation of storage and compute, using S3-compatible storage for ClickHouse tables. We've also found ClickHouse to be an excellent backend for OpenTelemetry data and crash analytics tools like Sentry. For teams seeking a fast, open-source analytics engine, ClickHouse is an excellent choice.

58. NeMo Guardrails

Adopt

NeMo Guardrails is an open-source toolkit from NVIDIA that makes it easy to add programmable safety and control mechanisms to LLM-based conversational applications. It ensures outputs remain safe, on-topic and compliant by defining and enforcing behavioral rules. Developers use Colang, a purpose-built language, to create flexible dialogue flows and manage conversations, enforcing predefined paths and operational procedures. NeMo Guardrails also provides an asynchronous-first API for performance and supports safeguards for content safety, security and moderation of inputs and outputs. We're seeing steady adoption across teams building applications that range from simple chatbots to complex agentic workflows. With its expanding feature set and maturing coverage of common LLM vulnerabilities, we're moving NeMo Guardrails to Adopt.

59. pnpm

Adopt

Since the last Radar, we've continued to receive positive feedback about pnpm from teams. pnpm is a Node.js package manager that delivers significant performance improvements over alternatives, both in speed and disk space efficiency. It hard-links duplicate packages from multiple projects' `node_modules` folders to a single location on disk and supports incremental, file-level optimizations that further boost performance. Because pnpm offers a much faster feedback loop with minimal compatibility issues, it has become our default choice for Node.js package management.

60. Pydantic

Adopt

Pydantic is a Python library that uses standard type hints to define data models and enforce data schemas at run time. Originally, type annotations were added to Python for static analysis, but their growing versatility has led to broader uses, including run-time validation. Built on a fast Rust core, it provides efficient data validation, parsing and serialization.

While it's best known for web API development, Pydantic has also become essential in LLM applications. We typically use the structured output from LLMs technique to manage the unpredictable nature of LLMs. By defining a strict data schema, it acts as a safety net for the unpredictable nature of model output — converting free-form text responses into deterministic, type-safe Python objects (e.g., JSON). This approach, often implemented through Pydantic AI or LangChain, turns potentially brittle LLM interactions into reliable, machine-readable data contracts. Our teams have successfully used Pydantic in production to extract structured representations from unstructured documents, ensuring the output conforms to a valid structure. Given its maturity, performance and reliability, Pydantic is now our default choice for production-level Python AI applications.

61. AI Design Reviewer

Trial

AI Design Reviewer is a Figma plugin for conducting design audits or heuristic evaluations and collecting actionable feedback on existing or new designs. Its audits cover UX critiques, UI inconsistencies, accessibility gaps, content quality and edge-case scenarios. Beyond identifying issues, it provides domain-aware recommendations that help teams build a shared design vocabulary and rationale behind design choices. Our teams used AI Design Reviewer to analyze legacy designs — identifying positive experiences to retain and negative ones to address — which informed UX goals for redesigns. It has also served as a peer-review substitute, offering early feedback on new designs before development handoff.

62. Barman

Trial

Barman (Backup and Recovery Manager) is an open-source tool for managing backups and disaster recovery of PostgreSQL servers. It supports the full disaster recovery process, simplifying the creation of physical backups through a variety of methods, organizing them into a comprehensive catalog and restoring backups to a live server with point-in-time recovery capabilities. We've found Barman to be powerful and easy to use, and have been impressed at the speed of point-in-time recovery operations during migration activities. We've also found it capable for scheduled backups, with an ability to handle complex, mixed configurations of scheduling and retention.

63. Claude Code

Trial

Anthropic's Claude Code is an agentic AI coding tool that provides a natural language interface and agentic execution model for planning and implementing complex, multi-step workflows. Released less than a year ago, it has already been widely adopted by developers inside and outside Thoughtworks, leading us to place it in Trial. Console-based coding agents such as OpenAI's Codex CLI, Google's Gemini CLI and the open-source OpenCode have been released, while IDE-based assistants like Cursor, Windsurf and GitHub Copilot now include agent modes. Even so, Claude Code remains a favorite. We see teams using it not only to write and modify code but also as a general-purpose AI agent for managing specifications, stories, configuration, infrastructure and documentation.

Agentic coding shifts the developer's focus from writing code to specifying intent and delegating implementation. While this can accelerate development cycles, it can also lead to complacency with AI-generated code, which in turn may result in code that is harder to maintain and evolve — for both humans and AI agents. It's therefore essential for teams to rigorously manage how Claude Code works, using techniques such as context engineering, curated shared instructions and potentially teams of coding agents.

64. Cleanlab

Trial

In the data-centric AI paradigm, improving data set quality often delivers greater performance gains than tuning the model itself. Cleanlab is an open-source Python library designed to address this challenge by automatically identifying common data issues — such as mislabeling, outliers and duplicates — across text, image, tabular and audio data sets. Built on the principle of confident learning, Cleanlab leverages model-predicted probabilities to estimate label noise and quantify data quality.

This model-agnostic approach enables developers to diagnose and correct data set errors, then retrain models for improved robustness and accuracy. Our teams have used Cleanlab successfully in production, confirming its effectiveness in real-world settings. We recommend it as a valuable tool for promoting data standardization and improving data set quality in AI engineering projects.

65. Context7

Trial

Context7 is an MCP server that addresses inaccuracies in AI-generated code. While LLMs rely on outdated training data, Context7 ensures they generate accurate, up-to-date and version-specific code for the libraries and frameworks used in a project. It does this by pulling the latest documentation and functional code examples directly from framework source repositories and injecting them into the LLM's context window at the moment of prompting. In our experience, Context7 has greatly reduced code hallucinations and reliance on stale training data. You can configure it with AI code editors such as Claude Code, Cursor or VS Code to generate, refactor or debug framework-dependent code.

66. Data Contract CLI

Trial

Data Contract CLI is an open-source command-line tool designed for working with the Data Contract specification. It helps you create and edit data contracts and, critically, lets you validate data against its contract, which is essential for ensuring the integrity and quality of your data products.

The CLI offers broad support for multiple schema definitions (Avro, SQL DDL, Open Data Contract Standard, etc.) and can compare different contract versions to immediately detect breaking changes. We've found it especially useful in the data mesh space to operationalize contract governance between data products via CI/CD integration. This approach reduces manual errors and ensures data quality, integrity and compatibility in data exchanges across services.

67. Databricks Assistant

Trial

Databricks Assistant is an AI-powered conversational tool integrated directly into the Databricks platform, acting as a contextual pair programmer for data professionals. Unlike general-purpose coding assistants, it benefits from a native understanding of the Databricks environment and data context, including metadata from the Unity Catalog. The Assistant goes beyond generating code snippets; it can craft complex, multi-step SQL and Python queries, diagnose errors and provide detailed, workspace-specific explanations. For organizations already invested in the Databricks ecosystem, it can accelerate productivity and lower the barrier to entry for complex data tasks.

68. Hoppscotch

Trial

Hoppscotch is a lightweight open-source tool for API development, debugging, testing and sharing. It supports multiple protocols — including HTTP, GraphQL and WebSocket — and offers cross-platform clients for web, desktop and CLI environments.

While the API tooling space is crowded with alternatives like Postman, Insomnia and Bruno, Hoppscotch stands out for its lightweight footprint and privacy-friendly design. It omits analytics, uses local-first storage and supports self-hosting. It's a strong choice for organizations seeking an intuitive way to share API scripts while maintaining strong data privacy.

69. NVIDIA DCGM Exporter

Trial

NVIDIA DCGM Exporter is an open-source tool that helps teams monitor distributed GPU training at scale. It converts proprietary telemetry from the NVIDIA Data Center GPU Manager (DCGM) into open formats compatible with standard monitoring systems. The Exporter exposes critical real-time metrics — including GPU utilization, temperature, power and ECC error counts — from both GPU and host servers. This visibility is essential for organizations fine-tuning custom LLMs or running long-duration, GPU-intensive training jobs. The straggler effect — where one slow worker bottlenecks the entire process — can reduce throughput by over 10% and waste up to 45% of allocated GPU hours. Designed for cloud-native, large-scale environments, the DCGM Exporter integrates seamlessly with Prometheus and Grafana, helping ensure every GPU operates within optimal performance bounds.

70. RelationalAI

Trial

When large volumes of diverse data are brought into Snowflake, the inherent relationships and implicit rules within that data can become obscured. Built as a Snowflake Native App, RelationalAI enables teams to build sophisticated models that capture meaningful concepts, define core business entities and embed complex logic directly against Snowflake tables. Its powerful Graph Reasoner allows users to then create, analyze and visualize relational knowledge graphs based on these models. Built-in algorithms help explore graph structures and reveal hidden patterns. For organizations managing massive, fast-changing data sets, constructing a knowledge graph can be essential for proactive monitoring and generating richer and more actionable insights.

71. UX Pilot

Trial

UX Pilot is an AI tool that supports multiple stages of the UX design process — from wireframing to high-fidelity visual design and review. It accepts text or image inputs and can automatically generate screens, flows and layouts. Its Autoflow feature creates user flow transitions, while Deep Design produces richer, more detailed outputs. UX Pilot also includes a Figma plugin that exports generated designs for refinement within standard design tools. Our teams have used UX Pilot for ideation and inspiration, generating multiple options during Crazy 8's exercises and translating project story lists into product vision boards and epic-level design concepts. Tools like UX Pilot also enable non-designers, such as product managers, to create quick prototypes and gather early stakeholder feedback — a growing trend in AI-assisted design workflows.

72. v0

Trial

v0 has evolved since we last featured it in the Radar. It now includes a design mode that further lowers the barrier for product managers to create and tweak self-serve UI prototypes. The latest release introduces an in-house model with large context windows and multimodal capabilities, enabling v0 to generate and improve UIs from both text and visual inputs. Another notable addition is its agentic mode, which allows the system to break down more complex work and select the appropriate model for each. However, this feature is still new, and early feedback has been mixed.

73. Augment Code

Assess

Augment Code is an AI coding assistant that delivers deep, context-aware support across large codebases. It stands out through advanced context engineering that enables rapid code index updates and fast retrieval, even as code changes frequently. Augment supports models such as Claude Sonnet 4 and 4.5 and GPT-5, integrates with GitHub, Jira and Confluence and supports the Model Context Protocol (MCP) for external tool interoperability. It provides turn-by-turn guidance for complex codebase changes — from refactors and dependency upgrades to schema updates — along with personalized in-line completions that reflect project-specific dependencies. Augment also promotes collaboration by allowing teams to query and share code insights directly within Slack.

74. Azure AI Document Intelligence

Assess

Azure AI Document Intelligence (formerly Form Recognizer) extracts text, tables and key-value pairs from unstructured documents and transforms them into structured data. It uses pre-trained deep learning models to interpret layouts and semantics, and custom models can be trained through a no-code interface for specialized formats. In some cases, however, power users may require a custom fine-tuning interface instead.

One of our teams reported that ADI significantly reduced manual data entry, improved data accuracy and accelerated reporting, leading to faster data-driven decisions. Like Amazon Textract and Google Document AI, it provides enterprise-grade document processing with strong layout understanding. An emerging open-source alternative is IBM's Docling, which offers a more flexible, code-centric approach to structured data extraction. Compared to traditional OCR tools, ADI captures not just text but also structure and relationships, making it easy to integrate into downstream data pipelines. That said, we've observed occasional latency when embedding it into synchronous user workflows, so we recommend using it primarily for asynchronous processing.

75. Docling

Assess

Docling is an open-source Python and TypeScript library for advanced document processing of unstructured data. It addresses the often overlooked "last mile" problem of converting real-world documents — like PDFs and PowerPoints — into clean, machine-readable formats. Unlike traditional extractors, Docling uses a computer vision–based approach to interpret document layout and semantic structure, which makes its output particularly valuable for retrieval-augmented generation (RAG) pipelines. It converts complex documents into structured formats such as JSON or Markdown, supporting techniques like structured output from LLMs. This contrasts with ColPali, which feeds page images directly to a vision-language model for retrieval.

Docling's open-source nature and Python core, built on a custom Pydantic-based data model, provide a flexible, self-hosted alternative to proprietary cloud tools such as Azure Document Intelligence, Amazon Textract and Google Document AI. Backed by IBM Research, the project's rapid development and plug-and-play architecture for integrating with other frameworks like LangGraph make it well worth assessing for teams building production-grade AI-ready data pipelines.

76. E2B

Assess

E2B is an open-source tool for running AI-generated code in secure, isolated sandboxes in the cloud. Agents can use these sandboxes, built on top of Firecracker microVMs, to safely execute code, analyze data, conduct research or operate a virtual machine. This enables you to build and deploy enterprise-grade AI agents with full control and security over the execution environment.

77. Helix editor

Assess

There's been somewhat of a resurgence in simple text editors aiming to replace the command-line favorite Vim. Helix is one such contender in the crowded space alongside Neovim and, more recently, Kakoune. Describing itself — somewhat playfully — as a post-modern text editor, Helix features multiple cursors, Tree-sitter support and integrated Language Server Protocol (LSP) support, which is what first drew our attention. Helix is actively developed with a plugin system on the way. Overall, it's a lightweight modal editor that feels familiar to Vim users but adds a few modern conveniences.

78. Kueue

Assess

Kueue is a Kubernetes-native controller for job queuing that manages quotas and resource consumption. It provides APIs for handling Kubernetes workloads with varying priorities and resource requirements, functioning as a job-level manager that determines when to admit or evict jobs. Designed for efficient resource management, job prioritization and advanced scheduling, Kueue helps optimize workload execution in Kubernetes environments — particularly for ML workloads using tools such as Kubeflow. It works alongside the cluster-autoscaler and kube-scheduler rather than replacing them, focusing on job admission based on order, quota, priority and topology awareness. As part of the Kubernetes Special Interest Group (SIG) ecosystem, Kueue adheres to its development standards.

79. MCP-Scan

Assess

MCP-Scan is a security scanner for Model Context Protocol (MCP) servers that operates in two modes: scan and proxy. In scan mode, it analyzes configurations and tool descriptions to detect known vulnerabilities such as prompt injections, tool poisoning and toxic flows. In proxy mode, MCP-Scan acts as a bridge between agent system and MCP server, continuously monitoring runtime traffic. This mode also enforces custom security rules and guardrails, including tool call validation, PII detection and data flow constraints. The tool provides a proactive security layer for agents, ensuring that even if a malicious prompt is accepted, the agent cannot execute harmful actions. MCP-Scan is a purpose-built security solution for the emerging field of agentic systems.

80. oRPC

Assess

oRPC (OpenAPI Remote Procedure Call) provides end-to-end typesafe APIs in TypeScript while fully adhering to the OpenAPI specification. It can automatically generate a complete OpenAPI spec, simplifying integration and documentation. We've found oRPC particularly strong for integrations. While alternatives such as tRPC and ElysiaJS often require adopting a new framework to gain type safety, oRPC integrates seamlessly with existing Node.js frameworks, including Express, Fastify, Hono and Next.js. This flexibility makes it an excellent choice for teams looking to adopt end-to-end type safety to existing APIs without a disruptive refactor.

81. Power user for dbt

Assess

Power user for dbt is an extension for Visual Studio Code that integrates directly with both dbt and dbt Cloud environments. Since dbt remains one of our favourite tools, anything that improves its usability is a welcome addition to the ecosystem. Previously, developers relied on multiple tools to validate SQL code or inspect model lineage outside the IDE. With this extension, those capabilities are now built into VS Code, offering code autocompletion, real-time query results and visual model and column lineage. This last feature makes it easy to navigate between models. Our teams report the plugin reduces pipeline errors and enhances the overall development experience. If you use dbt, we urge you to take a look at this tool.

82. Serena

Assess

Serena is a powerful coding toolkit that equips coding agents such as Claude Code with IDE-like capabilities for semantic code retrieval and editing. By operating at the symbol level and understanding the relational structure of code, Serena greatly improves token efficiency. Instead of reading entire files or relying on crude string replacements, coding agents can use precise Serena tools such as find_symbol, find_referencing_symbols and insert_after_symbol to locate and edit code. Although the impact is minimal on small projects, this efficiency is extremely valuable as the codebase grows.

83. SweetPad

Assess

The SweetPad extension enables developers to use VS Code or Cursor for the entire Swift application development lifecycle on Apple platforms. It eliminates the need to constantly switch to Xcode by integrating essential tools such as xcodebuild, xcode-build-server, and swift-format. Developers can build, run and debug Swift applications for iOS, macOS and watchOS directly from their IDEs, while also managing simulators and deploying to devices without opening Xcode.

84. Tape/Z (Tools for Assembly Program Exploration for Z/OS)

Assess

Tape/Z (Tools for Assembly Program Exploration for Z/OS) is an evolving toolkit for analyzing mainframe HLASM (High-Level Assembler) code. Developed by a Thoughtworker, it provides capabilities such as parsing, control flow graph construction, dependency tracing and flowchart visualization. We've long noted the scarcity of open, community-driven tools in the mainframe space, where most options remain proprietary or tied to vendor ecosystems. Tape/Z helps close that gap by offering accessible, scriptable analysis capabilities. Alongside COBOL REKT — a companion toolkit for COBOL that we've also used multiple times with clients — it represents encouraging progress toward modern, developer-friendly tooling for mainframe systems.

Languages and Frameworks

Adopt

85. Fastify
86. LangGraph
87. vLLM

Trial

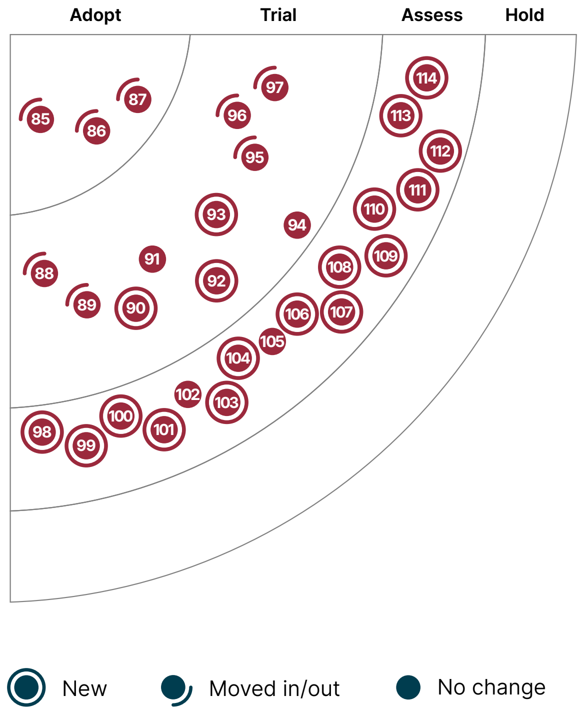
88. Crossplane
89. DeepEval
90. FastMCP
91. LiteLLM
92. MLForecast
93. Nuxt
94. Phoenix
95. Presidio
96. Pydantic AI
97. Tauri

Assess

98. Agent Development Kit (ADK)
99. Agno
100. assistant-ui
101. AutoRound
102. Browser Use
103. DeepSpeed
104. Drizzle
105. Java post-quantum cryptography
106. kagent
107. LangExtract
108. Langflow
109. LMCache
110. Mem0
111. Open Security Control Assessment Language (OSCAL)
112. OpenInference
113. Valibot
114. Vercel AI SDK

Hold

—



85. Fastify

Adopt

We continue to have a positive experience with Fastify — a fast, unopinionated, low-overhead web framework for Node.js. It provides all the essential capabilities of a minimal web framework — including parsing, validation and serialization — along with a robust plugin system and strong community support. Our teams have seen no significant downsides in using Fastify over alternatives such as Express.js, while also gaining measurable performance improvements, making it a compelling choice for minimal web development in Node.js.

86. LangGraph

Adopt

LangGraph is an orchestration framework for building stateful multi-agent applications using LLMs. It provides low-level primitives such as nodes and edges, along with built-in features that give developers granular control over agent workflows, memory management and state persistence. This means developers can start with a simple pre-built graph and scale to complex, evolving agent architectures. With support for streaming, advanced context management and resilience patterns like model fallbacks and tool error handling, LangGraph enables you to build robust, production-grade agentic applications. Its graph-based approach ensures predictable, customizable workflows and simplifies debugging and scaling. Our teams have had strong results using LangGraph to build multi-agent systems thanks to its lightweight and modular design.

87. vLLM

Adopt

vLLM is a high-throughput, memory-efficient inference engine for LLMs that can run in the cloud or on-premises. It supports multiple model architectures and popular open-source models. Our teams deploy dockerized vLLM workers on GPU platforms such as NVIDIA DGX and Intel HPC, hosting models including Llama 3.1 (8B and 70B), Mistral 7B and Llama-SQL for developer coding assistance, knowledge search and natural language database interactions. vLLM is compatible with the OpenAI SDK standard, enabling consistent model serving. Azure's AI Model Catalog uses a custom inference container built on vLLM to boost serving performance, with vLLM as the default inference engine due to its high throughput and efficient memory management. The vLLM framework has become a go-to framework for large-scale model deployments.

88. Crossplane

Trial

Since its last appearance on the Radar, Crossplane adoption has continued to grow, particularly for extending Kubernetes clusters. In our work, we've found Crossplane excels in specific use cases rather than as a general-purpose infrastructure-as-code (IaC) tool. Our earlier observations still hold true: Crossplane works best as a companion to workloads deployed within Kubernetes, not as a full replacement for tools like Terraform. Teams that went "all-in" on Crossplane as their primary IaC solution often struggled, whereas those using it pragmatically — for targeted, custom use-cases — saw strong results. Offloading resource lifecycle management to Crossplane while using XRD APIs for lightweight customization has proven especially effective. Crossplane is particularly valuable when managing resources whose lifecycles are straightforward but not native to Kubernetes. While it can now create Kubernetes clusters — a capability that was previously missing — we advise caution against adopting Crossplane as a complete Terraform replacement. In our experience, it works best when IaC serves as the foundational layer, with Crossplane layered on top for specialized requirements.

89. DeepEval

Trial

DeepEval is an open-source, Python-based evaluation framework for assessing LLM performance. It can be used to evaluate retrieval-augmented generation (RAG) and other applications built with frameworks such as LlamaIndex or LangChain, as well as to baseline and benchmark models. DeepEval goes beyond word-matching scores, assessing accuracy, relevance and consistency to provide more reliable evaluation in real-world scenarios. It includes metrics such as hallucination detection, answer relevancy and hyperparameter optimization and supports GEval for creating custom, use case–specific metrics. Our teams are using DeepEval to fine-tune agentic outputs using the LLM as a judge technique. It integrates with pytest and CI/CD pipelines, making it easy to adopt and valuable for continuous evaluation. For teams building LLM-based applications in regulated environments, Inspect AI, developed by the UK AI Safety Institute, offers an alternative with stronger focus on auditing and compliance.

90. FastMCP

Trial

The Model Context Protocol (MCP) is rapidly becoming a standard for providing context and tooling for LLM applications. However, implementing an MCP server typically involves substantial boilerplate for setup, protocol handling and error management. FastMCP is a Python framework that simplifies this process by abstracting away protocol complexity and allowing developers to define MCP resources and tooling through intuitive Python decorators. This abstraction enables teams to focus on business logic, resulting in cleaner, more maintainable MCP implementations.

While FastMCP 1.0 is already incorporated into the official SDK, the MCP standard continues to evolve quickly. We recommend monitoring the 2.0 release and ensuring teams stay aligned with changes to the official specification.

91. LiteLLM

Trial

LiteLLM is an SDK that provides seamless integration with multiple LLM providers through a standardized OpenAI API format. It supports a wide range of providers and models, offering a unified interface for text completion, embeddings and image generation. By abstracting provider-specific API differences, LiteLLM simplifies integration and automatically routes requests to the correct model endpoint. It also includes production-grade features such as guardrails, caching, logging, rate limiting and load balancing through its proxy framework. As organizations embed AI-powered applications more deeply into workflows, governance and observability become essential. Our teams have been using LiteLLM as an AI gateway to standardize, secure and gain visibility into enterprise-wide AI usage.

92. MLForecast

Trial

MLForecast is a Python framework and library for time series forecasting that applies machine learning models to large-scale data sets. It simplifies the typically complex process of automated feature engineering — including lags, rolling statistics and date-based features — and is one of the few libraries with native support for distributed computing frameworks like Spark and Dask, ensuring scalability. It also supports probabilistic forecasting using methods such as conformal prediction, providing quantitative measures of forecast uncertainty. In our evaluation, MLForecast scaled efficiently to millions of data points and consistently outperformed comparable tools. For teams looking to rapidly operationalize time series forecasting on high-volume data, MLForecast is a compelling choice.

93. Nuxt

Trial

Nuxt is an opinionated meta-framework built on top of Vue.js for creating full-stack web applications, often known as the "Next.js for Vue.js." Similar to its React counterpart, Nuxt provides SEO-friendly capabilities such as pre-rendering, server-side rendering (SSR) and metadata management, making it a strong choice for building performance-oriented, SEO-optimized websites on the Vue.js stack.

Nuxt is backed by Vercel, the same company behind Next.js, and supported by a strong community and ecosystem of official and third-party modules. These modules simplify integration of features such as image processing, sitemaps and Tailwind CSS. Nuxt is a good choice for teams looking for a comprehensive, opinionated framework for building SEO-friendly applications with Vue.js.

94. Phoenix

Trial

We continue to have positive experiences with Phoenix, a server-side web MVC framework written in Elixir. Phoenix builds on the rapid application development and developer experience learnings of Ruby on Rails, while also advancing into functional programming paradigms.

In this volume, we're highlighting the release of Phoenix LiveView 1.0. LiveView is an HTML-over-the-wire solution — similar to HTMX or Hotwire — that enables developers to build rich, real-time user experiences entirely with server-rendered HTML. While similar technologies usually handle partial updates through HTML switching, LiveView provides a full component-based architecture via LiveComponents, offering composition, props passing, state management and lifecycle hooks akin to React or Vue.js. LiveView combines the productivity and scalability of Phoenix with robust complexity management, making it well-suited for building highly interactive frontends without the need for a full JavaScript framework.

95. Presidio

Trial

Presidio is a data protection SDK for identifying and anonymizing sensitive data in structured and unstructured text. It detects personally identifiable information (PII) such as credit card numbers, names and locations using named entity recognition, regular expressions and rule-based logic. Presidio supports custom entity recognizers and de-identification pipelines, allowing organizations to tailor it to their privacy and compliance needs. Our teams have used Presidio in enterprise environments with strict data-sharing controls when integrating with LLMs. While it automates the detection of sensitive information, it isn't foolproof and may miss or misidentify entities. Teams should exercise caution when relying on its results.

96. Pydantic AI

Trial

Pydantic AI continues to prove itself as a stable, well-supported, open-source framework for building GenAI agents in production. Built on the trusted Pydantic foundation, it offers strong type safety, first-class observability through OpenTelemetry and built-in evaluation tooling. The release of version 1.0

on September 4, 2025 marked a significant milestone in its maturity. Since then, we've found it reliable and widely adopted for its simplicity and maintainability, joining the ranks of other popular agent frameworks like LangChain and LangGraph.

Recent updates have made it easier to implement Model Context Protocol (MCP) servers and clients, with added support for emerging standards such as AG-UI and A2A. With its clean API and growing ecosystem, Pydantic AI has become a compelling choice for our teams building production-ready GenAI applications in Python.

97. Tauri

Trial

Tauri is a framework for building high-performance desktop applications using a single web UI codebase. Unlike traditional web wrappers such as Electron, Tauri is built on Rust and leverages the operating system's native webview, resulting in smaller binaries and stronger security. We first evaluated Tauri several years ago; since then, it has expanded beyond desktop to support iOS and Android. The latest version introduces a more flexible permission and scope model, replaces the older permission list and features a hardened inter-process communication (IPC) layer that supports raw data transfer and improves performance. These updates are backed by an external security audit. Together with official distribution guidelines for major app stores, these improvements further strengthen Tauri's position in the cross-platform development space.

98. Agent Development Kit (ADK)

Assess

Agent Development Kit (ADK) is a framework for developing and deploying AI agents that applies modern software engineering discipline rather than relying solely on prompting. It introduces familiar abstractions such as classes, methods, workflow patterns and CLI support. Compared to frameworks like LangGraph or CrewAI, ADK's strength lies in its deep integration with Google's AI infrastructure, providing enterprise-ready grounding, data access and monitoring. It's also designed for interoperability, supporting tool wrappers and the A2A protocol for agent-to-agent communication. For organizations already invested in GCP, ADK presents a promising foundation for building scalable, secure and manageable agentic architecture. Though still early in its evolution, it signals Google's direction toward a native, full-stack agent development environment. We recommend keeping a close eye on its maturity and ecosystem growth.

99. Agno

Assess

Agno is a framework for building, running and managing multi-agent systems. It offers the flexibility to create fully autonomous agents or controlled, step-based workflows, with built-in support for human-in-the-loop, session management, memory and knowledge. We appreciate its focus on efficiency, with impressive agent startup times and low memory consumption. Agno also comes with its runtime, AgentOS, a FastAPI application with an integrated control plane for streamlined testing, monitoring and management of agentic systems.

100. assistant-ui

Assess

assistant-ui is an open-source TypeScript and React library for AI chat interfaces. It handles the complex parts of chat UI implementation — such as streaming, state management for message editing and branch switching and common UX features — while allowing developers to design their own components using Radix primitives. It supports integration with popular runtimes, including Vercel AI SDK and LangGraph, and offers customizable runtime solutions for complex use cases. We've successfully built a simple chat interface with assistant-ui and have been pleased with the results.

101. AutoRound

Assess

Intel's AutoRound is an advanced quantization algorithm for compressing large AI models, such as LLMs and vision language models (VLMs), with minimal loss of accuracy. It reduces model size to ultra-low bit widths (2–4 bits) using sign-gradient descent optimization and applies mixed bit widths across layers for optimal efficiency. This quantization process is also remarkably fast: You can quantize a 7-billion-parameter model in just minutes on a single GPU. Since AutoRound integrates with popular inference engines such as vLLM and Transformers, it's an attractive option for quantizing models.

102. Browser Use

Assess

Browser Use is an open-source Python library that enables LLM-based agents to operate web browsers and interact with web applications. It can navigate, enter data, extract text and manage multiple tabs to coordinate actions across applications. The library is particularly useful when AI agents need to access, manipulate or retrieve information from web content. It supports a range of LLMs and leverages Playwright to combine visual understanding with HTML structure extraction for richer web interactions. Our teams integrated Browser Use with the Pytest framework and Allure reporting to explore automated testing with LLMs. Test steps were written in natural language for the agent to execute, capturing screenshots on assertions or failures. The goal was to enable off-hours QA by automatically pulling test cases from Confluence for post-development verification. Early results are promising, though the agent's post-task responses often lack detailed failure descriptions, so require custom error reporting.

103. DeepSpeed

Assess

DeepSpeed is a Python library that optimizes distributed deep learning for both training and inference. For training, it integrates technologies such as the Zero Redundancy Optimizer (ZeRO) and 3D parallelism to efficiently scale models across thousands of GPUs. For inference, it combines tensor, pipeline, expert and ZeRO parallelism with custom kernels and communication optimizations to minimize latency. DeepSpeed has powered some of the world's largest language models, including Megatron-Turing NLG (530B) and BLOOM (176B). It supports both dense and sparse models, delivers high system throughput and allows training or inference across multiple resource-constrained GPUs. The library integrates seamlessly with popular Hugging Face Transformers, PyTorch Lightning and Accelerate, making it a highly effective option for large-scale or resource-limited deep learning workloads.

104. Drizzle

Assess

Drizzle is a lightweight TypeScript ORM. Unlike Prisma ORM, it gives developers both a simple SQL-like API and a more traditional ORM-style query interface. It also supports extracting schemas from existing databases, enabling both database-first and code-first approaches. Drizzle was designed with serverless environments in mind: It has a small bundle size and supports prepared statements, allowing SQL queries to be precompiled so that the database driver executes binary SQL directly instead of parsing queries each time. Its simplicity and serverless support make Drizzle an appealing choice for ORM use in the TypeScript ecosystem.

105. Java post-quantum cryptography

Assess

Quantum computers continues to advance rapidly, with SaaS offerings like AWS Braket now providing access to quantum algorithms across multiple architectures.

Since March, Java 24 has introduced Java post-quantum cryptography, adding support for post-quantum cryptographic algorithms such as ML-KEM and ML-DSA, and .Net 10 has expanded its support as well. Our advice is simple: if you're building software in these languages, begin adopting quantum-safe algorithms now to future-proof your systems.

106. kagent

Assess

Kagent is an open-source framework for running agentic AI inside Kubernetes clusters. It enables LLM-based agents to plan and execute operational tasks such as diagnosing issues, remediating configurations or interacting with observability tools through Kubernetes-native APIs and Model Context Protocol (MCP) integrations. Its goal is to bring "AgentOps" to cloud-native infrastructure by combining declarative management with autonomous reasoning.

As a CNCF Sandbox project, Kagent should be introduced with care, particularly given the risks of granting LLMs operational management capabilities. Techniques such as toxic flow analysis can be especially valuable when assessing and mitigating these risks.

107. LangExtract

Assess

LangExtract is a Python library that uses LLMs to extract structured information from unstructured text based on user-defined instructions. It processes domain-specific materials — such as clinical notes and reports — identifying and organizing key details while keeping each extracted data point traceable to its source. The extracted entities can be exported as a `.jsonl` file, a standard format for language model data and visualized through an interactive HTML interface for contextual review. Our teams evaluated LangExtract for extracting entities to populate a domain knowledge graph and found it effective for transforming complex documents into structured, machine-readable representations.

108. Langflow

Assess

Langflow is an open-source, low-code platform for building and visualizing LLM workflows. Built on top of LangChain, it allows developers to chain prompts, tools, vector databases and memory components through a drag-and-drop interface, while still supporting custom Python code for advanced logic. It's particularly useful for prototyping agentic applications without writing full back-end code. However, Langflow is still relatively new and has some rough edges for production use. Our usual caution around low-code platforms applies here. That said, we like that Langflow's workflows can be defined and versioned as code, which can mitigate some of the drawbacks of low-code platforms.

109. LMCache

Assess

LMCache is a key-value (KV) cache solution that accelerates LLM serving infrastructure. It acts as a specialized caching layer across a pool of LLM inference engines, storing precomputed KV cache entries for texts likely to be processed multiple times, such as chat histories or document collections. By persisting these values on disk, prefill computations can be offloaded from the GPU, reducing time-to-first-token (TTFT) and cutting inference costs across demanding workloads like RAG pipelines, multi-turn chat applications and agentic systems. You can integrate LMCache with major inference servers such as vLLM or NVIDIA Dynamo and think it's worth assessing its impact on your setup.

110. Mem0

Assess

Mem0 is a memory layer designed for AI agents. Naive approaches often store entire chat histories in a database and reuse them in future conversations, which leads to excessive token usage. Mem0 replaces this with a more sophisticated architecture that separates memory into short-term recall and an intelligent long-term layer that extracts and stores only salient facts and relationships. Its architecture combines a vector store for semantic similarity with a knowledge graph for understanding temporal and relational data. This design significantly reduces context token usage while enabling agents to maintain long-term awareness, which is extremely useful for personalization and many other use cases.

111. Open Security Control Assessment Language (OSCAL)

Assess

The Open Security Controls Assessment Language (OSCAL) is an open, machine-readable information exchange format designed to increase automation in compliance and risk management, and help teams move away from text-based manual approaches. Led by the National Institute of Standards and Technology (NIST), OSCAL provides standard representations in XML, JSON and YAML for expressing security controls associated with industry frameworks such as SOC 2 and PCI, as well as government frameworks such as FedRAMP in the United States, Singapore's Cybersecurity Control Catalogue and Australia's Information Security Manual.

While OSCAL has not yet been widely adopted outside the public sector and its ecosystem is still maturing, we're excited by its potential to streamline security assessments, reduce reliance on spreadsheets and box-ticking exercises and even enable automated compliance when incorporated into compliance-as-code and continuous compliance platforms.

112. OpenInference

Assess

OpenInference is a set of conventions and plugins; it's complementary to OpenTelemetry and designed to observe AI applications. It provides standardized instrumentation for machine-learning frameworks and libraries, which helps developers trace LLM invocations along with surrounding context such as vector store retrievals or external tool calls to APIs and search engines. Spans can be exported to any OTEL-compatible collector, ensuring alignment with existing telemetry pipelines. We previously blipped Langfuse, a commonly used LLM observability platform — the OpenInference SDK can log traces into Langfuse and other OpenTelemetry-compatible observability platforms.

113. Valibot

Assess

Valibot is a schema validation library in TypeScript. Like other popular TypeScript validation libraries such as Zod and Ajv, it provides type inference, but its modular design sets it apart. This architecture allows bundlers to perform effective tree shaking and code splitting, including only the validation functions actually used. Valibot can reduce the bundle size by up to 95% compared to Zod in optimal scenarios. It's an appealing choice for schema validation in environments where bundle size is critical, such as client-side validation or serverless functions.

114. Vercel AI SDK

Assess

Vercel AI SDK is an open-source, full-stack toolkit for building AI-powered applications and agents in the TypeScript ecosystem. It consists of two main components: AI SDK Core standardizes model-agnostic LLM calls, supporting text generation, structured object generation and tool calling; AI SDK UI simplifies front-end development with streaming, state management and real-time UI updates in React, Vue, Next.js and Svelte, similar to assistant-ui. For teams already working within the TypeScript and Next.js ecosystem, Vercel AI SDK provides a fast, seamless way to build AI applications with rich, client-side experiences.

Stay up to date with all Radar-related news and insights

Subscribe to the Technology Radar to receive emails every other month for tech insights from Thoughtworks and future Technology Radar releases.

Subscribe now

We are a global technology consultancy that delivers extraordinary impact by blending design, engineering and AI expertise.

For over 30 years, our culture of innovation and technology excellence has helped clients strengthen their enterprise systems, scale with agility and create seamless digital experiences.

We're dedicated to solving our clients' most critical challenges, combining AI and human ingenuity to turn their ambitious ideas into reality.

/thoughtworks

Design. Engineering. AI.