# Technology Radar

**An opinionated guide to today's technology landscape**

Volume 32
April 2025

thoughtworks

Design. Engineering. AI.

# 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.

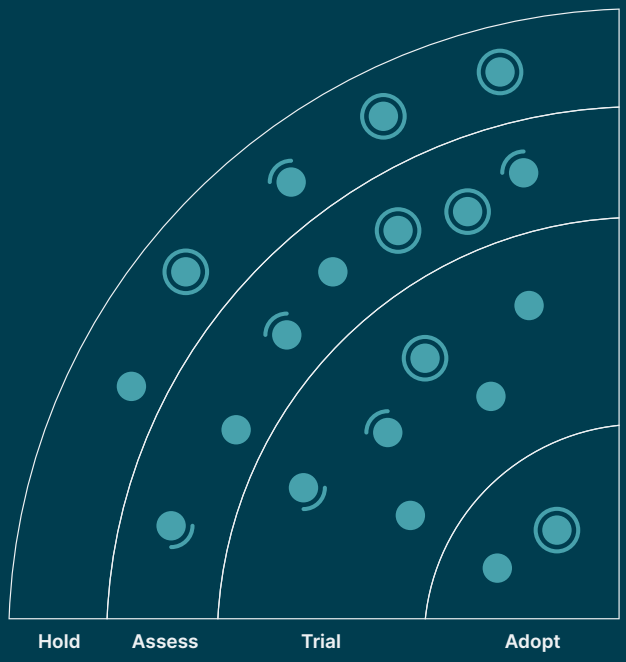
We encourage you to explore these technologies. The Radar is graphical in nature, grouping items into techniques, tools, platforms and languages and frameworks. When Radar items could appear in multiple quadrants, we chose the one that seemed most appropriate. We further group these items in four rings 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 21 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 Bangkok in February 2025.

Rachel Laycock (CTO)

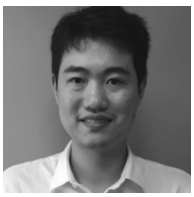
Martin Fowler (Chief Scientist)

Bharani Subramaniam

Birgitta Böckeler

Bryan Oliver

Camilla Falconi Crispim



Chris Chakrit Riddhagni

Effy Elden

Erik Dörnenburg

James Lewis

Ken Mugrage

Maya Ormaza

Mike Mason

Neal Ford

Ni Wang

Nimisha Asthagiri

Pawan Shah

Selvakumar Natesan

Shangqi Liu

Vanya Seth

Will Amaral

# Production Credits

## Editorial Team

- **Willian Amaral** — *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*
- **Kevin Barry** — *Multimedia Specialist*
- **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*
- **Anisha Thampy** — *Visual 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*
- **Linda Horiuchi** — *Public Relations Specialist*
- **Kathryn Jansing** — *Public Relations Specialist*
- **Soumyajit Dey** — *Campaigns and Advertisement Specialist*
- **Anushree Tapuriah** — *Campaigns and Advertisement Specialist*

# Themes

## Supervised agents in coding assistants

Two of our themes highlight the rapid innovation in generative AI, and one of them is about the accelerating capabilities of coding assistants. More and more of these tools allow developers to drive implementation directly from an AI chat within their IDE — a mode also called "agentic", "prompt-to-code" or "chat-oriented programming (CHOP)." In this approach, AI assistants go beyond answering questions or generating small snippets; they navigate and modify code, update tests, execute commands and, in some cases, proactively fix linting and compilation errors. While we remain skeptical of coding agents that promise fully autonomous development of large tasks, we've seen promising results with this supervised approach, where developers still guide and oversee the agent's actions. Cursor, Cline and Windsurf are leading this trend in the IDE-integrated tools space, with GitHub Copilot also advancing. Agentic assistants such as aider, goose and Claude Code are terminal-based alternatives. Despite these advancements, we remain cautious about how this increases complacency with AI-generated code, as in spite of some very good results, we still see a lot of need for steering and vigilance in the code review. With great power...

## Evolving observability

We've seen significant movement in the observability space, driven by the growing complexity of distributed architectures. While observability has long been essential, it continues to evolve alongside the rest of the software development ecosystem. One emerging focus is LLM observability, a critical piece in operationalizing AI. We've seen a surge in tools for monitoring and evaluating LLM performance, including Weights & Biases Weave, Arize Phoenix, Helicone and HumanLoop. Another trend is the integration of AI-assisted observability, where tools leverage AI to enhance analysis and insights. Additionally, the increasing adoption of OpenTelemetry is fostering a more standardized observability landscape, enabling teams to remain vendor-agnostic and more flexible in their tooling choices. Many leading observability tools — such as Alloy, Tempo and Loki — now support OpenTelemetry. The rapid innovation in observability tools demonstrates growing industry awareness of observability's importance, creating a cycle where evolving practices and technologies reinforce each other.

## R in RAG

We expect different aspects of the generative AI ecosystem to evolve at varying rates, and in this edition of the Radar, we see this happening for the R in RAG (retrieval-augmented generation). One of the key interactions with the LLM black box is customizing the prompt's inputs to generate relevant and useful responses. The growing need for effective retrieval in RAG has led to the emergence of new tools and techniques featured in this edition. For example, we discussed corrective RAG, which

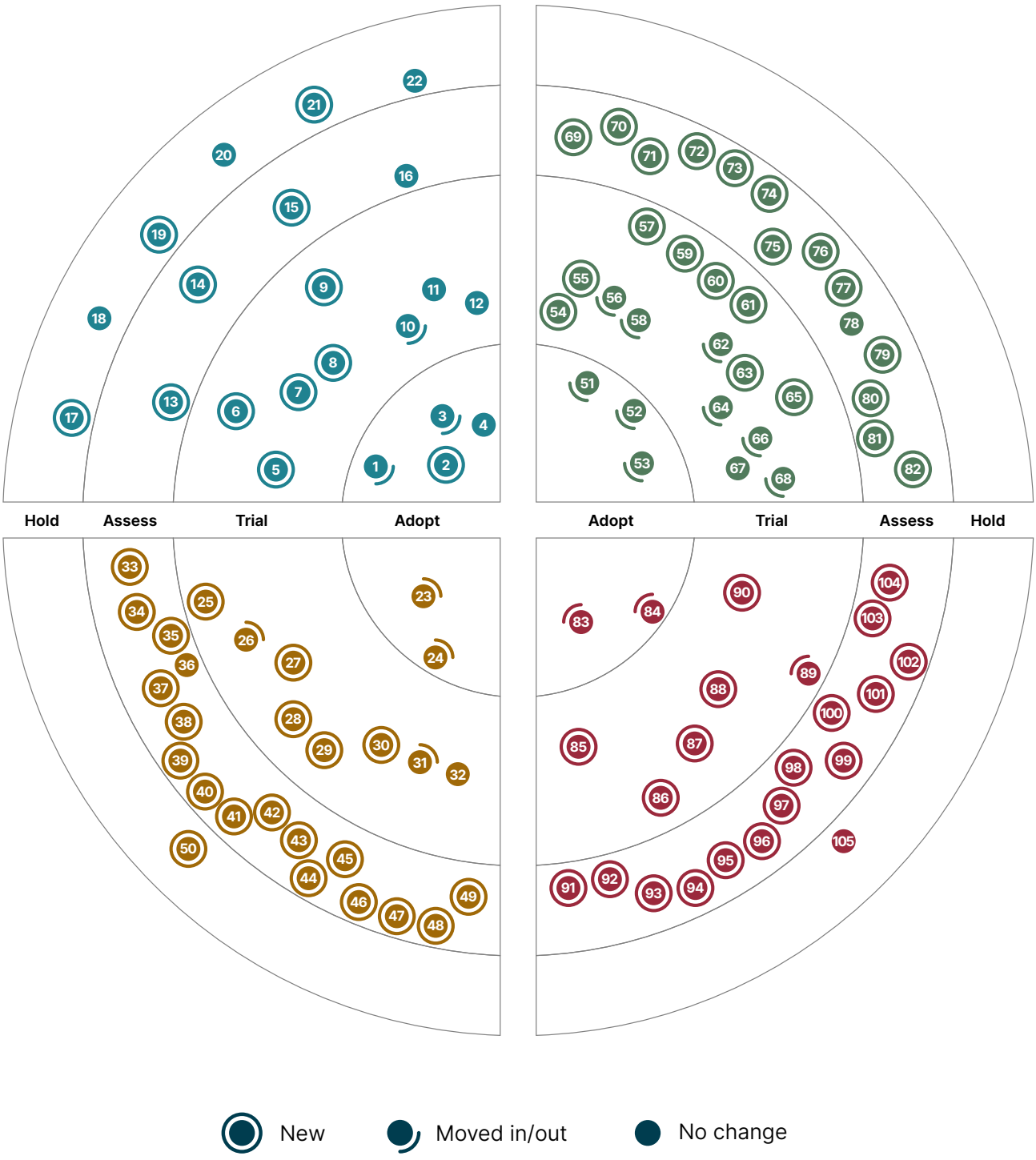
dynamically adjusts responses based on feedback or heuristics; Fusion-RAG, which combines multiple sources and retrieval strategies for more comprehensive and robust responses; and Self-RAG, which avoids the retrieval step altogether, fetching data on demand. We also highlighted FastGraphRAG, which aids understandability by creating human-navigable graphs. Based on our team's discussions and nominations, the R in RAG is a hot topic and evolving rapidly.

## Taming the data frontier

Big data has long been a key concern for the industry, but conversations around this edition of the Radar centered not so much on size but more specifically on handling rich and complex data. With the increasing presence — and importance — of unstructured data in the enterprise, ensuring data is effectively managed and packaged so it can be successfully leveraged for everything from AI applications to customer analytics is today vital for businesses.

This is reflected in many blips: from vector database tools to analytics products like Metabase, it's astonishing how much of the software ecosystem is being driven by what we want and need to do with data. It's not just about tooling though — in this edition we noted data product thinking, a framework that encourages teams to apply the principles of product thinking to the analytic parts of their ecosystem. The emergence of data product thinking is, to some extent, a result of the ongoing challenge of properly leveraging data (something that's been talked about for years, well before the rise of AI) — the fact that it has found its way into the spotlight (and our Radar conversations) demonstrates that the need for data discipline is now as great as it has ever been. Without it, organizations may struggle to innovate and could well be at a commercial disadvantage in the medium and long term.

# The Radar



# The Radar

## Techniques

**Adopt**

1. Data product thinking
2. Fuzz testing
3. Software Bill of Materials
4. Threat modeling

**Trial**

5. API request collection as API product artifact
6. Architecture advice process
7. GraphRAG
8. Just-in-time privileged access management
9. Model distillation
10. Prompt engineering
11. Small language models
12. Using GenAI to understand legacy codebases

**Assess**

13. AI-friendly code design
14. AI-powered UI testing
15. Competence envelope as a model for understanding system failures
16. Structured output from LLMs

**Hold**

17. AI-accelerated shadow IT
18. Complacency with AI-generated code
19. Local coding assistants
20. Replacing pair programming with AI
21. Reverse ETL
22. SAFe™

## Platforms

**Adopt**

23. GitLab CI/CD
24. Trino

**Trial**

25. ABsmartly
26. Dapr
27. Grafana Alloy
28. Grafana Loki
29. Grafana Tempo
30. Railway
31. Unblocked
32. Weights & Biases

**Assess**

33. Arize Phoenix
34. Chainloop
35. Deepseek R1
36. Deno
37. Graphiti
38. Helicone
39. Humanloop
40. Model Context Protocol (MCP)
41. Open WebUI
42. pg_mooncake
43. Reasoning models
44. Restate
45. Supabase
46. Synthesized
47. Tonic.ai
48. turbopuffer
49. VectorChord

**Hold**

50. Tyk hybrid API management

## Tools

**Adopt**

51. Renovate
52. uv
53. Vite

**Trial**

54. Claude Sonnet
55. Cline
56. Cursor
57. D2
58. Databricks Delta Live Tables
59. JSON Crack
60. MailSlurp
61. Metabase
62. NeMo Guardrails
63. Nyx
64. OpenRewrite
65. Plerion
66. Software engineering agents
67. Tuple
68. Turborepo

**Assess**

69. AnythingLLM
70. Gemma Scope
71. Hurl
72. Jujutsu
73. kubenetmon
74. Mergiraf
75. ModernBERT
76. OpenRouter
77. Redactive
78. System Initiative
79. TabPFN
80. v0
81. Windsurf
82. YOLO

**Hold**

—

## Languages and Frameworks

**Adopt**

83. OpenTelemetry
84. React Hook Form

**Trial**

85. Effect
86. Hasura GraphQL engine
87. LangGraph
88. MarkItDown
89. Module Federation
90. Prisma ORM

**Assess**

91. .NET Aspire
92. Android XR SDK
93. Browser Use
94. CrewAI
95. ElysiaJs
96. FastGraphRAG
97. Gleam
98. GoFr
99. Java post-quantum cryptography
100. Presidio
101. PydanticAI
102. Swift for resource-constrained applications
103. Tamagui
104. torchtune

**Hold**

105. Node overload

# Techniques

**Adopt**

1. Data product thinking
2. Fuzz testing
3. Software Bill of Materials
4. Threat modeling

**Trial**

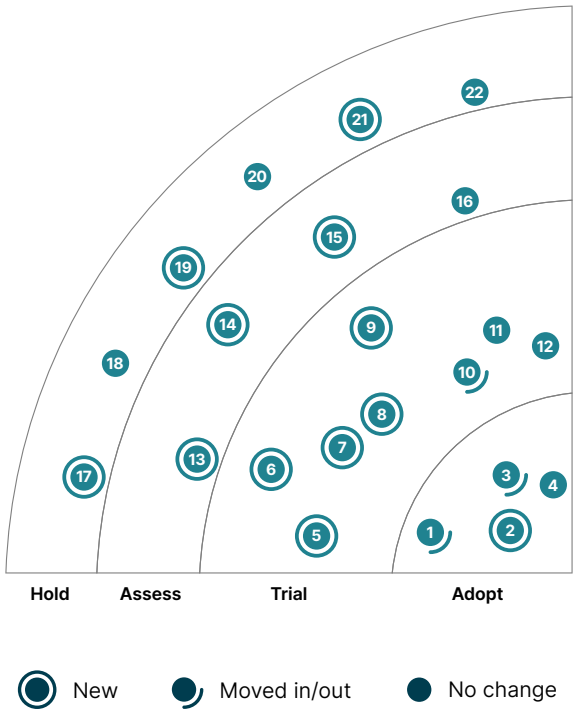
5. API request collection as API product artifact
6. Architecture advice process
7. GraphRAG
8. Just-in-time privileged access management
9. Model distillation
10. Prompt engineering
11. Small language models
12. Using GenAI to understand legacy codebases

**Assess**

13. AI-friendly code design
14. AI-powered UI testing
15. Competence envelope as a model for understanding system failures
16. Structured output from LLMs

**Hold**

17. AI-accelerated shadow IT
18. Complacency with AI-generated code
19. Local coding assistants
20. Replacing pair programming with AI
21. Reverse ETL
22. SAFe™



## 1. Data product thinking

***Adopt***

Organizations actively adopt data product thinking as a standard practice for managing data assets. This approach treats data as a product with its own lifecycle, quality standards and focus on meeting consumer needs. We now recommend it as default advice for data management, regardless of whether organizations choose architectures like data mesh or lakehouse.

We emphasize consumer-centricity in data product thinking to drive greater adoption and value realization. This means designing data products by working backward from use cases. We also focus on capturing and managing both business-relevant metadata and technical metadata using modern data catalogs like DataHub, Collibra, Atlan and Informatica. These practices improve data discoverability and usability. Additionally, we apply data product thinking to scale AI initiatives and create AI-ready data. This approach includes comprehensive lifecycle management, ensuring data is not only well-governed and high quality but also retired in compliance with legal and regulatory requirements when no longer needed.

## 2. Fuzz testing

***Adopt***

Fuzz testing , or simply fuzzing, is a testing technique that has been around for a long time but it is still one of the lesser-known techniques. The goal is to feed a software system all kinds of invalid input and observe its behavior. For an HTTP endpoint, for example, bad requests should result in 4*xx* errors, but fuzz testing often provokes 5*xx* errors or worse. Well-documented and supported by tools, fuzz testing is more relevant than ever with more AI-generated code and complacency with AI generated code. That means it's now a good time to adopt fuzz testing to ensure code remains robust and secure.

## 3. Software Bill of Materials

***Adopt***

Since our initial blip in 2021, Software Bill of Materials (SBOM) generation has transitioned from an emerging practice to a sensible default across our projects. The ecosystem has matured significantly, offering a robust tooling ecosystem and seamless CI/CD integration. Tools like Syft, Trivy and Snyk provide comprehensive SBOM generation from source code to container images and vulnerability scanning. Platforms such as FOSSA and Chainloop enhance security risk management by integrating with development workflows and enforcing security policies. While a universal SBOM standard is still evolving, broad support for SPDX and CycloneDX has minimized adoption challenges. AI systems also require SBOMs, as evidenced by the UK Government's AI Cyber Security Code of Practice and CISA's AI Cybersecurity Collaboration Playbook. We'll continue to monitor developments in this space.

## 4. Threat modeling

***Adopt***

In the rapidly evolving AI-driven landscape of software development, threat modeling is more crucial than ever for building secure software while maintaining agility and avoiding the security sandwich. Threat modeling — a set of techniques for identifying and classifying potential threats — applies across various contexts, including generative AI applications, which introduce unique security risks. To be effective, it must be performed regularly throughout the software lifecycle and works best alongside other security practices. These include defining cross-functional security requirements to address common risks in the project's technologies and leveraging automated security scanners for continuous monitoring.

## 5. API request collection as API product artifact

*Trial*

Treating APIs as a product means prioritizing developer experience, not only by putting sensible and standard design into APIs themselves but also providing comprehensive documentation as well as smooth onboarding experiences. While OpenAPI (Swagger) specifications can effectively document API interfaces, onboarding remains a challenge. Developers need rapid access to working examples, with preconfigured authentication and realistic test data. With the maturing of API client tools (such as Postman, Bruno and Insomnia), we recommend treating API request collection as API product artifact. API request collections should be thoughtfully designed to guide developers through key workflows, helping them to understand the API's domain language and functionality with minimal effort. To keep collections up to date, we recommend storing them in a repository and integrating them into the API's release pipeline.

## 6. Architecture advice process

*Trial*

One of the persistent challenges in large software teams is determining who makes the architectural decisions that shape the evolution of systems. The State of DevOps report reveals that the traditional approach of Architecture Review Boards is counterproductive, often hindering workflow and correlating with low organizational performance. A compelling alternative is an architectural advice process — a decentralized approach where anyone can make any architectural decision, provided they seek advice from those affected and those with relevant expertise. This method enables teams to optimize for flow without compromising architectural quality, both at small and large scales. At first glance, this approach may seem controversial, but practices such as Architecture Decision Records and advisory forums ensure that decisions remain informed, while empowering those closest to the work to make decisions. We've seen this model succeed at scale in an increasing number of organizations, including those in highly regulated industries.

## 7. GraphRAG

*Trial*

In our last update of RAG, we introduced GraphRAG, originally described in Microsoft's article as a two-step approach: (1) Chunking documents and using an LLM-based analysis of the chunks to create a knowledge graph; (2) retrieving relevant chunks at query time via embeddings while following edges in the knowledge graph to discover additional related chunks, which are then added to the augmented prompt. In many cases this approach enhances LLM-generated responses. We've observed similar benefits when using GenAI to understand legacy codebases, where we use structural information — such as abstract syntax trees and dependencies — to build the knowledge graph. The GraphRAG pattern has gained traction, with tools and frameworks like Neo4j's GraphRAG Python package emerging to support it. We also see Graphiti as fitting a broader interpretation of GraphRAG as a pattern.

## 8. Just-in-time privileged access management

*Trial*

Least privilege ensures users and systems have only the minimum access required to perform their tasks. Privileged credential abuse is a major factor in security breaches, with privilege escalation being a common attack vector. Attackers often start with low-level access and exploit software vulnerabilities or misconfigurations to gain administrator privileges, especially when accounts have excessive or unnecessary rights. Another overlooked risk is standing privileges — continuously available privileged access that expands the attack surface. Just-in-time privileged

access management (JIT PAM) mitigates this by granting access only when needed and revoking it immediately after, minimizing exposure. A true least-privilege security model ensures that users, applications and systems have only the necessary rights for the shortest required duration — a critical requirement for compliance and regulatory security. Our teams have implemented this through an automated workflow that triggers a lightweight approval process, assigns temporary roles with restricted access and enforces time to live (TTL) for each role, ensuring privileges expire automatically once the task is completed.

## 9. Model distillation

*Trial*

Scaling laws have been a key driver of the AI boom — the principle that larger models, datasets and compute resources lead to more powerful AI systems. However, consumer hardware and edge devices often lack the capacity to support large-scale models, creating the need for model distillation.

Model distillation transfers knowledge from a larger, more powerful model (teacher) to a smaller, cost-efficient model (student). The process typically involves generating a sample dataset from the teacher model and fine-tuning the student to capture its statistical properties. Unlike pruning or quantization, which focus on compressing models by removing parameters, distillation aims to retain domain-specific knowledge, minimizing accuracy loss. It can also be combined with quantization for further optimization.

Originally proposed by Geoffrey Hinton et al., model distillation has gained widespread adoption. A notable example is the Qwen/Llama distilled version of DeepSeek R1, which preserves strong reasoning capabilities in smaller models. With its growing maturity, the technique is no longer confined to research labs; it's now being applied to everything from industrial to personal projects. Providers such as OpenAI and Amazon Bedrock offer guides to help developers distill their own small language models (SLMs). We believe adopting model distillation can help organizations manage LLM deployment costs while unlocking the potential of on-device LLM inference.

## 10. Prompt engineering

*Trial*

Prompt engineering refers to the process of designing and refining prompts for generative AI models to produce high-quality, context-aware responses. This involves crafting clear, specific and relevant prompts tailored to the task or application to optimize the model's output. As LLM capabilities evolve, particularly with the emergence of reasoning models, prompt engineering practices must also adapt. Based on our experience with AI code generation, we've observed that few-shot prompting may underperform compared to simple zero-shot prompting when working with reasoning models. Additionally, the widely used chain-of-thought (CoT) prompting can degrade reasoning model performance — likely because reinforcement learning has already fine-tuned their built-in CoT mechanism.

Our hands-on experience aligns with academic research, which suggests "advanced models may eliminate the need for prompt engineering in software engineering." However, traditional prompt engineering techniques still play a crucial role in reducing hallucinations and improving output quality, especially given the differences in response time and token costs between reasoning models and general LLMs. When building agentic applications, we recommend choosing models strategically, based on your needs, while continuing to refine your prompt templates and corresponding techniques. Striking the right balance among performance, response time and token cost remains key to maximizing LLM effectiveness.

## 11. Small language models

*Trial*

The recent announcement of DeepSeek R1 is a great example of why small language models (SLMs) continue to be interesting. The full-size R1 has 671 billion parameters and requires about 1,342 GB of VRAM in order to run, only achievable using a "mini cluster" of eight state-of-the-art NVIDIA GPUs. But DeepSeek is also available "distilled" into Qwen and Llama — smaller, open-weight models — effectively transferring its abilities and allowing it to be run on much more modest hardware. Though the model gives up some performance at those smaller sizes, it still allows a huge leap in performance over previous SLMs. The SLM space continues to innovate elsewhere, too. Since the last Radar, Meta introduced Llama 3.2 at 1B and 3B sizes, Microsoft released Phi-4, offering high-quality results with a 14B model, and Google released PaliGemma 2, a vision-language model at 3B, 10B and 28B sizes. These are just a few of the models currently being released at smaller sizes and definitely an important trend to continue to watch.

## 12. Using GenAI to understand legacy codebases

*Trial*

In the past few months, using GenAI to understand legacy codebases has made some real progress. Mainstream tools such as GitHub Copilot are being touted as being able to help modernize legacy codebases. Tools such as Sourcegraph's Cody are making it easier for developers to navigate and understand entire codebases. These tools use a multitude of GenAI techniques to provide contextual help, simplifying work with complex legacy systems. On top of that, specialized frameworks like S3LLM are showing how LLMs can handle large-scale scientific software — such as that written in Fortran or Pascal — bringing GenAI-enhanced understanding to codebases outside of traditional enterprise IT. We think this technique is going to continue to gain traction given the sheer amount of legacy software in the world.

## 13. AI-friendly code design

*Assess*

Supervised software engineering agents are increasingly capable of identifying necessary updates and making larger changes to a codebase. At the same time, we're seeing growing complacency with AI-generated code, and developers becoming reluctant to review large AI-made change sets. A common justification for this is that human-oriented code quality matters less since AI can handle future modifications; however, AI coding assistants also perform better with well-factored codebases, making AI-friendly code design crucial for maintainability.

Fortunately, good software design for humans also benefits AI. Expressive naming provides domain context and functionality; modularity and abstractions keep AI's context manageable by limiting necessary changes; and the DRY (don't repeat yourself) principle reduces duplicate code — making it easier for AI to keep the behavior consistent. So far, the best AI-friendly patterns align with established best practices. As AI evolves, expect more AI-specific patterns to emerge, so thinking about code design with this in mind will be extremely helpful.

## 14. AI-powered UI testing

*Assess*

New techniques for AI-powered assistance on software teams are emerging beyond just code generation. One area gaining traction is AI-powered UI testing, leveraging LLMs' abilities to interpret graphical user interfaces. There are several approaches to this. One category of tools uses multi-

modal LLMs fine-tuned for UI snapshot processing, allowing test scripts written in natural language to navigate an application. Examples in this space include QA.tech or LambdaTests' KaneAI. Another approach, seen in Browser Use, combines multi-modal foundation models with Playwright's insights into a web page's structure rather than relying on fine-tuned models.

When integrating AI-powered UI tests into a test strategy, it's crucial to consider where they provide the most value. These methods can complement manual exploratory testing, and while the non-determinism of LLMs may introduce flakiness, their fuzziness can be an advantage. This could be useful for testing legacy applications with missing selectors or applications that frequently change labels and click paths.

## 15. Competence envelope as a model for understanding system failures

***Assess***

The theory of graceful extensibility defines the basic rules governing adaptive systems, including the socio-technical systems involved in building and operating software. A key concept in this theory is the competence envelope — the boundary within which a system can function robustly in the face of failure. When a system is pushed beyond its competence envelope, it becomes brittle and is more likely to fail. This model provides a valuable lens for understanding system failure, as seen in the complex failures that led to the 2024 Canva outage. Residuality theory, a recent development in software architecture thinking, offers a way to test a system's competence envelope by deliberately introducing stressors and analyzing how the system has adapted to historical stressors over time. The approaches align with concepts of anti-fragility, resilience and robustness in socio-technical systems, and we're eager to see practical applications of these ideas emerge in the field.

## 16. Structured output from LLMs

***Assess***

Structured output from LLMs refers to the practice of constraining a language model's response into a defined schema. This can be achieved either by instructing a generalized model to respond in a particular format or by fine-tuning a model so it "natively" outputs, for example, JSON. OpenAI now supports structured output, allowing developers to supply a JSON Schema, pydantic or Zod object to constrain model responses. This capability is particularly valuable for enabling function calling, API interactions and external integrations, where accuracy and adherence to a format are critical. Structured output not only enhances the way LLMs can interface with code but also supports broader use cases like generating markup for rendering charts. Additionally, structured output has been shown to reduce the chance of hallucinations in model outputs.

## 17. 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.

## 18. Complacency with AI-generated code

*Hold*

As AI coding assistants continue to gain traction, so does the growing body of data and research highlighting concerns about complacency with AI-generated code. GitClear's latest code quality research shows that in 2024, duplicate code and code churn have increased even more than predicted, while refactoring activity in commit histories has declined. Also reflecting AI complacency, Microsoft research on knowledge workers found 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 supervised software engineering agents further amplifies the risks, because when AI generates larger and larger change sets, developers face greater challenges in reviewing results. The emergence of "vibe coding" — where developers let AI generate code with minimal review — illustrates the growing trust of AI-generated outputs. While this approach can be appropriate for things like prototypes or other types of throw-away code, we strongly caution against using it for production code.

## 19. Local coding assistants

*Hold*

Organizations remain wary of third-party AI coding assistants, particularly due to concerns about code confidentiality. As a result, many developers are considering using local coding assistants — AI that runs entirely on their machines — eliminating the need to send code to external servers. However, local assistants still lag behind their cloud-based counterparts, which rely on larger, more capable models. Even on high-end developer machines, smaller models remain limited in their capabilities. We've found that they struggle with complex prompts, lack the necessary context window for larger problems and often cannot trigger tool integrations or function calls. These capabilities are especially essential to agentic workflows, which is the cutting edge in coding assistance right now.

So while we recommend to proceed with low expectations, there are some capabilities that are valid locally. Some popular IDEs do now embed smaller models into their core features, such as Xcode's predictive code completion and JetBrains' full-line code completion. And locally runnable LLMs like Qwen Coder are a step forward for local inline suggestions and handling simple coding queries. You can test these capabilities with Continue, which supports the integration of local models via runtimes like Ollama.

## 20. Replacing pair programming with AI

*Hold*

When people talk about coding assistants, the topic of pair programming inevitably comes up. Our profession has a love-hate relationship with it: some swear by it, others can't stand it. Coding assistants now raise the question: can a human pair with the AI instead of another human and get the same results for the team? GitHub Copilot even calls itself "your AI pair programmer." While we do think a coding assistant can bring some of the benefits of pair programming, we advise against fully replacing pair programming with AI. Framing coding assistants as pair programmers ignores one of the

key benefits of pairing: to make the team, not just the individual contributors, better. Coding assistants can offer benefits for getting unstuck, learning about a new technology, onboarding or making tactical work faster so that we can focus on the strategic design. But they don't help with any of the team collaboration benefits, like keeping the work-in-progress low, reducing handoffs and relearning, making continuous integration possible or improving collective code ownership.

## 21. Reverse ETL

***Hold***

We're seeing a worrying proliferation of so-called Reverse ETL. Regular ETL jobs have their place in traditional data architectures, where they transfer data from transaction processing systems to a centralized analytics system, such as a data warehouse or data lake. While this architecture has well-documented shortcomings, many of which are addressed by a data mesh, it remains common in enterprises. In such an architecture, moving data back from a central analytics system to a transaction system makes sense in certain cases — for example, when the central system can aggregate data from multiple sources or as part of a transitional architecture when migrating toward a data mesh. However, we're seeing a growing trend where product vendors use Reverse ETL as an excuse to move increasing amounts of business logic into a centralized platform — their product. This approach exacerbates many of the issues caused by centralized data architectures, and we suggest exercising extreme caution when introducing data flows from a sprawling, central data platform to transaction processing systems.

## 22. SAFe™

***Hold***

We see continued adoption of SAFe™ (Scaled Agile Framework®). We also continue to observe that SAFe's over-standardized, phase-gated processes create friction, that it can promote silos and that its top-down control generates waste in the value stream and discourages engineering talent creativity, while limiting autonomy and experimentation in teams. A key reason for adoption is the complexity of making an organization agile, with enterprises hoping that a framework like SAFe offers a simple, process-based shortcut to becoming agile. Given the widespread adoption of SAFe — including among our clients — we've trained over 100 Thoughtworks consultants to better support them. Despite this in-depth knowledge and no lack of trying we come away thinking that sometimes there just is no simple solution to a complex problem, and we keep recommending leaner, value-driven approaches and governance that work in conjunction with a comprehensive change program.

Scaled Agile Framework® and SAFe™ are trademarks of Scaled Agile, Inc.

# Platforms

**Adopt**

23. GitLab CI/CD
24. Trino

**Trial**

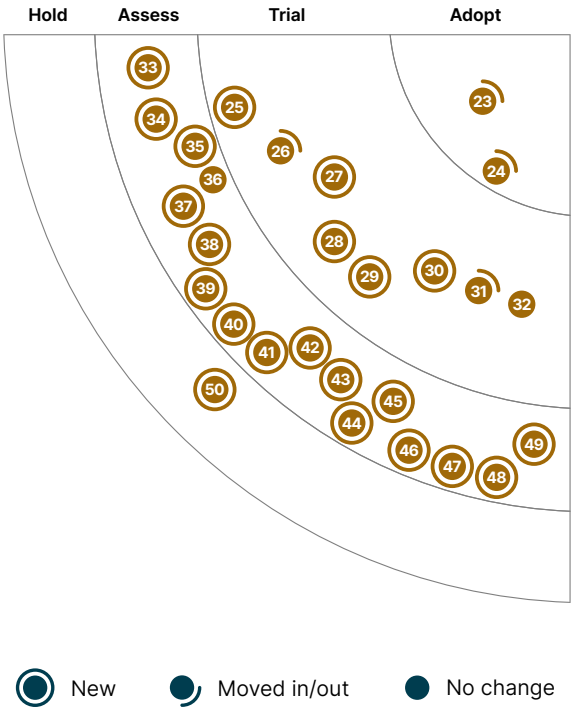
25. ABsmartly
26. Dapr
27. Grafana Alloy
28. Grafana Loki
29. Grafana Tempo
30. Railway
31. Unblocked
32. Weights & Biases

**Assess**

33. Arize Phoenix
34. Chainloop
35. Deepseek R1
36. Deno
37. Graphiti
38. Helicone
39. Humanloop
40. Model Context Protocol (MCP)
41. Open WebUI
42. pg_mooncake
43. Reasoning models
44. Restate
45. Supabase
46. Synthesized
47. Tonic.ai
48. turbopuffer
49. VectorChord

**Hold**

50. Tyk hybrid API management



## 23. GitLab CI/CD

*Adopt*

GitLab CI/CD has evolved into a fully integrated system within GitLab, covering everything from code integration and testing to deployment and monitoring. It supports complex workflows with features like multi-stage pipelines, caching, parallel execution and auto-scaling runners and is suitable for large-scale projects and complex pipeline needs. We want to highlight its built-in security and compliance tools (such as SAST and DAST analysis) which make it well-suited for use cases with high-compliance requirements. It also integrates seamlessly with Kubernetes, supporting cloud-native workflows, and offers real-time logging, test reports and traceability for enhanced observability.

## 24. Trino

*Adopt*

Trino is an open-source, distributed SQL query engine designed for interactive analytic queries over big data. It's optimized to run both on-premise and cloud environments and supports querying data where it resides, including relational databases and various proprietary datastores via connectors. Trino can also query data stored in file formats like Parquet and open-table formats like Apache Iceberg. Its built-in query federation capabilities enable data from multiple sources to be queried as a single logical table, making it a great choice for analytic workloads that require aggregating data across diverse sources. Trino is a key part of popular stacks like AWS Athena, Starburst and other proprietary data platforms. Our teams have successfully used it in various use cases, and when it comes to querying data sets across multiple sources for analytics, Trino has been a reliable choice.

## 25. ABsmartly

*Trial*

ABsmartly is an advanced A/B testing and experimentation platform designed for rapid, trustworthy decision-making. Its standout feature is the Group Sequential Testing (GST) engine, which accelerates test results by up to 80% compared to traditional A/B testing tools. The platform offers real-time reporting, deep data segmentation and seamless full-stack integration through an API-first approach, supporting experiments across web, mobile, microservices and ML models.

ABsmartly addresses key challenges in scalable, data-driven experimentation by enabling faster iteration and more agile product development. Its zero-lag execution, deep segmentation capabilities and support for multi-platform experiments make it particularly valuable for organizations looking to scale their experimentation culture and prioritize data-backed innovation. By significantly reducing test cycles and automating result analysis, ABsmartly helped us optimize features and user experiences more efficiently than traditional A/B testing platforms.

## 26. Dapr

*Trial*

Dapr has evolved considerably since we last featured it in the Radar. Its many new features include job scheduling, virtual actors as well as more sophisticated retry policies and observability components. Its list of building blocks continues to grow with jobs, cryptography and more. Our teams also note its increasing focus on secure defaults, with support for mTLS and distroless images. All in all, we've been happy with Dapr and are looking forward to future developments.

## 27. Grafana Alloy

*Trial*

Formerly known as Grafana Agent, Grafana Alloy is an open-source OpenTelemetry Collector. Alloy is designed to be an all-in-one telemetry collector for all telemetry data, including logs, metrics and traces. It supports collecting commonly used telemetry data formats such as OpenTelemetry, Prometheus and Datadog. With Promtail's recent deprecation, Alloy is emerging as a go-to choice for telemetry data collection — especially for logs — if you're using the Grafana observability stack.

## 28. Grafana Loki

*Trial*

Grafana Loki is a horizontally scalable and highly available multi-tenant log aggregation system inspired by Prometheus. Loki only indexes metadata about your logs as a set of labels for each log stream. Log data is stored in a block storage solution such as S3, GCS or Azure Blob Storage. The upshot is that Loki promises a reduction in operational complexity and storage costs over competitors. As you'd expect, it integrates tightly with Grafana and Grafana Alloy, although other collection mechanisms can be used.

Loki 3.0 introduced native OpenTelemetry support, making ingestion and integration with OpenTelemetry systems as simple as configuring an endpoint. It also offers advanced multi-tenancy features, such as tenant isolation via shuffle-sharding, which prevents misbehaving tenants (e.g., heavy queries or outages) from impacting others in a cluster. If you haven't been following developments in the Grafana ecosystem, now is a great time to take a look as it is evolving rapidly.

## 29. Grafana Tempo

*Trial*

Grafana Tempo is a high-scale distributed tracing backend that supports open standards like OpenTelemetry. Designed to be cost-efficient, it relies on object storage for long-term trace retention and enables trace search, span-based metric generation and correlation with logs and metrics. By default, Tempo uses a columnar block format based on Apache Parquet, enhancing query performance and enabling downstream tools to access trace data. Queries are executed via TraceQL and the Tempo CLI. Grafana Alloy too can be configured to collect and forward traces to Tempo. Our teams self-hosted Tempo in GKE, using MinIO for object storage, OpenTelemetry collectors and Grafana for trace visualization.

## 30. Railway

*Trial*

Heroku used to be an excellent choice for many developers who wanted to release and deploy their applications quickly. In recent years, we've also seen the rise of deployment platforms like Vercel, which are more modern, lightweight and easy to use but designed for front-end applications. A full-stack alternative in this space is Railway, a PaaS cloud platform that streamlines everything from GitHub/Docker deployment to production observability.

Railway supports most mainstream programming frameworks, databases as well as containerized deployment. As a long-term hosted platform for an application, you may need to compare the costs of different platforms carefully. At present, our team has had a good experience with Railway's deployment and observability. The operation is smooth and can be well integrated with the continuous deployment practices we advocate.

## 31. Unblocked

*Trial*

Unblocked is an off-the-shelf AI team assistant. Once integrated with codebase repositories, corporate documentation platforms, project management tools and communication tools, Unblocked helps answer questions about complex business and technical concepts, architectural design and implementation as well as operational processes. This is particularly useful for navigating large or legacy systems. While using Unblocked, we've observed that teams value quick access to contextual information over code and user-story generation. For scenarios requiring more extensive code generation or task automation, dedicated software engineering agents or coding assistants are more suitable.

## 32. Weights & Biases

*Trial*

Weights & Biases has continued to evolve, adding more LLM-focused features since it was last featured in the Radar. They are expanding Traces and introducing Weave, a full-fledged platform that goes beyond tracking LLM-based agentic systems. Weave enables you to create system evaluations, define custom metrics, use LLMs as judges for tasks like summarization and save data sets that capture different behaviors for analysis. This helps optimize LLM components and track performance at both local and global levels. The platform also facilitates iterative development and effective debugging of agentic systems, where errors can be difficult to detect. Additionally, it enables the collection of valuable human feedback, which can later be used for fine-tuning models.

## 33. Arize Phoenix

*Assess*

With the popularity of LLM and agentic applications, LLM observability is becoming more and more important. Previously, we've recommended platforms such as Langfuse and Weights & Biases (W&B). Arize Phoenix is another emerging platform in this space, and our team has had a positive experience using it. It offers standard features like LLM tracing, evaluation and prompt management, with seamless integration into leading LLM providers and frameworks. This makes it easy to gather insights on LLM output, latency and token usage with minimal configuration. So far, our experience is limited to the open-source tool but the broader Arize platform offers more comprehensive capabilities. We look forward to exploring it in the future.

## 34. Chainloop

*Assess*

Chainloop is an open-source supply chain security platform that helps security teams enforce compliance while allowing development teams to seamlessly integrate security compliance into CI/CD pipelines. It consists of a control plane, which acts as the single source of truth for security policies, and a CLI, which runs attestations within CI/CD workflows to ensure compliance. Security teams define workflow contracts specifying which artifacts — such as SBOMs and vulnerability reports — must be collected, where to store them and how to evaluate compliance. Chainloop uses Rego, OPA's policy language, to validate attestations — for example, ensuring a CycloneDX SBOM meets version requirements. During workflow execution, security artifacts like SBOMs are attached to an attestation and pushed to the control plane for enforcement and auditing. This approach ensures compliance can be enforced consistently and at scale while minimizing friction in development workflows. This results in an SLSA level-three–compliant single source of truth for metadata, artefacts and attestations.

## 35. Deepseek R1

*Assess*

DeepSeek-R1 is DeepSeek's first-generation of reasoning models. Through a progression of non-reasoning models, the engineers at DeepSeek designed and used methods to maximize hardware utilization. These include Multi-Head Latent Attention (MLA), Mixture of Experts (MoE) gating, 8-bit floating points training (FP8) and low-level PTX programming. Their high-performance computing co-design approach enables DeepSeek-R1 to rival state-of-the-art models at significantly reduced cost for training and inference.

DeepSeek-R1-Zero is notable for another innovation: the engineers were able to elicit reasoning capabilities from a non-reasoning model using simple reinforcement learning without any supervised fine-tuning. All DeepSeek models are open-weight, which means they are freely available, though training code and data remain proprietary. The repository includes six dense models distilled from DeepSeek-R1, based on Llama and Qwen, with DeepSeek-R1-Distill-Qwen-32B outperforming OpenAI-o1-mini on various benchmarks.

## 36. Deno

*Assess*

Created by Ryan Dahl, the inventor of Node.js, Deno was designed to address what he saw as mistakes in Node.js. It features a stricter sandboxing system, built-in dependency management and native TypeScript support — a key draw for its user base. Many of us prefer Deno for TypeScript projects, as it feels like a true TypeScript run time and toolchain, rather than an add-on to Node.js.

Since its inclusion in the Radar in 2019, Deno has made significant advancements. The Deno 2 release introduces backward compatibility with Node.js and npm libraries, long-term support (LTS) releases and other improvements. Previously, one of the biggest barriers to adoption was the need to rewrite Node.js applications. These updates reduce migration friction while expanding dependency options for supporting tools and systems. Given the massive Node.js and npm ecosystem, these changes should drive further adoption.

Additionally, Deno's Standard Library has stabilized, helping combat the proliferation of low-value npm packages across the ecosystem. Its tooling and Standard Library make TypeScript or JavaScript more appealing for server-side development. However, we caution against choosing a platform solely to avoid polyglot programming.

## 37. Graphiti

*Assess*

Graphiti builds dynamic, temporally-aware knowledge graphs that capture evolving facts and relationships. Our teams use GraphRAG to uncover data relationships, which enhances retrieval and response accuracy. As data sets constantly evolve, Graphiti maintains temporal metadata on graph edges to record relationship lifecycles. It ingests both structured and unstructured data as discrete episodes and supports queries using a fusion of time-based, full-text, semantic and graph algorithms. For LLM-based applications — whether RAG or agentic — Graphiti enables long-term recall and state-based reasoning.

## 38. Helicone

***Assess***

Similar to Langfuse, Weights & Biases and Arize Phoenix, Helicone is a managed LLMOps platform designed to meet the growing enterprise demand for LLM cost management, ROI evaluation and risk mitigation. Open-source and developer-focused, Helicone supports production-ready AI applications, offering prompt experimentation, monitoring, debugging and optimization across the entire LLM lifecycle. It enables real-time analysis of costs, utilization, performance and agentic stack traces across various LLM providers. While it simplifies LLM operations management, the platform is still emerging and may require some expertise to fully leverage its advanced features. Our team has been using it with good experience so far.

## 39. Humanloop

***Assess***

Humanloop is an emerging platform focused on making AI systems more reliable, adaptable and aligned with user needs by integrating human feedback at key decision points. It offers tools for human labeling, active learning and human-in-the-loop fine-tuning as well as LLM evaluation against business requirements. Additionally, it helps manage the cost-effective lifecycle of GenAI solutions with greater control and efficiency. Humanloop supports collaboration through a shared workspace, version-controlled prompt management and CI/CD integration to prevent regressions. It also provides observability features such as tracing, logging, alerting and guardrails to monitor and optimize AI performance. These capabilities make it particularly relevant for organizations deploying AI in regulated or high-risk domains where human oversight is critical. With its focus on responsible AI practices, Humanloop is worth evaluating for teams looking to build scalable and ethical AI systems.

## 40. Model Context Protocol (MCP)

***Assess***

One of the biggest challenges in prompting is ensuring the AI tool has access to all the context relevant to the task. Often, this context already exists within the systems we use all day: wikis, issue trackers, databases or observability systems. Seamless integration between AI tools and these information sources can significantly improve the quality of AI-generated outputs.

The Model Context Protocol (MCP), an open standard released by Anthropic, provides a standardized framework for integrating LLM applications with external data sources and tools. It defines MCP servers and clients, where servers access the data sources and clients integrate and use this data to enhance prompts. Many coding assistants have already implemented MCP integration, allowing them to act as MCP clients. MCP servers can be run in two ways: Locally, as a Python or Node process running on the user's machine, or remotely, as a server that the MCP client connects to via SSE (though we haven't seen any usage of the remote server variant yet). Currently, MCP is primarily used in the first way, with developers cloning open-source MCP server implementations. While locally run servers offer a neat way to avoid third-party dependencies, they remain less accessible to nontechnical users and introduce challenges such as governance and update management. That said, it's easy to imagine how this standard could evolve into a more mature and user-friendly ecosystem in the future.

## 41. Open WebUI

*Assess*

Open WebUI is an open-source, self-hosted AI platform with a versatile feature set. It supports OpenAI-compatible APIs and integrates with providers like OpenRouter and GroqCloud, among others. It can run entirely offline by connecting to local or self-hosted models via Ollama. Open WebUI includes a built-in capability for RAG, allowing users to interact with local and web-based documents in a chat-driven experience. It offers granular RBAC controls, enabling different models and platform capabilities for different user groups. The platform is extensible through Functions — Python-based building blocks that customize and enhance its capabilities. Another key feature is model evaluation, which includes a model arena for side-by-side comparisons of LLMs on specific tasks. Open WebUI can be deployed at various scales — as a personal AI assistant, a team collaboration assistant or an enterprise-grade AI platform.

## 42. pg_mooncake

*Assess*

pg_mooncake is a PostgreSQL extension that adds columnar storage and vectorized execution. Columnstore tables are stored as Iceberg or Delta Lake tables in the local file system or S3-compatible cloud storage. pg_mooncake supports loading data from file formats like Parquet, CSV and even Hugging Face datasets. It can be a good fit for heavy data analytics that typically requires columnar storage, as it removes the need to add dedicated columnar store technologies into your stack.

## 43. Reasoning models

*Assess*

One of the most significant AI advances since the last Radar is the breakthrough and proliferation of reasoning models. Also marketed as "thinking models," these models have achieved top human-level performance in benchmarks like frontier mathematics and coding.

Reasoning models are usually trained through reinforcement learning or supervised fine-tuning, enhancing capabilities such as step-by-step thinking (CoT), exploring alternatives (ToT) and self-correction. Examples include OpenAI's o1/o3, DeepSeek R1 and Gemini 2.0 Flash Thinking. However, these models should be seen as a distinct category of LLMs rather than simply more advanced versions.

This increased capability comes at a cost. Reasoning models require longer response time and higher token consumption, leading us to jokingly call them "Slower AI" (as if current AI wasn't slow enough). Not all tasks justify this trade-off. For simpler tasks like text summarization, content generation or fast-response chatbots, general-purpose LLMs remain the better choice. We advise using reasoning models in STEM fields, complex problem-solving and decision-making — for example, when using LLMs as judges or improving explainability through explicit CoT outputs. At the time of writing, Claude 3.7 Sonnet, a hybrid reasoning model, had just been released, hinting at a possible fusion between traditional LLMs and reasoning models.

## 44. Restate

*Assess*

Restate is a durable execution platform, similar to Temporal, developed by the original creators of Apache Flink. Feature-wise it offers workflows as code, stateful event processing, the saga pattern and durable state machines. Written in Rust and deployed as a single binary, it uses a distributed

log to record events, implemented using a virtual consensus algorithm based on Flexible Paxos; this ensures durability in the event of node failure. SDKs are available for the usual suspects: Java, Go, Rust and TypeScript. We still maintain that it's best to avoid distributed transactions in distributed systems, because of both the additional complexity and the inevitable additional operational overhead involved. However, this platform is worth assessing if you can't avoid distributed transactions in your environment.

## 45. Supabase

***Assess***

Supabase is an open-source Firebase alternative for building scalable and secure backends. It offers a suite of integrated services, including a PostgreSQL database, authentication, instant APIs, Edge Functions, real-time subscriptions, storage and vector embeddings. Supabase aims to streamline back-end development, allowing developers to focus on building front-end experiences while leveraging the power and flexibility of open-source technologies. Unlike Firebase, Supabase is built on top of PostgreSQL. If you're working on prototyping or an MVP, Supabase is worth considering, as it will be easier to migrate to another SQL solution after the prototyping stage.

## 46. Synthesized

***Assess***

A common challenge in software development is generating test data for development and test environments. Ideally, test data should be as production-like as possible, while ensuring no personally identifiable or sensitive information is exposed. Though this may seem straightforward, test data generation is far from simple. That's why we're interested in Synthesized — a platform that can mask and subset existing production data or generate statistically relevant synthetic data. It integrates directly into build pipelines and offers privacy masking, providing per-attribute anonymization through irreversible data obfuscation techniques such as hashing, randomization and binning. Synthesized can also generate large volumes of synthetic data for performance testing. While it includes the obligatory GenAI features, its core functionality addresses a real and persistent challenge for development teams, making it worth exploring.

## 47. Tonic.ai

***Assess***

Tonic.ai is part of a growing trend in platforms designed to generate realistic, de-identified synthetic data for development, testing and QA environments. Similar to Synthesized, Tonic.ai is a platform with a comprehensive suite of tools addressing various data synthesis needs in contrast to the library-focused approach of Synthetic Data Vault. Tonic.ai generates both structured and unstructured data, maintaining the statistical properties of production data while ensuring privacy and compliance through differential privacy techniques. Key features include automatic detection, classification and redaction of sensitive information in unstructured data, along with on-demand database provisioning via Tonic Ephemeral. It also offers Tonic Textual, a secure data lakehouse that helps AI developers leverage unstructured data for retrieval-augmented generation (RAG) systems and LLM fine-tuning. Teams looking to accelerate engineering velocity while generating scalable, realistic data — all while adhering to stringent data privacy requirements — should consider evaluating Tonic.ai.

## 48. turbopuffer

*Assess*

turbopuffer is a serverless, multi-tenant search engine that seamlessly integrates vector and full-text search on object storage. We quite like its architecture and design choices, particularly its focus on durability, scalability and cost efficiency. By using object storage as a write-ahead log while keeping its query nodes stateless, it's well-suited for high-scale search workloads.

Designed for performance and accuracy, turbopuffer delivers high recall out of the box, even for complex filter-based queries. It caches cold query results on NVMe SSDs and keeps frequently accessed namespaces in memory, enabling low-latency search across billions of documents. This makes it ideal for large-scale document retrieval, vector search and retrieval-augmented generation (RAG) AI applications. However, its reliance on object storage introduces trade-offs in query latency, making it most effective for workloads that benefit from stateless, distributed compute. turbopuffer powers high-scale production systems like Cursor but is currently only available by referral or invitation.

## 49. VectorChord

*Assess*

VectorChord is a PostgreSQL extension for vector similarity search, developed by the creators of pgvecto.rs as its successor. It's open source, compatible with pgvector data types and designed for disk-efficient, high-performance vector search. It employs inverted file indexing (IVF) along with RaBitQ quantization to enable fast, scalable and accurate vector search while significantly reducing computation demands. Like other PostgresSQL extensions in this space, it leverages the PostgreSQL ecosystem, allowing vector search alongside standard transactional operations. Though still in its early stages, VectorChord is worth assessing for vector search workloads.

## 50. Tyk hybrid API management

*Hold*

We've observed multiple teams encountering issues with the Tyk hybrid API management solution. While the concept of a managed control plane and self-managed data planes offers flexibility for complex infrastructure setups (such as multi-cloud and hybrid cloud), teams have experienced control plane incidents that were only discovered internally rather than by Tyk, highlighting potential observability gaps in Tyk's AWS-hosted environment. Furthermore, the level of incident support appears slow; communicating via tickets and emails isn't ideal in these situations. Teams have also reported issues with the maturity of Tyk's documentation, often finding it inadequate for complex scenarios and issues. Additionally, other products in the Tyk ecosystem seem immature as well, for example, the enterprise developer portal is reported to not be backward compatible and has limited customization capabilities. Especially for Tyk's hybrid setup, we recommend proceeding with caution and will continue to monitor its maturity.

# Tools

**Adopt**

51. Renovate
52. uv
53. Vite

**Trial**

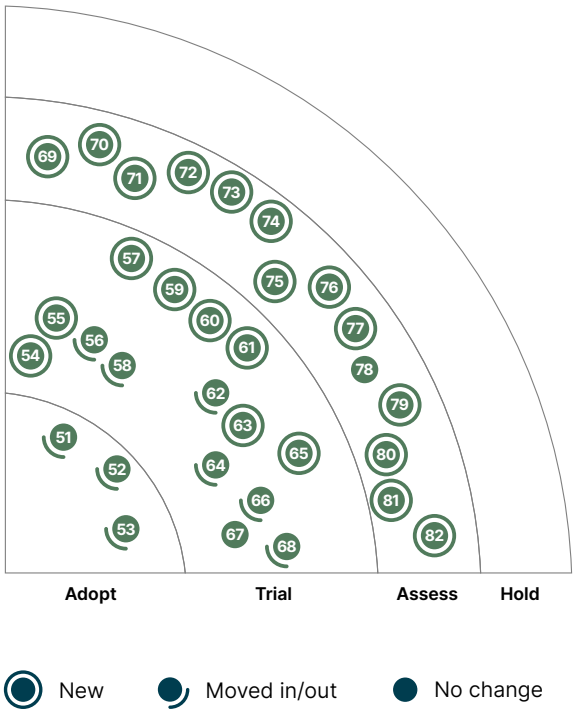
54. Claude Sonnet
55. Cline
56. Cursor
57. D2
58. Databricks Delta Live Tables
59. JSON Crack
60. MailSlurp
61. Metabase
62. NeMo Guardrails
63. Nyx
64. OpenRewrite
65. Plerion
66. Software engineering agents
67. Tuple
68. Turborepo

**Assess**

69. AnythingLLM
70. Gemma Scope
71. Hurl
72. Jujutsu
73. kubenetmon
74. Mergiraf
75. ModernBERT
76. OpenRouter
77. Redactive
78. System Initiative
79. TabPFN
80. v0
81. Windsurf
82. YOLO

**Hold**

—



## 51. Renovate

*Adopt*

Renovate has become the tool of choice for many of our teams looking to take a proactive approach to dependency version management. While Dependabot remains a safe default choice for GitHub-hosted repositories, we continue to recommend evaluating Renovate as a more comprehensive and customizable solution. To maximize Renovate's benefits, configure it to monitor and update all dependencies, including tooling, infrastructure and private or internally hosted dependencies. To reduce developer fatigue, consider automatic merging of dependency update PRs.

## 52. uv

*Adopt*

Since the last Radar, we've gained more experience with uv, and feedback from teams has been overwhelmingly positive. uv is a next-generation Python package and project management tool written in Rust, with a key value proposition: it's "extremely fast." It outperforms other Python package managers by a large margin in benchmarks, accelerating build and test cycles and significantly improving developer experience. Beyond performance, uv offers a unified toolset, effectively replacing tools like Poetry, pyenv and pipx. However, our concerns around package management tools remain: a strong ecosystem, mature community and long-term support are critical. Given that uv is relatively new, moving it to the Adopt ring is bold. That said, many data teams are eager to move away from Python's legacy package management system, and our frontline developers consistently recommend uv as the best tool available today.

## 53. Vite

*Adopt*

Since Vite was last mentioned in the Radar, it has gained even more traction. It's a high-performance front-end build tool with fast hot-reloading. It's being adopted and recommended as a default choice in many front-end frameworks, including Vue, SvelteKit and React which recently deprecated create-react-app. Vite also recently received significant investment, which led to the founding of VoidZero, an organization dedicated to Vite's development. This investment should accelerate development and enhance the project's long-term sustainability.

## 54. Claude Sonnet

*Trial*

Claude Sonnet is an advanced language model that excels in coding, writing, analysis and visual processing. It's available in the browser, terminal, most major IDEs and even integrates with GitHub Copilot. As of writing, benchmarking shows it outperforms previous models with versions 3.5 and 3.7, including earlier Claude models. It's also adept at interpreting charts and extracting text from images, and it features a developer-focused experience, such as with the "Artifacts" feature in the browser UI for generating and interacting with dynamic content such as code snippets and HTML designs.

We've used version 3.5 of Claude Sonnet extensively in software development and found it significantly boosts productivity across various projects. It excels in greenfield projects, particularly in collaborative software design and architectural discussions. While it may be too early to call any AI model "stable" for coding assistance, Claude Sonnet is among the most reliable models we've worked with. At the time of writing, Claude 3.7 has also been released and is promising, though we've not yet fully tested it in production.

## 55. Cline

*Trial*

Cline is an open-source VSCode extension that is currently one of the strongest contenders in the space of supervised software engineering agents. It lets developers drive their implementation entirely from the Cline chat, integrating seamlessly with the IDE they already use. Key features like Plan & Act mode, transparent token usage and MCP integration help developers interact effectively with LLMs. Cline has demonstrated advanced capabilities in handling complex development tasks, especially with Claude 3.5 Sonnet. It supports large codebases, automates headless browser testing and proactively fixes bugs. Unlike cloud-based solutions, Cline enhances privacy by storing data locally. Its open-source nature not only ensures greater transparency but also enables community-driven improvements. However, developers should be mindful of token usage cost, as Cline's code context orchestration, while very effective, is resource-intensive. Another potential bottleneck is rate limiting, which can slow down workflows. Until this is resolved, using API providers like OpenRouter, which provide better rate limits, is advisable.

## 56. Cursor

*Trial*

We continue to be impressed by the AI-first code editor Cursor, which remains a leader in the competitive AI coding assistance space. Its code context orchestration is very effective, and it supports a wide range of models, including the option to use a custom API key. The Cursor team often comes up with innovative user experience features before the other vendors, and they include an extensive list of context providers in their chat, such as the referencing of git diffs, previous AI conversations, web search, library documentation and MCP integration. Alongside tools like Cline and Windsurf, Cursor also stands out for its strong agentic coding mode. This mode allows developers to guide their implementation directly from an AI chat interface, with the tool autonomously reading and modifying files, as well as executing commands. Additionally, we appreciate Cursor's ability to detect linting and compilation errors in generated code and proactively correct them.

## 57. D2

*Trial*

D2 is an open-source diagrams-as-code tool that helps users create and customize diagrams from text. It introduces the D2 diagram scripting language, which prioritizes readability over compactness with a simple, declarative syntax. D2 ships with a default theme and leverages the same layout engine as Mermaid. Our teams appreciate its lightweight syntax, which is specifically designed for software documentation and architecture diagrams.

## 58. Databricks Delta Live Tables

*Trial*

Delta Live Tables (DLT) continues to prove its value in simplifying and streamlining data pipeline management, supporting both real-time streaming and batch processing through a declarative approach. By automating complex data engineering tasks, such as manual checkpoint management, DLT reduces operational overhead while ensuring a robust end-to-end system. Its ability to orchestrate simple pipelines with minimal manual intervention enhances reliability and flexibility, while features like materialized views provide incremental updates and performance optimization for specific use cases.

However, teams must understand DLT's nuances to fully leverage its benefits and avoid potential pitfalls. As an opinionated abstraction, DLT manages its own tables and restricts data insertion to a single pipeline at a time. Streaming tables are append-only, requiring careful design considerations. Additionally, deleting a DLT pipeline also deletes the underlying table and data, potentially creating operational issues.

## 59. JSON Crack

*Trial*

JSON Crack is a Visual Studio Code extension that renders interactive graphs from textual data. Despite its name it supports multiple formats, including YAML, TOML and XML. Unlike Mermaid and D2, where the textual form is a means to create a specific visual graph, JSON Crack is a tool to look at data that happens to be in a textual format. The layout algorithm works well and the tool allows selective hiding of branches and nodes, making it a great choice for exploring data sets. A companion web-based tool is also available, but our reservations about relying on online services for formatting or parsing code apply. JSON Crack does have a node limit, and directs users to a commercial sibling tool for handling files with more than a few hundred nodes.

## 60. MailSlurp

*Trial*

Testing workflows that involve email are often complex and time-consuming. Development teams must build custom email API clients for automation while also setting up temporary inboxes for manual testing scenarios, such as user testing or internal product training before major releases. These challenges become even more pronounced when developing customer onboarding products. We've had a positive experience with MailSlurp, a mail server and SMS API service. It provides REST APIs for creating inboxes and phone numbers as well as validating emails and messages directly in code, and its no-code dashboard is also useful for manual testing preparations. Additional features like custom domains, webhooks, auto-reply and forwarding are worth checking out for more complex scenarios.

## 61. Metabase

*Trial*

Metabase is an open-source analytics and business intelligence tool that allows users to visualize and analyze data from a variety of data sources, including relational and NoSQL databases. The tool helps users create visualizations and reports, organize them into dashboards and easily share insights. It also offers an SDK for embedding interactive dashboards in web applications, matching the theme and style of the application — making it developer-friendly. With both officially supported and community-backed data connectors, Metabase is versatile across data environments. As a lightweight BI tool, our teams find it useful for managing interactive dashboards and reports in their applications.

## 62. NeMo Guardrails

*Trial*

NeMo Guardrails is an easy-to-use open-source toolkit from NVIDIA that empowers developers to implement guardrails for LLMs used in conversational applications. Since we last mentioned it in the Radar, NeMo has seen significant adoption across our teams and continues to improve. Many of the latest enhancements to NeMo Guardrails focus on expanding integrations and strengthening security, data and control, aligning with the project's core goal.

A major update to NeMo's documentation has improved usability and new integrations have been added, including AutoAlign and Patronus Lynx, along with support for Colang 2.0. Key upgrades include enhancements to content safety and security as well as a recent release that supports streaming LLM content through output rails for improved performance. We've also seen added support for Prompt Security. Additionally, Nvidia released three new microservices: content safety NIM microservice, topic control NIM microservice and jailbreak detection, all of which have been integrated with NeMo Guardrails.

Based on its growing feature set and increased usage in production, we're moving NeMo Guardrails to Trial. We recommend reviewing the latest release notes for a complete overview of the changes since our last blip.

## 63. Nyx

*Trial*

Nyx is a versatile semantic release tool that supports a wide range of software engineering projects. It's language-agnostic and works with all major CI and SCM platforms, making it highly adaptable. While many teams use semantic versioning in trunk-based development, Nyx also supports workflows like Gitflow, OneFlow and GitHub Flow. One key advantage of Nyx in production is its automatic changelog generation, with built-in support for Conventional Commits.

As noted in previous Radar editions, we caution against development patterns that rely on long-lived branches (e.g., Gitflow, GitOps), as they introduce challenges that even powerful tools like Nyx cannot mitigate. We highly recommend trying Nyx in CI/CD workflows, especially for trunk-based development, where we've seen repeated success.

## 64. OpenRewrite

*Trial*

OpenRewrite continues to serve us well as a tool for large-scale refactorings that follow a set of rules such as moving to a new API version of a widely used library or applying updates to many services that were created from the same template. Support for languages beyond Java, notably JavaScript, has been introduced. With short LTS release cycles in frameworks like Angular, keeping projects updated to newer versions has become increasingly important. OpenRewrite supports this process effectively. Using an AI coding assistant is an alternative, but for rule-based changes, it's usually slower, more expensive and less reliable. We like that OpenRewrite comes bundled with a catalog of recipes (rules), which describe the changes to be made. The refactoring engine, bundled recipes and build tool plugins are open-source software, which makes it easier for teams to reach for OpenRewrite when they need it.

## 65. Plerion

*Trial*

Plerion is an AWS-focused cloud security platform that integrates with hosting providers to uncover risks, misconfigurations and vulnerabilities across your cloud infrastructure, servers and applications. Similar to Wiz, Plerion uses risk-based prioritization for detected issues, promising to let you "focus on the 1% that matters." Our teams report positive experiences with Plerion, noting it has provided our clients with significant insights and reinforced the importance of proactive security monitoring for their organizations.

## 66. Software engineering agents

*Trial*

Since we last wrote about software engineering agents six months ago, the industry still lacks a shared definition of the term "agent." However, a major development has emerged — not in fully autonomous coding agents (which remain unconvincing) but in supervised agentic modes within the IDE. These modes allow developers to drive implementation via chat, with tools not only modifying code in multiple files but also executing commands, running tests and responding to IDE feedback like linting or compile errors.

This approach, sometimes called chat-oriented programming (CHOP) or prompt-to-code, keeps developers in control while shifting more responsibility to AI than traditional coding assistants like auto-suggestions. Leading tools in this space include Cursor, Cline and Windsurf, with GitHub Copilot slightly behind but catching up. The usefulness of these agentic modes depends on both the model used (with Claude's Sonnet series the current state of the art) and how well the tool integrates with the IDE to provide a good developer experience.

We've found these workflows intriguing and promising, with a notable increase in coding speed. However, keeping problem scopes small helps developers better review AI-generated changes. This works best with low-abstraction prompts and AI-friendly codebases that are well-structured and properly tested. As these modes improve, they'll also heighten the risk of complacency with AI-generated code. To mitigate this, employ pair programming and other disciplined review practices, especially for production code.

## 67. Tuple

*Trial*

Tuple, a tool optimized for remote pair programming, was originally designed to fill the gap left by Slack's Screenhero. Since we last mentioned it in the Radar, it has seen wider adoption, addressed previous quirks and constraints and now supports Windows. A key improvement is enhanced desktop sharing with a built-in privacy feature, allowing users to hide private app windows (such as text messages) while sharing tools like the browser window. Previously, UI limitations made Tuple feel like a pair programming tool rather than a general collaboration tool. With these updates, users can now collaborate on content beyond the IDE.

However, it's important to note that the remote pair has access to the entire desktop. If not configured properly, this could be a security concern, especially if the pair is not trustworthy. We strongly recommend educating teams on Tuple's privacy settings, best practices and etiquette before use.

We encourage teams to try the latest version of Tuple in your development workflow. It aligns with our pragmatic remote pairing recommendation, offering low-latency pairing, an intuitive UX and significant usability improvements.

## 68. Turborepo

*Trial*

Turborepo helps manage large JavaScript or TypeScript monorepos by analyzing, caching, parallelizing and optimizing build tasks to speed up the process. In large monorepos, projects often depend on each other; rebuilding all dependencies for every change is inefficient and time-consuming, but Turborepo makes this easier. Unlike Nx, Turborepo's default setup uses multiple package.json files — one per project — which allows having dependencies with different versions (multiple versions of

React, for example) in a single monorepo, which Nx discourages. While this might be considered an anti-pattern, it does address certain use cases, like migrating from multi- to monorepo, where teams may temporarily require multiple versions of dependencies. In our experience, TurboRepo is quite simple to set up and performs well.

## 69. AnythingLLM

***Assess***

AnythingLLM is an open-source desktop application to chat with large documents or pieces of content, backed by out-of-the-box integration with LLMs and vector databases. It has a pluggable architecture for embedder models and can be used with most of the commercial LLMs as well as open-weight models that can be managed by Ollama. In addition to RAG, different skills can be created and organized as agents to perform custom tasks and workflows. It lets users organize the documents and interactions with them in different workspaces and they act as long lived threads with different contexts. Recently, it also became possible to deploy it as a multi-user web application with a simple Docker image. Some of our teams are using it as a local personal assistant and finding it a powerful and useful utility.

## 70. Gemma Scope

***Assess***

Mechanistic interpretability — understanding the inner workings of large language models — is becoming an increasingly important field. Tools like Gemma Scope and the open-source library Mishax provide insights into the Gemma2 family of open models. Interpretability tools play a crucial role in debugging unexpected behavior, identifying components responsible for hallucinations, biases or other failure cases, and ultimately building trust by offering deeper visibility into models. While this field may be of particular interest to researchers, it's worth noting that with the recent release of DeepSeek-R1, model training is becoming more feasible for companies beyond the established players. As GenAI continues to evolve, both interpretability and safety will only grow in importance.

## 71. Hurl

***Assess***

Hurl is a Swiss Army knife for making sequences of HTTP requests, defined in plain text files using Hurl-specific syntax. Beyond sending requests, Hurl can validate responses, ensuring a request returns a specific HTTP status code; assert conditions on response headers or content using XPATH, JSONPath or regular expressions; and extract response data into variables, which can then be used to chain requests.

With its feature set, Hurl is useful for simple API automations but also serves as an automated API testing tool. Its ability to generate detailed test reports in HTML or JSON enhances its utility for testing workflows. While dedicated tools like Bruno and Postman offer GUIs and additional features, we like Hurl for its simplicity. Like Bruno, which also uses plain text files, Hurl tests can be stored in the code repository.

## 72. Jujutsu

***Assess***

Git is the dominant distributed version control system (VCS), holding the vast majority of market share. Yet, despite over a decade of dominance, developers still struggle with its complex workflows for branching, merging, rebasing and conflict resolution. This ongoing frustration has fueled a wave of

tools designed to ease the pain — some offering visualizations to clarify complexity, others providing their own graphical interfaces to abstract it away entirely.

Jujutsu takes this a step further, offering a full-fledged alternative to Git while maintaining compatibility by using Git repositories as a storage backend. This allows developers to utilise existing Git servers and services while benefiting from Jujutsu's streamlined workflows. Positioned as "both simple and powerful," Jujutsu emphasizes ease of use for developers of all experience levels. One standout feature is its first-class conflict resolution, which has the potential to significantly improve the developer experience.

## 73. kubenetmon

***Assess***

Monitoring and understanding the network traffic associated with Kubernetes can prove a challenge, particularly when your infrastructure spans multiple zones, regions or clouds. kubenetmon, built by ClickHouse and recently open sourced, hopes to solve this problem by offering detailed Kubernetes data transfer metering across the major cloud providers. If you're running Kubernetes and have been frustrated by opaque data transfer costs on your bill it may be worth exploring kubenetmon.

## 74. Mergiraf

***Assess***

Resolving merge conflicts is probably one of the least liked activities in software development. And while there are techniques that reduce the complexity of merges — for example, practicing continuous integration in the original sense of merging to a shared mainline at least daily — we're seeing too much effort spent on merges. Long-lived feature branches are one culprit, but AI-assisted coding also has a tendency to increase the size of change sets. Help may come in the form of Mergiraf, a new tool that resolves merge conflicts by looking at the syntax tree rather than treating code as lines of text. As a git merge driver, it can be set up so that git subcommands like `merge` and `cherry-pick` automatically use Mergiraf instead of the default heuristics.

## 75. ModernBERT

***Assess***

The successor to BERT (Bidirectional Encoder Representations from Transformers), ModernBERT is a next-generation family of encoder-only transformer models designed for a wide range of natural language processing (NLP) tasks. As a drop-in replacement, ModernBERT improves both performance and accuracy while addressing some of BERT's limitations — notably including support for dramatically longer context lengths thanks to Alternating Attention. Teams with NLP needs should consider ModernBERT before defaulting to a general-purpose generative model.

## 76. OpenRouter

***Assess***

OpenRouter is a unified API for accessing multiple large language models. It provides a single integration point for mainstream LLM providers, simplifies experimentation, reduces vendor lock-in, and optimizes costs by routing requests to the most appropriate model. Popular tools like Cline and Open WebUI use OpenRouter as their endpoint. During our Radar discussion, we questioned whether most projects truly need to switch between models, given that OpenRouter must add price markup as a profit model on top of this encapsulation layer. However, we also recognize that OpenRouter

provides various load-balancing strategies to help optimize costs. One particularly useful feature is its ability to bypass API rate limits. If your application exceeds the rate limit of a single LLM provider, OpenRouter can help you break through this limitation and achieve better throughput.

## 77. Redactive

*Assess*

Redactive is an enterprise AI enablement platform designed to help regulated organizations securely prepare unstructured data for AI applications, such as AI-powered assistants and copilots. It integrates with content platforms like Confluence, creating secure text indices for retrieval-augmented generation (RAG) searches. By serving only live data and enforcing real-time user permissions from source systems, Redactive ensures AI models access accurate, authorized information without compromising security. Additionally, it provides engineering teams with tools to build AI use cases safely using any LLM. For organizations exploring AI-driven solutions, Redactive offers a streamlined approach to data preparation and compliance, balancing security and accessibility for teams experimenting with AI capabilities in a controlled environment.

## 78. System Initiative

*Assess*

We continue to be excited by System Initiative. This experimental tool represents a radical new direction for DevOps work. We really like the creative thinking that has gone into this tool and hope it will encourage others to break with the status quo of infrastructure-as-code approaches. System Initiative is now out of beta and available free and open source under an Apache 2.0 license. While the tool's developers use it to manage production infrastructure, it still has a way to go before it can scale to meet the demands of large enterprises. However, we continue to think it's worth checking out to experience a completely different approach to DevOps tooling.

## 79. TabPFN

*Assess*

TabPFN is a transformer-based model designed for fast and accurate classification on small tabular data sets. It leverages in-context learning (ICL) to make predictions directly from labeled examples without hyperparameter tuning or additional training. Pretrained on millions of synthetic data sets, TabPFN generalizes well across diverse data distributions and handles missing values and outliers effectively. Its strengths include efficient processing of heterogeneous data and robustness to uninformative features.

TabPFN is particularly suitable for small-scale applications where speed and accuracy are crucial. However, it faces scalability challenges with larger data sets and has limitations in handling regression tasks. As a cutting-edge solution, TabPFN is worth evaluating for its potential to outperform traditional models in tabular classification, especially where transformers are less commonly applied.

## 80. v0

*Assess*

v0 by Vercel is an AI tool for generating front-end code from a screenshot, Figma design or simple prompt. It supports React, Vue, shadcn and Tailwind among other front-end frameworks. Beyond AI-generated code, v0 offers a great user experience, including the ability to preview the generated code and deploy it to Vercel in one step. While building real-world applications involves integrating multiple functionalities beyond a single screen, v0 provides a solid way to prototype and can be used to initialize a starting point for developing complex applications.

## 81. Windsurf

*Assess*

Windsurf is an AI coding assistant by Codeium that stands out for its agentic capabilities. Similar to Cursor and Cline, it lets developers drive their implementation from an AI chat that navigates and changes code and executes commands. It frequently releases interesting new features and integrations for the agentic mode. Recently, for instance, it released a browser preview that makes it easy for the agent to access DOM elements and the browser console, and a web research capability that lets Windsurf look for documentation and solutions on the internet when appropriate. Windsurf provides access to a range of popular models, and users can activate and reference web search, library documentation and MCP integration as additional context providers.

## 82. YOLO

*Assess*

The YOLO (You Only Look Once) series, developed by Ultralytics, continues to advance computer vision models. The latest release, YOLO11, delivers significant improvements in both precision and efficiency over previous versions. YOLO11 can perform image classification at high speed with minimum resources, making it suitable for real-time applications in edge devices. We also found that the ability to use the same framework to do pose estimation, object detection, image segmentation and other tasks is very powerful. This significant development also reminds us that using 'traditional' machine-learning models for specific tasks can be more powerful than general AI models, such as LLMs.

# Languages and Frameworks

**Adopt**

83. OpenTelemetry
84. React Hook Form

**Trial**

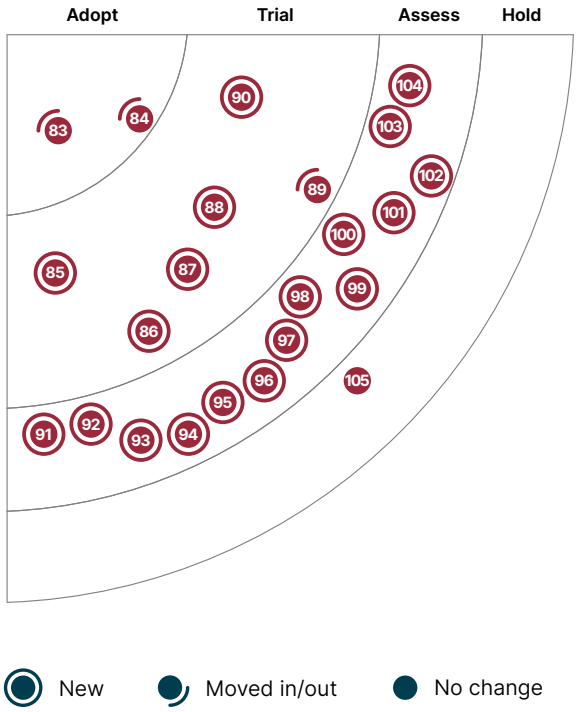
85. Effect
86. Hasura GraphQL engine
87. LangGraph
88. MarkItDown
89. Module Federation
90. Prisma ORM

**Assess**

91. .NET Aspire
92. Android XR SDK
93. Browser Use
94. CrewAI
95. ElysiaJs
96. FastGraphRAG
97. Gleam
98. GoFr
99. Java post-quantum cryptography
100. Presidio
101. PydanticAI
102. Swift for resource-constrained applications
103. Tamagui
104. torchtune

**Hold**

105. Node overload



## 83. OpenTelemetry

*Adopt*

OpenTelemetry is quickly becoming the industry standard for observability. The release of the OpenTelemetry protocol (OTLP) specification established a standardized way to handle traces, metrics and logs, reducing the need for multiple integrations or major rewrites as monitoring distributed solutions and interoperability requirements grow. As OpenTelemetry expands to support logs and profiling, OTLP ensures a consistent transport format across all telemetry data, simplifying instrumentation and making full-stack observability more accessible and scalable for microservices architectures.

Adopted by vendors like Datadog, New Relic and Grafana, OTLP enables organizations to build flexible, vendor-agnostic observability stacks without being locked into proprietary solutions. It supports gzip and zstd compression, reducing telemetry data size and lowering bandwidth usage — a key advantage for environments handling high volumes of telemetry data. Designed for long-term growth, OTLP ensures OpenTelemetry remains a robust and future-proof standard, solidifying its position as the de-facto choice for telemetry transport.

## 84. React Hook Form

*Adopt*

We noted React Hook Form as an alternative to Formik. By defaulting to uncontrolled components, it delivers significantly better out-of-the-box performance, especially for large forms. React Hook Form is well-integrated with various schema-based validation libraries, including Yup, Zod and more. Additionally, React Hook Form offers a lot of flexibility, making it easy to integrate with existing codebases and other libraries. You can use React Hook Form with external controlled components libraries such as shadcn or AntD. With strong performance, seamless integration and active development, it's a solid choice for building large form or form-heavy application.

## 85. Effect

*Trial*

Effect is a powerful TypeScript library for building complex synchronous and asynchronous programs. Web application development often requires boilerplate code for tasks such as asynchrony, concurrency, state management and error handling. Effect-TS streamlines these processes using a functional programming approach. Leveraging TypeScript's type system, Effect helps catch hard-to-detect issues at compile time. Our team previously used fp-ts for functional programming but found that Effect-TS provides abstractions that align more closely with daily tasks. It also makes code easier to combine and test. While traditional approaches like `Promise/try-catch` or `async/await` can handle such scenarios, after using Effect, our team found no reason to go back.

## 86. Hasura GraphQL engine

*Trial*

The Hasura GraphQL engine is a universal data access layer that simplifies building, running and governing high-quality APIs on different data sources. It provides instant GraphQL APIs over various databases (including PostgreSQL, MongoDB and ClickHouse) and data sources, enabling developers to fetch only the data they need quickly and securely. We found Hasura easy to implement GraphQL for server-side resource aggregation and have applied it in multiple data product projects. However, we remain cautious about its powerful federated query and unified schema management. A noteworthy recent addition is Hasura's PromptQL feature, which allows developers to leverage LLMs for more natural and intuitive data interactions.

## 87. LangGraph

*Trial*

LangGraph is an orchestration framework designed to build stateful multi-agent applications using LLMs. It provides a lower-level set of primitives like edges and nodes compared to LangChain's higher-level abstractions, offering developers fine-grained control over agent workflows, memory management and state persistence. This graph-based approach ensures predictable and customizable workflows, making debugging, scaling and maintaining production applications easier. Although it has a steeper learning curve, LangGraph's lightweight design and modularity make it a powerful framework for creating agentic applications.

## 88. MarkItDown

*Trial*

MarkItDown converts various formats (PDF, HTML, PowerPoint, Word) into Markdown, enhancing text readability and context retention. Since LLMs derive context from formatting cues like headings and sections, Markdown helps preserve structure for better comprehension. In RAG-based applications, our teams used MarkItDown to pre-process documents into Markdown, ensuring logical markers (headers, subsections) remained intact. Before embedding generation, structure-aware chunking helped maintain full section context which improves the clarity of query responses, especially for complex documents. Widely used for documentation, Markdown also makes MarkItDown's CLI a valuable developer productivity tool.

## 89. Module Federation

*Trial*

Module Federation allows for the specification of shared modules and dependency deduplication across micro frontends. With version 2.0, it has evolved to function independently of webpack. This update introduces key features, including a federation run time, a new plugin API and support for popular frameworks like React and Angular as well as popular bundlers like Rspack and Vite. By adopting Module Federation, large web applications can be divided into smaller, manageable micro frontends, allowing different teams to develop, deploy and scale independently while sharing dependencies and components efficiently.

## 90. Prisma ORM

*Trial*

Prisma ORM is an open-source database toolkit that simplifies working with databases in Node.js and TypeScript applications. It offers a modern, type-safe approach to database access, automates database schema migrations and provides an intuitive query API. Unlike typical ORMs, PrismaORM uses plain JavaScript objects to define database types without decorators or classes. Our experience with Prisma ORM is positive; we find it not only better aligns with the general TypeScript development landscape, it also neatly integrates with the functional programming paradigm.

## 91. .NET Aspire

*Assess*

.NET Aspire is designed to simplify the orchestration of distributed applications on a developer's local machine. Aspire lets you orchestrate multiple services in a local development environment — including multiple .NET projects, dependent databases and Docker containers — all with a single command. Furthermore, Aspire provides observability tools — including logging, tracing and metrics dashboards

— for local development, decoupled from the tools used in staging or production environments. This significantly improves the developer experience when building, tweaking and debugging the observability aspects of any system they are working on.

## 92. Android XR SDK

*Assess*

Google, in collaboration with Samsung and Qualcomm, has introduced Android XR, a new operating system designed for XR headsets. Support is planned for glasses and other devices. Most Android apps are supported with no or minimal changes, but the idea is to build new spatial apps from scratch or to "spatialize" existing apps. The new Android XR SDK is positioned as the go-to SDK for such projects, and Google provides guidance on how to choose tools and technologies bundled in the SDK. It's available in developer preview now.

## 93. Browser Use

*Assess*

Browser Use is an open-source python library that enables LLM-based AI agents to use web browsers and access web applications. It can control the browser and perform steps that include navigations, inputs and text extractions. With the ability to manage multiple tabs, it can perform coordinated actions across multiple web apps. It's useful for scenarios where LLM-based agents need to access web content, perform actions on it and get the results. The library can work with a variety of LLMs. It leverages Playwright to control the browser, combining visual understanding with HTML structure extraction for improved web interaction. This library is gaining traction in multi-agent scenarios, enabling agents to collaborate on complex workflows involving web interactions.

## 94. CrewAI

*Assess*

CrewAI is a platform designed to help you build and manage AI agents that can work together to accomplish complex tasks. Think of it as a way to create a crew of AI workers, each with their own special skills, who can collaborate to achieve a common goal. We've mentioned it previously in the Radar under LLM-powered autonomous agents. In addition to the open-source Python library, CrewAI now has an enterprise solution so organizations can create agent-based applications for real-world business cases, run them on their cloud infrastructure and connect to existing data sources such as Sharepoint or JIRA. We've used CrewAI multiple times to tackle production challenges, from automated validation of promotion codes to investigating transaction failures and customer support queries. While the agentic landscape continues to evolve rapidly, we're confident in placing CrewAI in Assess.

## 95. ElysiaJs

*Assess*

ElysiaJS is an end-to-end type-safe web framework for TypeScript, designed primarily for Bun but also compatible with other JavaScript run times. Unlike alternatives such as tRPC, which enforces specific API interface structures, ElysiaJS does not impose any API interface structure. This allows developers to create APIs that follow established industry practices such as RESTful, JSON:API or OpenAPI and also provide end-to-end type safety. ElysiaJS is highly performant when used with Bun run time, even comparable to Java or Go web frameworks in some benchmarks. ElysiaJS is worth considering, especially when building a backend-for-frontend (BFF).

## 96. FastGraphRAG

*Assess*

FastGraphRAG is an open-source implementation of GraphRAG designed for high retrieval accuracy and performance. It uses Personalized PageRank to limit graph navigation to the most relevant nodes among all the related nodes in the graph, enhancing retrieval accuracy and improving LLM response quality. It also provides a visual representation of the graph, helping users understand node relationships and the retrieval process. With support for incremental updates, it's well-suited to dynamic and evolving data sets. Optimized for large-scale GraphRAG use cases, FastGraphRAG improves performance while minimizing resource consumption.

## 97. Gleam

*Assess*

Erlang/OTP is a powerful platform for building highly concurrent, scalable and fault-tolerant distributed systems. Traditionally, its languages have been dynamically typed, but Gleam introduces type safety at the language level. Built on BEAM, Gleam combines the expressiveness of functional programming with compile-time type safety, reducing run-time errors and improving maintainability. With a modern syntax, it integrates well with the OTP ecosystem, leveraging the strengths of Erlang and Elixir while ensuring strong interoperability. The Gleam community is active and welcoming, and we look forward to its continued development.

## 98. GoFr

*Assess*

GoFr is a framework for building microservices in Golang, designed to simplify development by abstracting away boilerplate code for common microservice functionalities such as logging, traces, metrics, configuration management and Swagger API documentation. It supports multiple databases, handles database migrations and facilitates pub/sub with brokers like Kafka and NATs. Additionally, GoFr includes task scheduling with cron jobs. It reduces the complexity of building and maintaining microservices, and allows developers to focus on writing business logic rather than infrastructure concerns. Even though there are other popular Go libraries for building web APIs, GoFr is gaining traction and is worth exploring for Golang-based microservices.

## 99. Java post-quantum cryptography

*Assess*

At the core of asymmetric cryptography, which secures most modern communication, lies a mathematically hard problem. However, the problem used in today's algorithms will be easy to solve with quantum computers, driving research for alternatives. Lattice-based cryptography is currently the most promising candidate. Although cryptographically relevant quantum computers are still years away, post-quantum cryptography is worth considering for applications that must remain secure for decades. There is also the risk that encrypted data is recorded today in order to be decrypted once quantum computers become available.

Java post-quantum cryptography takes its first steps in JDK 24, set for general availability in late March. This release includes JEP 496 and JEP 497 — which implement a key encapsulation mechanism and a digital signature algorithm — both standards-based and designed to be resistant to future quantum computing attacks. While liboqs from the Open Quantum Safe project provides C-based implementations with a JNI wrapper, it's encouraging to see a native Java implementation emerging as well.

## 100. Presidio

*Assess*

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 PII entity recognition and de-identification, allowing businesses to tailor it to their specific privacy requirements. Although Presidio automates the identification of sensitive information, it's not foolproof and may miss or misidentify data. Exercise caution when relying on its results.

## 101. PydanticAI

*Assess*

As the technologies to build LLM-backed applications and agents continue to evolve rapidly, frameworks for building and orchestrating such applications often struggle to keep up or find the right timeless abstractions. PydanticAI is the latest entrant in this space, aiming to simplify implementations while avoiding unnecessary complexity. Developed by the creators of the popular Pydantic, it builds on lessons learned from earlier frameworks — many of which already rely on Pydantic. Rather than trying to be a Swiss Army knife, PydanticAI offers a lightweight yet powerful approach. It integrates with all major model APIs, includes built-in structured output handling and introduces a graph-based abstraction for managing complex agentic workflows.

## 102. Swift for resource-constrained applications

*Assess*

Since the release of Swift 6.0, the language has expanded beyond Apple's ecosystem with improved support for major operating systems, making it more viable to use Swift for resource-constrained applications. Traditionally, this space has been dominated by C, C++ and, more recently, Rust, due to their low-level control, high performance and availability of certified compilers and libraries that comply with standards such as MISRA, ISO 26262 and ASIL. While Rust has begun achieving similar certifications, Swift has yet to pursue this process, limiting its use in safety-critical applications.

Swift's growing adoption is driven by its balance of performance and safety features, including strong type safety and automatic reference counting for memory management. While Rust's ownership model offers stronger memory safety guarantees, Swift provides a different trade-off that some developers find more approachable. Both Swift and Rust share the LLVM/Clang compiler backend, allowing advancements in one to benefit the other. With its ability to compile to optimized machine code, its open-source development and its expanding cross-platform support, Swift is emerging as a contender for a wider range of applications — far beyond its iOS roots.

## 103. Tamagui

*Assess*

Tamagui is a library for efficiently sharing styles between React web and React Native. It offers a design system with reusable styled and unstyled components that render seamlessly across platforms. Its optional optimizing compiler boosts performance by converting styled components into atomic CSS with divs on the web and hoisted style objects on native views.

## 104. torchtune

*Assess*

torchtune is a PyTorch library for authoring, post-training and experimenting with LLMs. It supports single and multi-GPU setups and enables distributed training with FSDP2. The library provides YAML-based recipes for tasks like fine-tuning, inference, evaluation and quantization-aware training. Each recipe offers a focused set of features, avoiding complex flag-based configurations. It prioritizes simplicity, favoring code clarity over excessive abstractions. It also includes a CLI for downloading models, managing recipes and running experiments efficiently.

## 105. Node overload

*Hold*

A few years ago, we observed Node overload: Node.js was often used for questionable reasons or without even considering any alternatives. While we understand that some teams prefer a single-language stack — despite the trade-offs — we continue to advocate for polyglot programming. At the time, we noted that Node.js had a deserved reputation for efficiency in IO-heavy workloads, but we mentioned that other frameworks had caught up which offered better APIs and superior overall performance. We also cautioned that Node.js was never well-suited to compute-heavy workloads, a limitation that remains a significant challenge. Now, with the rise of data-heavy workloads, we're seeing teams struggle with these as well.

## 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