Technology Radar

Volume 30

April 2024

An opinionated guide to today's technology landscape

/thoughtworks

Strategy. Design. Engineering.

About the Radar

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

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

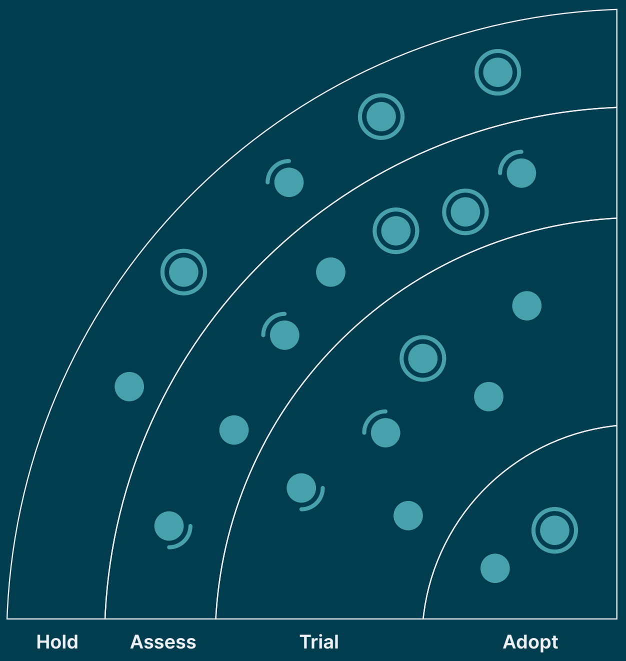
We encourage you to explore these technologies. The Radar is graphical in nature, grouping items into techniques, tools, platforms and languages and frameworks. When Radar items could appear in multiple quadrants, we chose the one that seemed most appropriate. We further group these items in four rings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

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

Radar at a glance

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

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



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

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

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

Hold: Proceed with caution.

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

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

Contributors

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

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 New York in February 2024.

Rebecca Parsons (CTO Emerita)

Rachel Laycock (CTO)

Martin Fowler (Chief Scientist)

Bharani Subramaniam

Birgitta Böckeler

Brandon Byars

Camilla Falconi Crispim

Erik Dörnenburg

Fausto de la Torre

Hao Xu

James Lewis

Marisa Hoenig

Maya Ormaza

Mike Mason

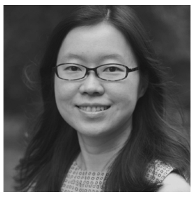
Neal Ford

Pawan Shah

Scott Shaw

Selvakumar Natesan

Shangqi Liu


Sofia Tania

Vanya Seth

Will Amaral

Themes

Open-ish source licenses

Two types of discussions about licenses arose during our meeting. First, for many years, the open-source software development ecosystem relied on a set of licenses, cataloged by the Open Source Initiative (OSI), with a small number of popular licenses used in most cases. Recently, however, we've seen churn in this previously serene landscape. Several prominent tools have recently garnered bad press, when their maintainers switched — in several cases abruptly — from an open-source license to a commercial model. We have no problem paying for software and are fine with the common model of commercial licenses for additional functionality. However, we find it problematic when core functionality of a widely used tool is suddenly put behind a paywall, especially when an ecosystem has developed around the tool. Second, the other interesting development concerns software that proclaims to be open source yet fundamental capabilities only appear after consumers pay subscriptions or other charges. Even though this business model has existed before, it seems to be exploited more with many of the shiny new AI tools — offering amazing capabilities a little too hidden under the fine print. We advise particular diligence around license issues. Pay attention to caveats and make sure that all files in a repository are covered by the license at the top level.

AI-assisted software development teams

The topic of AI obviously dominated our conversations; one-third of our blips concerned some aspect of it. While we discussed several developer-focused AI tools like GitHub Copilot, CodiumAI, aider and Continue, we also had numerous conversations about how the holistic use of AI across an entire team changes aspects of software development. We talked about a variety of tools that didn't make the final cut, including AI-assisted terminals like Warp, the ability to convert screenshots to code, ChatOps backed by LLMs and a host of other topics. Although the developer tools tend toward a higher degree of maturity, we suspect that all aspects of software development can gradually benefit from the pragmatic use of AI and derived tools, and we're actively following innovations across the development landscape. Of course, with the almost magical new capabilities offered by AI come new risks to software quality and security. This calls for teams to remain vigilant, including keeping non-developers in the loop about potential hazards.

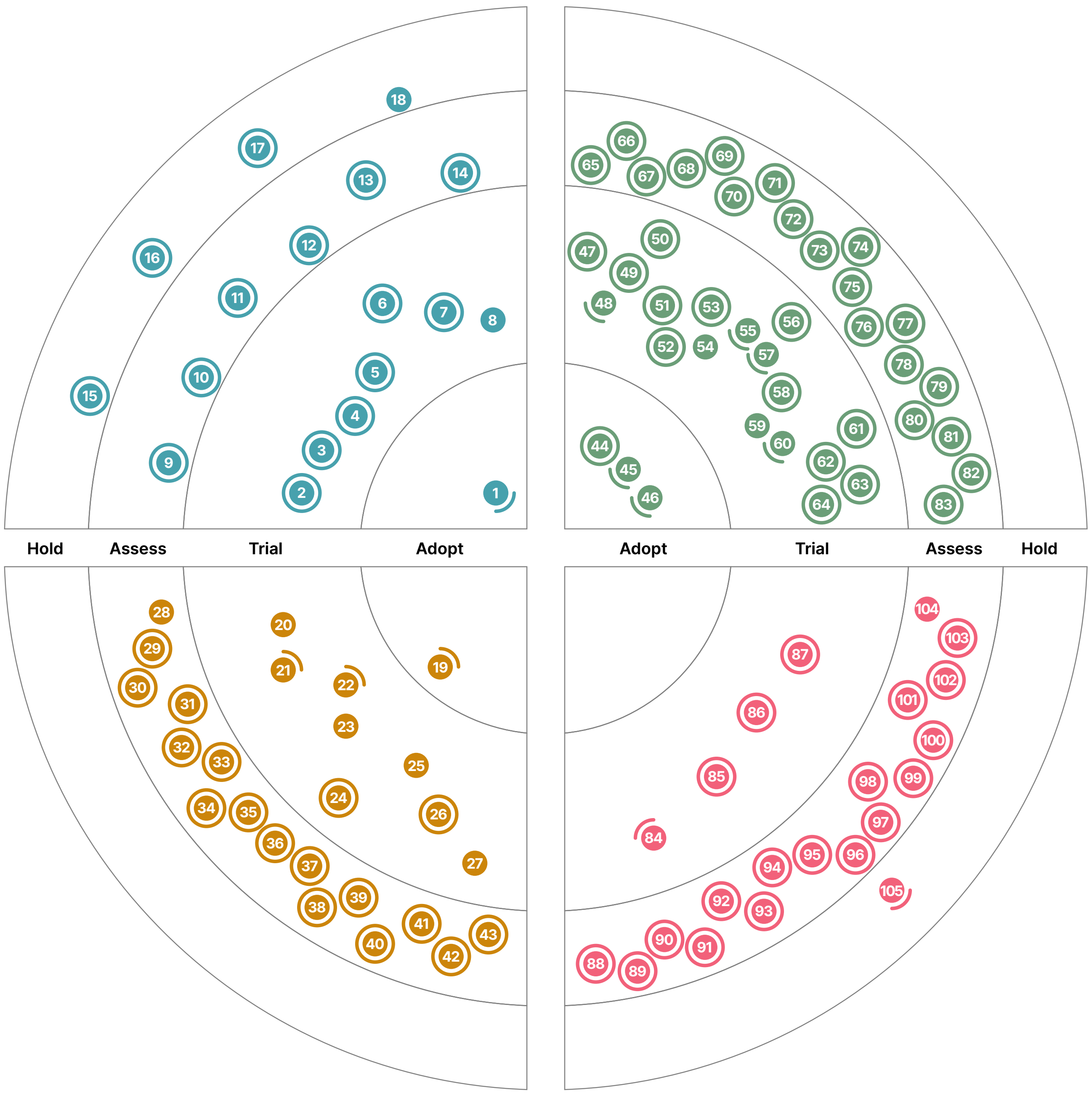
Emerging architecture patterns for LLMs

Patterns are popular in the technology world because they provide a succinct name for a solution to a particular problem. With the growing use of large language models (LLMs), we're starting to see the emergence of specific architecture patterns to support common contexts. For example, we discussed NeMo Guardrails, which allows developers to build governance policies around LLM usage. We also talked about tools such as Langfuse that allow greater observability into the steps leading to an LLM's output and how to deal with (and validate) bloated code bases full of generated code. We discussed how retrieval-augmented generation (RAG) is our preferred pattern to enhance the quality of LLM outputs, especially in the enterprise ecosystem. Additionally, we discussed techniques like using a lower-powered (and more cost-efficient) LLM to produce material which is selectively vetted by a more powerful (and expensive) LLM. Patterns form an important vocabulary for technologies, and we expect to see an explosion of patterns (and the inevitable anti-patterns) as generative AI continues to suffuse software development.

Dragging PRs closer to proper CI

Thoughtworks has always been a strong proponent of fast feedback loops during software development and thus a big supporter of continuous integration (CI). To assist with adoption, we built the first-ever CI server — CruiseControl — that was open-sourced in the late 1990s. Recently, our chief scientist Martin Fowler updated the canonical definition of continuous integration on his bliki to renew attention on this practice. However, many of our teams are compelled to ignore the CI part of CI/CD because they find themselves in situations where pull requests (PRs) are mandated. Although the practice of PRs was originally developed to manage massively distributed open-source teams and unreliable contributors, they've become a synonym for peer review commonplace even on small, close-knit delivery teams. In these circumstances, many developers yearn for the same sense of flow that they get from practicing actual CI. We surveyed several tools that are trying to alleviate the pains of PR review processes, including gitStream and Github merge queue. We also discussed techniques such as stacked diffs that hold promise for aligning with the core principles of CI by enabling more granular control over the integration process and discussed methods for deriving metrics from PRs to identify inefficiencies and bottlenecks during software delivery. Tooling helps tremendously in this space because of the trend toward generative AI for coding. With AI coding assistants, coding throughput increases, which leads to a tendency to create larger PRs. This puts even more pressure on asynchronous code review processes. Even though we still prefer the original practice of CI, we encourage teams who cannot use it because of external constraints to find ways to improve the accuracy of integration and the speed of their feedback cycle.

The Radar



Hold
Assess
Trial
Adopt
Adopt
Trial
Assess
Hold

New
Moved in/out
No change

The Radar

Techniques

Adopt

1. Retrieval-augmented generation (RAG)

Trial

2. Automatically generate Backstage entity descriptors
3. Combining traditional NLP with LLMs
4. Continuous compliance
5. Edge functions
6. Security champions
7. Text to SQL
8. Tracking health over debt

Assess

9. AI team assistants
10. Graph analysis for LLM-backed chats
11. LLM-backed ChatOps
12. LLM-powered autonomous agents
13. Using GenAI to understand legacy codebases
14. VISS

Hold

15. Broad integration tests
16. Overenthusiastic LLM use
17. Rush to fine-tune LLMs
18. Web components for SSR web apps

Platforms

Adopt

19. CloudEvents

Trial

20. Arm in the cloud
21. Azure Container Apps
22. Azure OpenAI Service
23. DataHub
24. Infrastructure orchestration platforms
25. Pulumi
26. Rancher Desktop
27. Weights & Biases

Assess

28. Bun
29. Chronosphere
30. DataOS
31. Dify
32. Elasticsearch Relevance Engine
33. FOCUS
34. Gemini Nano
35. HyperDX
36. IcePanel
37. Langfuse
38. Qdrant
39. RISC-V for embedded
40. Tigerbeetle
41. WebTransport
42. Zarf
43. ZITADEL

Hold

—

Tools

Adopt
44. Conan
45. Kaniko
46. Karpenter

Trial
47. 42Crunch API Conformance Scan
48. actions-runner-controller
49. Android Emulator Container
50. AWS CUDOS
51. aws-nuke
52. Bruno
53. Develocity
54. GitHub Copilot
55. Gradio
56. Gradle Version Catalog
57. Maestro
58. Microsoft SBOM tool
59. Open Policy Agent (OPA)
60. Philips's self-hosted GitHub runner
61. Pop
62. Renovate
63. Terrascan
64. Velero

Assess
65. aider
66. Akvorado
67. Baichuan 2
68. Cargo Lambda
69. Codium AI
70. Continue
71. Fern Docs
72. Granted
73. LinearB
74. LLaVA
75. Marimo
76. Mixtral
77. NeMo Guardrails
78. Ollama
79. OpenTofu
80. QAnything
81. System Initiative
82. Tetragon
83. Winglang

Hold

—

Languages and Frameworks

Adopt

—

Trial
84. Astro
85. DataComPy
86. Pinia
87. Ray

Assess
88. Android Adaptability
89. Concrete ML
90. Crabviz
91. Crux
92. Databricks Asset Bundles
93. Electric
94. LiteLLM
95. LLaMA-Factory
96. MLX
97. Mojo
98. Otter
99. Pkl
100. Rust for UI
101. vLLM
102. Voyager
103. WGPU
104. Zig

Hold
105. LangChain

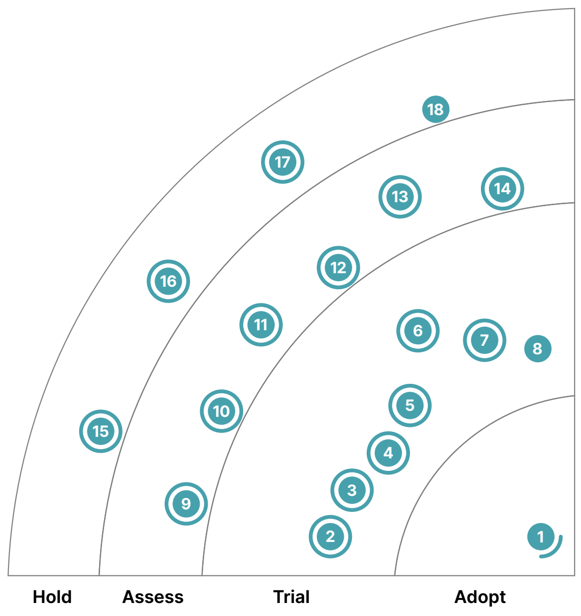
Techniques

Adopt
1. Retrieval-augmented generation (RAG)

Trial
2. Automatically generate Backstage entity descriptors
3. Combining traditional NLP with LLMs
4. Continuous compliance
5. Edge functions
6. Security champions
7. Text to SQL
8. Tracking health over debt

Assess
9. AI team assistants
10. Graph analysis for LLM-backed chats
11. LLM-backed ChatOps
12. LLM-powered autonomous agents
13. Using GenAI to understand legacy codebases
14. VISS

Hold
15. Broad integration tests
16. Overenthusiastic LLM use
17. Rush to fine-tune LLMs
18. Web components for SSR web apps



◉ New ◔ Moved in/out ● No change

1. Retrieval-augmented generation (RAG)

Adopt

Retrieval-augmented generation (RAG) is the preferred pattern for our teams to improve the quality of responses generated by a large language model (LLM). We've successfully used it in several projects, including the popular Jugalbandi AI Platform. With RAG, information about relevant and trustworthy documents — in formats like HTML and PDF — are stored in databases that supports a vector data type or efficient document search, such as pgvector, Qdrant or Elasticsearch Relevance Engine. For a given prompt, the database is queried to retrieve relevant documents, which are then combined with the prompt to provide richer context to the LLM. This results in higher quality output and greatly reduced hallucinations. The context window — which determines the maximum size of the LLM input — is limited, which means that selecting the most relevant documents is crucial. We improve the relevancy of the content that is added to the prompt by reranking. Similarly, the documents are usually too large to calculate an embedding, which means they must be split into smaller chunks. This is often a difficult problem, and one approach is to have the chunks overlap to a certain extent.

2. Automatically generate Backstage entity descriptors

Trial

Backstage from Spotify has been widely adopted across our client base as the preferred platform to host developer experience portals. Backstage, on its own, is just a shell that hosts plugins and provides an interface to manage the catalog of assets that make up a platform ecosystem. Any entity to be displayed or managed by Backstage is configured in the catalog-info file, which contains data such as status, lifecycle, dependencies and APIs among other details. By default, individual entity descriptors are written by hand and usually maintained and versioned by the team responsible for the component in question. Keeping the descriptors up to date can be tedious and create a barrier to developer adoption. Also, there is always the possibility that changes are overlooked or that some components are missed entirely. We've found it more efficient and less error-prone to automatically generate Backstage entity descriptors. Most organizations have existing sources of information that can jump-start the process of populating catalog entries. Good development practices, for example, putting appropriate tags on AWS resources or adding metadata to source files, can simplify entity discovery and descriptor generation. These automated processes can then be run on a regular basis — once a day, for example — to keep the catalog fresh and up to date.

3. Combining traditional NLP with LLMs

Trial

Large language models (LLMs) are the Swiss Army knives of natural language processing (NLP). But they're also quite expensive and not always the best tool for the job — sometimes it's more effective to use a proper corkscrew. Indeed, there's a lot of potential in combining traditional NLP with LLMs, or in building multiple NLP approaches in conjunction with LLMs to implement use cases and leverage LLMs for the steps where you actually need their capabilities. Traditional data science and NLP approaches for document clustering, topic identification and classification and even summarization are cheaper and can be more effective for solving a part of your use case problem. We then use LLMs when we need to generate and summarize longer texts, or combine multiple large documents, to take advantage of the LLM's superior attention span and memory. For example, we've successfully used this combination of techniques to generate a comprehensive trends report for a domain from a large corpus of individual trend documents, using traditional clustering alongside the generative power of LLMs.

4. Continuous compliance

Trial

Continuous compliance is the practice of ensuring that software development processes and technologies comply with industry regulations and security standards on an ongoing basis, by heavily leveraging automation. Manually checking for security vulnerabilities and adhering to regulations can slow down development and introduce errors. As an alternative, organizations can automate compliance checks and audits. They can integrate tools into software development pipelines, allowing teams to detect and address compliance issues early in the development process. Codifying compliance rules and best practices helps enforce policies and standards consistently across teams. It enables you to scan code changes for vulnerabilities, enforce coding standards and track infrastructure configuration changes to ensure they meet compliance requirements. Lastly, automated reporting of the above simplifies audits and provides clear evidence of compliance. We've already talked about techniques like publishing SBOMs and applying the recommendations from SLSA — they can be very good starting points. The benefits of this technique are multifold. First, automation leads to more secure software by identifying and mitigating vulnerabilities early and, second, development cycles accelerate as manual tasks are eliminated. Reduced costs and enhanced consistency are additional perks. For safety-critical industries like software-driven vehicles, automated continuous compliance can improve the efficiency and reliability of the certification process, ultimately leading to safer and more reliable vehicles on the road.

5. Edge functions

Trial

Although not a new concept, we've noticed the growing availability and use of decentralized code execution via content delivery networks (CDNs). Services such as Cloudflare Workers or Amazon CloudFront Edge Functions provide a mechanism to execute snippets of serverless code close to the client's geographic location. Edge functions not only offer lower latency if a response can be generated at the edge,they also present an opportunity to rewrite requests and responses in a location-specific way on their way to and from the regional server. For example, you might rewrite a request URL to route to a specific server that has local data relevant to a field found in the request body. This approach is best suited to short, fast-running stateless processes since the computational power at the edge is limited.

6. Security champions

Trial

Security champions are team members who think critically about security repercussions of both technical and nontechnical delivery decisions. They raise these questions and concerns with team leadership and have a firm understanding of basic security guidelines and requirements. They help development teams approach all activities during software delivery with a security mindset, thus reducing the overall security risks for the systems they develop. A security champion is not a separate position but a responsibility assigned to an existing member of the team who is guided by appropriate training from security practitioners. Equipped with this training, security champions improve the security awareness of the team by spreading knowledge and acting as a bridge between the development and security teams. One great example of an activity security champions can help drive within the team is threat modeling, which helps teams think about security risks from the start. Appointing and training a security champion on a team is a great first step, but relying solely on champions without proper commitment from leaders can lead to problems. Building a security mindset, in our experience, requires commitment from the entire team and managers.

7. Text to SQL

Trial

Text to SQL is a technique that converts natural language queries into SQL queries that can be executed by a database. Although large language models (LLMs) can understand and transform natural language, creating accurate SQL for your own schema can be challenging. Enter Vanna, an open-source Python retrieval-augmented generation (RAG) framework for SQL generation. Vanna works in two steps: first you create embeddings with the data definition language statements (DDLs) and sample SQLs for your schema, and then you ask questions in natural language. Although Vanna can work with any LLMs, we encourage you to assess NSQL, a domain-specific LLM for text-to-SQL tasks.

8. Tracking health over debt

Trial

We keep experiencing the improvements teams make to their ecosystem by treating the health rating the same as other service-level objectives (SLOs) and prioritizing enhancements accordingly, instead of solely focusing on tracking technical debt. By allocating resources efficiently to address the most impactful issues related to health, teams and organizations can reduce long-term maintenance costs and evolve products more efficiently. This approach also enhances communication between technical and nontechnical stakeholders, fostering a common understanding of the system's state. Although metrics may vary among organizations (see this blog post for examples) they ultimately contribute to long-term sustainability and ensure software remains adaptable and competitive. In a rapidly changing digital landscape, focusing on tracking health over debt of systems provides a structured and evidence-based strategy to maintain and enhance them.

9. AI team assistants

Assess

AI coding assistance tools like GitHub Copilot are currently mostly talked about in the context of assisting and enhancing an individual's work. However, software delivery is and will remain team work, so you should be looking for ways to create AI team assistants to help create the "10x team," as opposed to a bunch of siloed AI-assisted 10x engineers. We've started using a team assistance approach that can increase knowledge amplification, upskilling and alignment through a combination of prompts and knowledge sources. Standardized prompts facilitate the use of agreed-upon best practices in the team context, such as techniques and templates for user story writing or the implementation of practices like threat modeling. In addition to prompts, knowledge sources made available through retrieval-augmented generation provide contextually relevant information from organizational guidelines or industry-specific knowledge bases. This approach gives team members access to the knowledge and resources they need just in time.

10. Graph analysis for LLM-backed chats

Assess

Chatbots backed by large language models (LLMs) are gaining a lot of popularity right now, and we're seeing emerging techniques around productionizing and productizing them. One such productization challenge is understanding how users are conversing with a chatbot that is driven by something as generic as an LLM, where the conversation can go in many directions. Understanding the reality of conversation flows is crucial to improving the product and improving conversion rates. One technique to tackle this problem is to use graph analysis for LLM-backed chats. The agents that support a chat with a specific desired outcome — such as a shopping action or a successful resolution of a customer's problem — can usually be represented as a desired state machine. By loading all conversations into a graph, you can analyze actual patterns and look for discrepancies to the expected state machine. This helps find bugs and opportunities for product improvement.

11. LLM-backed ChatOps

Assess

LLM-backed ChatOps is an emerging application of large language models through a chat platform (primarily Slack) that allows engineers to build, deploy and operate software via natural language. This has the potential to streamline engineering workflows by enhancing the discoverability and user-friendliness of platform services. At the time of writing, two early examples are PromptOps and Kubiya. However, considering the finesse needed for production environments, organizations should thoroughly evaluate these tools before allowing them anywhere near production.

12. LLM-powered autonomous agents

Assess

LLM-powered autonomous agents are evolving beyond single agents and static multi-agent systems with the emergence of frameworks like Autogen and CrewAI. These frameworks allow users to define agents with specific roles, assign tasks and enable agents to collaborate on completing those tasks through delegation or conversation. Similar to single-agent systems that emerged earlier, such as AutoGPT, individual agents can break down tasks, utilize preconfigured tools and request human input. Although still in the early stages of development, this area is developing rapidly and holds exciting potential for exploration.

13. Using GenAI to understand legacy codebases

Assess

Generative AI (GenAI) and large language models (LLMs) can help developers both write and understand code. In practical application, this is so far mostly limited to smaller code snippets, but more products and technology developments are emerging for using GenAI to understand legacy codebases. This is particularly useful in the case of legacy codebases that aren't well-documented or where the documentation is outdated or misleading. For example, Driver AI or bloop use RAG approaches that combine language intelligence and code search with LLMs to help users find their way around a codebase. Emerging models with larger and larger context windows will also help to make these techniques more viable for sizable codebases. Another promising application of GenAI for legacy code is in the space of mainframe modernization, where bottlenecks often form around reverse engineers who need to understand the existing codebase and turn that understanding into requirements for the modernization project. Using GenAI to assist those reverse engineers can help them get their work done faster.

14. VISS

Assess

Zoom recently open-sourced its Vulnerability Impact Scoring System, or VISS. This system is mainly focused on vulnerability scoring that prioritizes actual demonstrated security measures. VISS differs from the Common Vulnerability Scoring System (CVSS) by not focusing on worst-case scenarios and attempting to more objectively measure the impact of vulnerabilities from a defender's perspective. To this aim, VISS provides a web-based UI to calculate the vulnerability score based on several parameters — categorized into platform, infrastructure and data groups — including the impact on the platform, the number of tenants impacted, data impact and more. Although we don't have too much practical experience with this specific tool yet, we think this kind of priority-tailored assessment approach based on industry and context is worth practicing.

15. Broad integration tests

Hold

While we applaud a focus on automated testing, we continue to see numerous organizations over-invested in what we believe to be ineffective broad integration tests. As the term "integration test" is ambiguous, we've taken the broad classification from Martin Fowler's bliki entry on the subject which indicates a test that requires live versions of all run-time dependencies. Such a test is obviously expensive, because it requires a full-featured test environment with all the necessary infrastructure, data and services. Managing the right versions of all those dependencies requires significant coordination overhead, which tends to slow down release cycles. Finally, the tests themselves are often fragile and unhelpful. For example, it takes effort to determine if a test failed because of the new code, mismatched version dependencies or the environment, and the error message rarely helps pinpoint the source of the error. Those criticisms don't mean that we take issue with automated "black box" integration testing in general, but we find a more helpful approach is one that balances the need for confidence with release frequency. This can be done in two stages by first validating the behavior of the system under test assuming a certain set of responses from run-time dependencies, and then validating those assumptions. The first stage uses service virtualization to create test doubles of run-time dependencies and validates the behavior of the system under test. This simplifies test data management concerns and allows for deterministic tests. The second stage uses contract tests to validate those environmental assumptions with real dependencies.

16. Overenthusiastic LLM use

Hold

In the rush to leverage the latest in AI, many organizations are quickly adopting large language models (LLMs) for a variety of applications, from content generation to complex decision-making processes. The allure of LLMs is undeniable; they offer a seemingly effortless solution to complex problems, and developers can often create such a solution quickly and without needing years of deep machine learning experience. It can be tempting to roll out an LLM-based solution as soon as it's more or less working and then move on. Although these LLM-based proofs of value are useful, we advise teams to look carefully at what the technology is being used for and to consider whether an LLM is actually the right end-stage solution. Many problems that an LLM can solve — such as sentiment analysis or content classification — can be solved more cheaply and easily using traditional natural language processing (NLP). Analyzing what the LLM is doing and then analyzing other potential solutions not only mitigates the risks associated with overenthusiastic LLM use but also promotes a more nuanced understanding and application of AI technologies.

17. Rush to fine-tune LLMs

Hold

As organizations are looking for ways to make large language models (LLMs) work in the context of their product, domain or organizational knowledge, we're seeing a rush to fine-tune LLMs. While fine-tuning an LLM can be a powerful tool to gain more task-specificity for a use case, in many cases it's not needed. One of the most common cases of a misguided rush to fine-tuning is about making an LLM-backed application aware of specific knowledge and facts or an organization's codebases. In the vast majority of these cases, using a form of retrieval-augmented generation (RAG) offers a better solution and a better cost-benefit ratio. Fine-tuning requires considerable computational resources and expertise and introduces even more challenges around sensitive and proprietary data than RAG. There is also a risk of underfitting, when you don't have enough data available for fine-tuning, or, less frequently, overfitting, when you have too much data and are therefore not hitting the right balance of task specificity that you need. Look closely at these trade-offs and consider the alternatives before you rush to fine-tune an LLM for your use case.

18. Web components for SSR web apps

Hold

With the adoption of frameworks like Next.js and htmx, we're seeing more usage of server-side rendering (SSR). As a browser technology, it's not trivial to use web components on the server. Frameworks have sprung up to make this easier, sometimes even using a browser engine, but the complexity is still there. Our developers find themselves needing workarounds and extra effort to order front-end components and server-side components. Worse than the developer experience is the user experience: page load performance is impacted when custom web components have to be loaded and hydrated in the browser, and even with pre-rendering and careful tweaking of the component, a "flash of unstyled content" or some layout shifting is all but unavoidable. As mentioned in the previous Radar, one of our teams had to move their design system away from the web components-based Stencil because of these issues. Recently, we received reports from another team that they ended up replacing server-side–generated components with browser-side components because of the development complexity. We caution against the use of web components for SSR web apps, even if supported by frameworks.

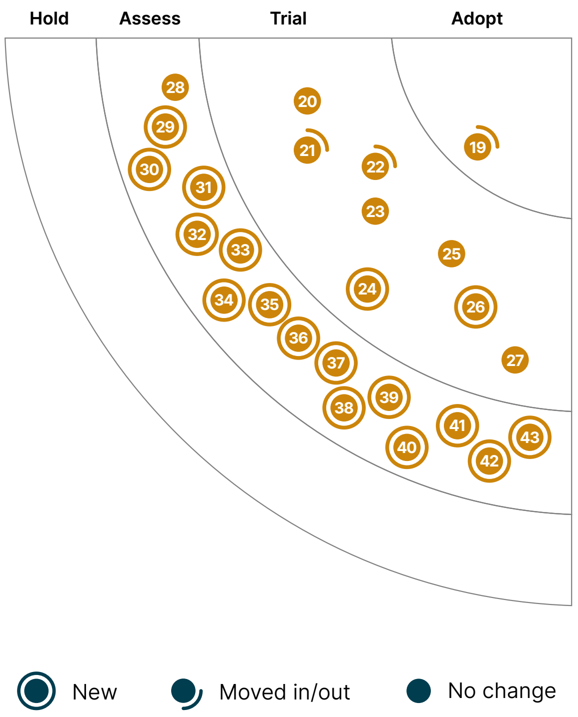
Platforms

Adopt
19. CloudEvents

Trial
20. Arm in the cloud
21. Azure Container Apps
22. Azure OpenAI Service
23. DataHub
24. Infrastructure orchestration platforms
25. Pulumi
26. Rancher Desktop
27. Weights & Biases

Assess
28. Bun
29. Chronosphere
30. DataOS
31. Dify
32. Elasticsearch Relevance Engine
33. FOCUS
34. Gemini Nano
35. HyperDX
36. IcePanel
37. Langfuse
38. Qdrant
39. RISC-V for embedded
40. Tigerbeetle
41. WebTransport
42. Zarf
43. ZITADEL

Hold
—



19. CloudEvents

Adopt

Events are common mechanisms in event-driven architecture or serverless applications. However, producers or cloud providers tend to support them in different forms, which prevents interoperability across platforms and infrastructures. CloudEvents is a specification for describing event data in common formats to provide interoperability across services, platforms and systems. It provides SDKs in multiple languages so you can embed the spec into your application or toolchain. Our teams use it not only for cross-cloud platform purposes but also for domain event specification, among other scenarios. CloudEvents is hosted by the Cloud Native Computing Foundation (CNCF) and is now a graduated project. Our teams default to using CloudEvents for building event-driven architectures and for that reason we're moving it to Adopt.

20. Arm in the cloud

Trial

Arm compute instances in the cloud have become increasingly popular in recent years due to their cost and energy efficiency compared to traditional x86-based instances. Many cloud providers now offer Arm-based instances, including AWS, Azure and GCP. The cost benefits of running Arm in the cloud can be particularly beneficial for businesses that run large workloads or need to scale. We're seeing many teams of ours moving to Arm instances for workloads like JVM services and even databases (including RDS) without any change in the code and minimal changes in the build scripts. New cloud-based applications and systems increasingly default to Arm in the cloud. Based on our experiences, we recommend Arm compute instances for all workloads unless there are architecture-specific dependencies. The tooling to support multiple architectures, such as multi-arch Docker images, also simplifies build and deploy workflows.

21. Azure Container Apps

Trial

Azure Container Apps is a managed Kubernetes application platform that streamlines the deployment of containerized workloads. In comparison to Azure Kubernetes Service (AKS), the operational and administrative burden of running containerized applications is reduced, but this comes at the expense of some flexibility and control, which is a trade-off teams need to consider. Another product in this area, Azure Container Instances, is usually too limited for production use. Our teams started using Azure Container Apps last year, when it was still in public preview, with good results, even when running large containers. Now that it is generally available, we're considering it for more use cases. Both Dapr and the KEDA Autoscaler are supported.

22. Azure OpenAI Service

Trial

Azure OpenAI Service provides access to OpenAI's GPT-4, GPT-35-Turbo, Embeddings, DALL-E model and more through a REST API, a Python SDK and web-based interface. The models can be adapted to tasks such as content generation, summarization, semantic search and translating natural language to code. Fine-tuning is also available via few-shot learning and the customization of hyperparameters. In comparison to OpenAI's own API, Azure OpenAI Service benefits from Azure's enterprise-grade security and compliance features, is available for more regions (although availability is limited for each of the larger geographic regions) and supports private networking, content filtering and manual model version control. For these reasons and our positive experience with it, we recommend that enterprises already using Azure consider using Azure OpenAI Service instead of the OpenAI API.

23. DataHub

Trial

When you build data products using data product thinking, it's essential to consider data lineage, data discoverability and data governance. Our teams have found that DataHub can provide particularly useful support here. Although earlier versions of DataHub required you to fork and manage the sync from the main product (if there was a need to update the metadata model), improvements in recent releases have introduced features that allow our teams to implement custom metadata models with a plugin-based architecture. Another useful feature of DataHub is the robust end-to-end data lineage from source to processing to consumption. DataHub supports both push-based integration as well as pull-based lineage extraction that automatically crawls the technical metadata across data sources, schedulers, orchestrators (scanning the Airflow DAG), processing pipeline tasks and dashboards, to name a few. As an open-source option for a holistic data catalog, DataHub is emerging as a default choice for our teams.

24. Infrastructure orchestration platforms

Trial

In-house infrastructure orchestration codebases frequently become a time sink to maintain and troubleshoot. Infrastructure orchestration platforms are appearing, promising to standardize and productize various aspects of infrastructure code delivery and deployment workflows. These include build tools like Terragrunt and Terraspace, services from IaC tool vendors such as Terraform Cloud and Pulumi Cloud as well as tool-agnostic platforms and services like env0 and Spacelift. There is a rich ecosystem of Terraform-specific orchestration tools and services, often called TACOS (Terraform Automation and Collaboration Software), including Atlantis, Digger, Scalr, Terramate and Terrateam. Each of these platforms enables different workflows, including GitOps, Continuous Delivery and compliance as code. We welcome the growth of solutions in this space. We recommend infrastructure and platform engineering teams explore how to use them to reduce the amount of non-differentiating custom code they need to develop and maintain their infrastructure. Standardization of how infrastructure code is structured, shared, delivered and deployed should also create opportunities for the emergence of an ecosystem of compatible tools for testing, measuring and monitoring infrastructure.

25. Pulumi

Trial

Tooling in the infrastructure-as-code space continues to evolve, and we're pleased to see that Pulumi is no exception to this trend. The platform recently added support for Java and YAML, for managing infrastructure at scale as well as for a multitude of cloud configurations and integrations, making the platform even more compelling. For our teams, it's still the main alternative to Terraform for developing code for multiple cloud platforms.

26. Rancher Desktop

Trial

Changes in licensing for Docker Desktop have left us scrambling for alternatives for running a fleet of containers on a developer's local laptop. Recently we've had good success with Rancher Desktop. This free and open-source app is relatively easy to download and install for Apple, Windows or Linux machines and provides a handy local Kubernetes cluster with a GUI for configuration and monitoring. Although Colima has become our Docker Desktop alternative of choice, it's primarily a CLI tool. In contrast, Rancher Desktop will appeal to those who don't want to give up the graphical interface that Docker Desktop provides. Like Colima, Rancher Desktop allows you to choose between dockerd or containerd as the underlying container run time. The choice of direct containerd frees you from the DockerCLI, but the dockerd option provides compatibility with other tools that depend on it to communicate with the run-time daemon.

27. Weights & Biases

Trial

Weights & Biases is a machine learning (ML) platform for building models faster through experiment tracking, data set versioning, visualizing model performance and model management. It can be integrated with existing ML code to get live metrics, terminal logs and system statistics streamed to the dashboard for further analysis. Recently, Weights & Biases has expanded into LLM observability with Traces. Traces visualizes the execution flow of prompt chains as well as intermediate inputs/outputs and provides metadata around chain execution (such as tokens used and start and end time). Our teams find it useful for debugging and getting a greater understanding of the chain architecture.

28. Bun

Assess

Bun is a new JavaScript run time, similar to Node.js or Deno. Unlike Node.js or Deno, however, Bun is built using WebKit's JavaScriptCore instead of Chrome's V8 engine. Designed as a drop-in replacement for Node.js, Bun is a single binary (written in Zig) that acts as a bundler, transpiler and package manager for JavaScript and TypeScript applications. Since our last volume, Bun has gone from beta into a stable 1.0 release. Bun has been built from the ground up with several optimizations — including fast startup, improved server-side rendering and a much faster alternative package manager — and we encourage you to assess it for your JavaScript run-time engine.

29. Chronosphere

Assess

When managing distributed architectures, accounting for the cost of sorting, indexing and accessing data is as critical as observability. Chronosphere takes a unique approach to cost management, tracking the use of observability data so that organizations can consider the cost-value trade-offs of various metrics. With the help of the Metrics Usage Analyzer, part of the Chronosphere Control Plane, teams can identify and exclude metrics they rarely (or never) use, thus yielding significant cost savings by reducing the amount of data organizations have to comb through. Given these advantages, as well as the ability of Chronosphere to match the functionality of other observability tools for cloud-hosted solutions, we believe it to be a compelling option for organizations to look into.

30. DataOS

Assess

With data mesh adoption on the rise, our teams have been on the lookout for data platforms that treat data products as a first-class entity. DataOS is one such product. It provides end-to-end lifecycle management to design, build, deploy and evolve data products. It offers standardized declarative specs written in YAML that abstract the low-level complexity of infrastructure setup and allow developers to define the data products easily via CLI/API. It supports access control policies with ABAC and data policies for filtering and masking data. Another notable feature is its ability to federate data across a variety of data sources, which reduces data duplication and the movement of data to a central place. DataOS fits best for greenfield scenarios where it does the heavy lifting since it provides an out-of-the-box solution for data governance, data discoverability, infrastructure resource management and observability. For brownfield scenarios, the ability to orchestrate resources outside of DataOS (for example, data stacks like Databricks) is in its nascent stage and still evolving. If your ecosystem doesn't exert a lot of opinion on data tooling, DataOS is a good way to expedite your journey for building, deploying and consuming data products in an end-to-end fashion.

31. Dify

Assess

Dify is a UI-driven platform for developing large language model (LLM) applications that makes prototyping them even more accessible. It supports the development of chat and text generation apps with prompt templates. Additionally, Dify supports retrieval-augmented generation (RAG) with imported data sets and can work with multiple models. We're excited about this category of applications. Based on our experience, however, Dify is not quite ready for prime time yet, because some features are buggy or don't seem fully fleshed out. At the moment, though, we're not aware of a competitor that is better.

32. Elasticsearch Relevance Engine

Assess

Although vector databases have been gaining popularity for retrieval-augmented generation (RAG) use cases, research and experience reports suggest combining traditional full-text search with vector search (into a hybrid search) can yield superior results. Through Elasticsearch Relevance Engine (ESRE), the well-established full-text search platform Elasticsearch supports built-in and custom embedding models, vector search and hybrid search with ranking mechanisms such as Reciprocal Rank Fusion. Even though this space is still maturing, in our experience, using these ESRE features along with the traditional filtering, sorting and ranking capabilities that come with Elasticsearch has yielded promising results, suggesting that established search platforms that support semantic search are not to be passed over.

33. FOCUS

Assess

Cloud and SaaS billing data can be complex, inconsistent among providers and difficult to understand. The FinOps Open Cost and Usage Specification (FOCUS) aims to reduce this friction with a spec containing a set of terminologies (aligned with the FinOps framework), a schema and a minimum set of requirements for billing data. The spec is intended to support use cases common to a variety of FinOps practitioners. Although still in the early stages of development and adoption, it's worth watching because, with growing industry adoption, FOCUS will make it easier for platforms and end users to get a holistic view of cloud spend across a long tail of cloud and SaaS providers.

34. Gemini Nano

Assess

Google's Gemini is a family of foundational LLMs designed to run on a wide range of hardware, from data centers to mobile phones. Gemini Nano has been specifically optimized and scaled down to run on mobile silicon accelerators. It enables capabilities such as high-quality text summarization, contextual smart replies and advanced grammar correction. For example, the language understanding of Gemini Nano enables the Pixel 8 Pro to summarize content in the Recorder app. Running on-device removes many of the latency and privacy concerns associated with cloud-based systems and allows the features to work without network connection. Android AICore simplifies the integration of the model into Android apps, but only a few devices are supported at the time of writing.

35. HyperDX

Assess

HyperDX is an open-source observability platform that unifies all three pillars of observability: logs, metrics and tracing. With it, you can correlate end-to-end and go from browser session replay to logs and traces in just a few clicks. The platform leverages ClickHouse as a central data store for all telemetry data, and it scales to aggregate log patterns and condense billions of events into distinctive clusters. Although you can choose from several observability platforms, we want to highlight HyperDX for its unified developer experience.

36. IcePanel

Assess

IcePanel facilitates collaborative architectural modeling and diagramming using the C4 model, which allows technical and business stakeholders to zoom in to the level of technical detail they need. It supports modeling architecture objects whose metadata and connections can be reused across diagrams, along with the visualization of flows between those objects. Versioning and tagging allows collaborators to model different architecture states (e.g., as-is versus to-be) and track user-defined classifications of various parts of the architecture. We're keeping an eye on IcePanel for its potential to improve architecture collaboration, particularly for organizations with complex architectures. For an alternative that better supports diagrams as code, check out Structurizr.

37. Langfuse

Assess

Langfuse is an engineering platform for observability, testing and monitoring large language model (LLM) applications. Its SDKs support Python, JavaScript and TypeScript, OpenAI, LangChain and LiteLLM among other languages and frameworks. You can self-host the open-source version or use it as a paid cloud service. Our teams have had a positive experience, particularly in debugging complex LLM chains, analyzing completions and monitoring key metrics such as cost and latency across users, sessions, geographies, features and model versions. If you're looking to build data-driven LLM applications, Langfuse is a good option to consider.

38. Qdrant

Assess

Qdrant is an open-source vector database written in Rust. In the September 2023 edition of the Radar, we talked about pgvector, a PostgreSQL extension for vector search. However, if you have to scale the vector database horizontally across nodes, we encourage you to assess Qdrant. It has built-in single instruction, multiple data (SIMD) acceleration support for improved search performance, and it helps you associate JSON payloads with vectors.

39. RISC-V for embedded

Assess

While the Arm architecture continues to expand its impact — we've updated our assessment of Arm in the cloud in this edition — interest in the newer and less established RISC-V architecture also grows. RISC-V doesn't bring breakthroughs in performance or efficiency — in fact, its per-watt performance is similar to Arm's, and it can't quite compete on absolute performance — but it's open source, modular and not tied to a single company. This makes it an attractive proposition for embedded systems, where the cost of licensing proprietary architectures is a significant concern. This is also why the field of RISC-V for embedded is maturing, and several companies, including SiFive and espressif, are offering development boards and SoCs for a wide range of applications. Microcontrollers and microprocessors capable of running the Linux kernel are available today, along with the corresponding software stack and toolchains. We're keeping an eye on this space and expect to see more adoption in the coming years.

40. Tigerbeetle

Assess

Tigerbeetle is an open-source distributed database for financial accounting. Unlike other databases, it's designed to be a domain-specific state machine for safety and performance. The state from one node in the cluster is replicated in a deterministic order to other nodes via the Viewstamped Replication consensus protocol. We quite like the design decisions behind Tigerbeetle to implement double-entry bookkeeping with strict serializability guarantees. It's a relatively new and actively evolving database, but not quite ready for production.

41. WebTransport

Assess

WebTransport is a protocol that builds on top of HTTP/3 and offers bidirectional communication between servers and apps. WebTransport offers several benefits over its predecessor, WebSockets, including faster connections, lower latency and the ability to handle both reliable and ordered data streams as well as unordered ones (such as UDP). It can handle multiple streams in the same connection without head-of-line blocking, allowing for more efficient communication in complex applications. Overall, WebTransport is suitable for a wide range of use cases, including real-time web apps, streaming media and Internet of Things (IoT) data communications. Even though WebTransport is still in the early stages — support across browsers is gradually maturing, with popular libraries such as socket.io adding support for WebTransport — our teams are currently assessing its potential for real-time IoT apps.

42. Zarf

Assess

Zarf is a declarative package manager for offline and semi-connected Kubernetes environments. With Zarf, you can build and configure applications while connected to the internet; once created, you can package and ship to a disconnected environment for deployment. As a standalone tool, Zarf packs several useful features, including automatic Software Bill of Materials (SBOM) generation, built-in Docker registry, Gitea and K9s dashboards to manage clusters from the terminal. Air-gap software delivery for cloud-native applications has its challenges; Zarf addresses most of them.

43. ZITADEL

Assess

ZITADEL is an open-source identity and user management tool, and an alternative to Keycloak. It's lightweight (written in Golang), has flexible deployment options and is easy to configure and manage. It's also multi-tenant, offers comprehensive features for building secure and scalable authentication systems, particularly for B2B applications, and has built-in security features like multi-factor authentication and audit trails. By using ZITADEL, developers can reduce development time, enhance application security and achieve scalability for growing user bases. If you're looking for a user-friendly, secure and open-source tool for user management, ZITADEL is a strong contender.

Tools

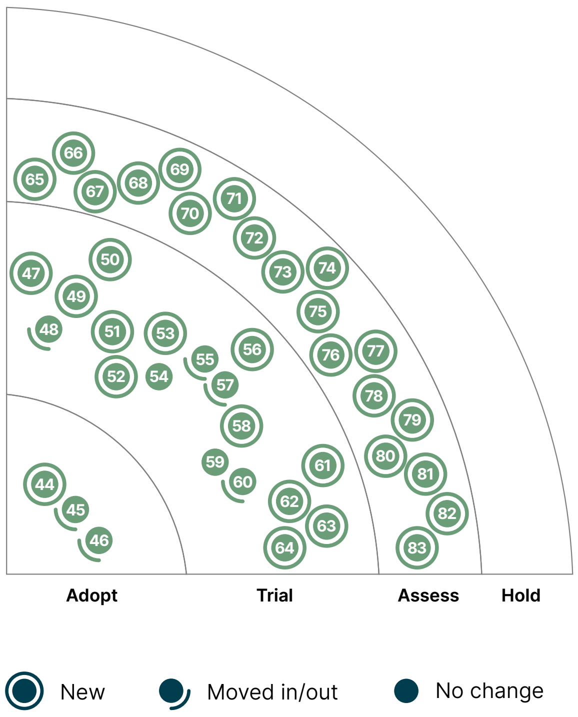
Adopt
44. Conan
45. Kaniko
46. Karpenter

Trial
47. 42Crunch API Conformance Scan
48. actions-runner-controller
49. Android Emulator Container
50. AWS CUDOS
51. aws-nuke
52. Bruno
53. Develocity
54. GitHub Copilot
55. Gradio
56. Gradle Version Catalog
57. Maestro
58. Microsoft SBOM tool
59. Open Policy Agent (OPA)
60. Philips's self-hosted GitHub runner
61. Pop
62. Renovate
63. Terrascan
64. Velero

Assess
65. aider
66. Akvorado
67. Baichuan 2
68. Cargo Lambda
69. Codium AI
70. Continue
71. Fern Docs
72. Granted
73. LinearB
74. LLaVA
75. Marimo
76. Mixtral
77. NeMo Guardrails
78. Ollama
79. OpenTofu
80. QAnything
81. System Initiative
82. Tetragon
83. Winglang

Hold

—



44. Conan

Adopt

Conan is an open-source dependency management tool for C/C++ applications. It provides an intuitive interface for defining, fetching and managing dependencies which makes it easy for developers to integrate third-party libraries into their projects. Conan works across all major operating systems and can target a variety of platforms, including servers and desktop, mobile and embedded devices. It can also be used for building and publishing C/C++ libraries and packages. The packages can be shared across teams via JFrog Artifactory servers. By leveraging prebuilt binaries, it significantly reduces build times, especially for hefty dependencies. Conan integrates with popular build systems like CMake and also has a Python SDK for extending the build system for tasks like signing. In our experience, Conan translates to improved build reproducibility across environments and faster development cycles. The resulting codebases are cleaner and easier to maintain, a major win for large-scale projects. If you're wrestling with dependency management in your C or C++ projects, Conan is a must-consider tool to boost your development efficiency.

45. Kaniko

Adopt

We added Kaniko to the Radar in October 2022, shortly after Kubernetes moved on from supporting Docker, highlighting at the time the trend away from Docker as the default to build container images within container-based pipelines. Since then, we've expanded our experience with Kaniko across different pipelines' tooling and configurations. Our teams appreciate its flexibility and performance which is why we're moving it to Adopt, highlighting Kaniko as the default tool in this space.

46. Karpenter

Adopt

One of the fundamental capabilities of Kubernetes is horizontal autoscaling: its ability to launch new pods when additional capacity is needed and shut them down when loads decrease. However, this only works if the nodes needed to host the pods already exist. Cluster Autoscaler can do some rudimentary cluster expansion triggered by pod failures, but it has limited flexibility; Karpenter, however, is a smarter, open-source Kubernetes Operator node autoscaler: it analyzes current workloads and pod scheduling constraints, selects an appropriate instance type and then starts or stops it as needed. Karpenter is an operator in the spirit of tools such as Crossplane that can provision cloud resources outside the cluster. Even though Karpenter was originally developed by AWS for EKS, it's becoming the default node autoprovisioner across all cloud Kubernetes service providers, and Azure recently started supporting Karpenter with AKS Karpenter Provider.

47. 42Crunch API Conformance Scan

Trial

42Crunch API Conformance Scan is a dynamic testing tool designed to identify discrepancies between your API's documented behavior and its actual implementation. This tool takes your API's spec definition in OpenAPI format, which outlines the expected functionalities and responses, and compares it to the API's actual behavior. By generating real traffic and interacting with live endpoints, the tool can identify any discrepancies between what the API promises and what it delivers. This translates into several benefits for development teams. For example, it catches inconsistencies early in development, saving time and preventing issues from reaching production. The tool also helps improve API quality and security by identifying potential vulnerabilities arising from deviations from the documented behavior. Overall, API Scan helps you assess the security posture of your APIs by identifying problems such as weak authentication protocols, insecure data handling practices and insufficient input validation. It provides detailed reports highlighting any issues found, along with recommendations for remediation.

48. actions-runner-controller

Trial

actions-runner-controller is a Kubernetes controller that operates self-hosted runners for GitHub Actions. Self-hosted runners are helpful in scenarios where the job that GitHub Actions runs needs to access resources that are either not accessible to GitHub cloud runners or have specific operating system and environmental requirements that are different from what GitHub provides. In such scenarios where the team uses Kubernetes clusters, actions-runner-controller orchestrates and scales these runners. Our teams like its ability to scale runners based on the number of workflows running in a given repository, organization, enterprise or Kubernetes cluster, as well as its ability to handle both Linux and Windows runners.

49. Android Emulator Container

Trial

Android Emulator Containers streamline Android app testing by eliminating the complexities arising from OS compatibility issues and system dependencies as well as from setting up emulators for multiple Android versions. Traditionally, this complexity led to extra effort or teams foregoing automated testing completely, which, in turn, resulted in slower development and testing cycles. Android Emulator Containers simplify this process, allowing seamless integration into CI pipelines for automated testing. Our teams utilize these containers primarily for instrumented tests, which are automatically executed with each commit to provide instantaneous feedback to developers. Additionally, we leverage Android Emulator Containers for running nightly end-to-end tests as well.

50. AWS CUDOS

Trial

Our advice has always been to monitor costs as a fitness function. Cloud providers offer a variety of services to monitor cloud spend such as AWS Cost Explorer or Google Cloud FinOps Hub. In the AWS ecosystem, our teams use the CUDOS (Cost and Usage Dashboards Operations Solution) dashboards to monitor AWS Marketplace spend segregated by the business departments or legal entities in a large parent organization. This dashboard provides comprehensive cost and usage details, with resource-level granularity that helps optimize costs, track usage goals and achieve operational excellence.

51. aws-nuke

Trial

aws-nuke is an open-source tool that tackles the common challenge of unused resources accumulating in development and sandbox AWS accounts that can lead to cost inefficiencies. The tool identifies and removes all deletable resources within an AWS account or region with the exception of default or AWS-managed resources, essentially resetting the environment to a Day One state. It also offers customizable exclusion policies to ensure critical resources remain protected. We've used this tool for the default use case of cost optimization as well as for disaster recovery (DR) contexts with good results. By automating cleanup in development and sandbox environments, aws-nuke helps minimize unnecessary resource expenditure. It also facilitates efficient teardown of temporary DR infrastructure after drills or exercises. Although stable, aws-nuke is a very destructive tool and is not intended to be used in production environments. Always perform a dry run to confirm that essential resources won't be deleted.

52. Bruno

Trial

Bruno is an open-source desktop alternative to Postman and Insomnia for API testing, development and debugging. It stores your collections locally on the filesystem so you can use Git or a version control of your choice to collaborate. Several Thoughtworks teams are using Bruno and like its simple offline-only design.

53. Develocity

Trial

Develocity (previously Gradle Enterprise) addresses the pain point of lengthy build and test cycles in large-scale software projects. It employs performance improvements such as build caching and predictive test selection to shorten developer feedback loops in both local and CI/CD environments. Our platform teams have found it useful for speeding up builds and tests, analyzing commands to determine what part of the workflow still needs to be optimized, identifying and troubleshooting flaky tests and performing analysis on the hardware used to run them.

54. GitHub Copilot

Trial

While the AI coding assistance market is getting busier and busier, GitHub Copilot remains our default choice and is used by many of our teams. Since we last wrote about GitHub Copilot, the most interesting improvements came in the chat feature. For instance, it's no longer necessary to clutter the code with comments as prompts; instead, an inline chat helps you prompt without writing a comment. The inline chat can also change code, not just write new lines. You can now also significantly expand the context of the chat when asking questions about your code by using the @workspace tag. This allows you to ask questions about the entire codebase, not just the open files. You can expand this context even further with the Copilot Enterprise version, which pulls in context from all repositories you host on GitHub. Finally, GitHub has started routing some chat requests to a more powerful GPT-4–based model, and availability of the chat in the popular Jetbrains IDEs is imminent (although still in private beta at the time of writing). These releases show that the pace of improvements in the space has not slowed down. If you tried a coding assistant last year and dismissed it, we recommend that you keep monitoring the features being released and give it another try.

55. Gradio

Trial

Gradio is an open-source Python library that facilitates the creation of interactive web-based interfaces for machine learning (ML) models. A graphical user interface on top of ML models provides a better understanding of the inputs, constraints and outputs by nontechnical audiences. Gradio has gained a lot of traction in the generative AI space, as it is one of the tools that makes generative models so accessible to experiment with. Usually, we put technologies into the Trial ring when we've seen them used in production at least once. Gradio's purpose and strength are experimentation and prototyping, and we've used it for that purpose many times. Recently, one of our teams even used it to help a client with live demonstrations at big events. We're very happy with Gradio's capabilities for those use cases, and therefore move it into the Trial ring.

56. Gradle Version Catalog

Trial

Gradle version catalog is a useful feature of the Gradle build tool that allows you to manage dependencies centrally in the build file. Our teams have found it especially useful with Android multi-module projects. Instead of hardcoding dependency names and versions in individual build files and managing upgrades, you can create a central version catalog of these dependencies and then reference it in a type-safe way with Android Studio assistance.

57. Maestro

Trial

Maestro is extremely useful when testing complex flows in mobile applications. Our teams have found it easy to learn, easy to understand and easy to integrate into our development workflow. Maestro supports a range of mobile platforms including iOS, Android, React Native and Flutter apps. Its declarative YAML syntax simplifies the automation of complex mobile UI interactions. Based on the tool's evolution, marked by enhanced features like comprehensive iOS support and the introduction of tools like Maestro Studio and Maestro Cloud, we encourage anyone seeking to optimize their mobile application testing processes to give it a try.

58. Microsoft SBOM tool

Trial

Microsoft SBOM tool is an open-source tool to generate SPDX-compatible Software Bill of Materials (SBOM). We have blipped about the need for SBOM previously, and this tool makes it easier to get started. SBOM tool supports a variety of popular package managers (including npm, pip and Gradle), making it compatible with a wide range of projects. It's very easy to use and can be integrated into existing development workflows, including integration with CI/CD pipelines. By leveraging SBOM generated with this tool, developers gain multiple advantages. Improved software security is a key benefit, as a clear view of components allows for easier vulnerability identification and risk management. License compliance is also enhanced, as developers can ensure adherence to all relevant agreements. Furthermore, SBOM promotes transparency within the software supply chain, aiding dependency tracking and mitigating potential risks. If you're looking to streamline SBOM generation, improve software security and gain control over your software supply chain, you should give Microsoft SBOM tool a try.

59. Open Policy Agent (OPA)

Trial

Open Policy Agent (OPA) is a uniform framework and language for declaring, enforcing and controlling policies. For our teams, it has become a favored way of defining policies for distributed systems, particularly where we need to implement compliance at the point of change. OPA allows teams to implement various platform engineering patterns, such as controlling what is deployed to Kubernetes clusters, enforcing access control across services in a service mesh and implementing fine-grained security policy as code for accessing application resources. While there is some complexity associated with OPA implementations, it has proven to be a highly valuable tool for ensuring compliance in a DevOps culture. We're also continuing to keep an eye on the extension and maturity of OPA beyond operational systems to (big) data-centric solutions.

60. Philips's self-hosted GitHub runner

Trial

While GitHub Actions runners cover a wide range of the most common run times and are quickest to start with, teams sometimes need to manage self-hosted runners, such as when organizational policy only allows deployments to a privately hosted infrastructure from within the organization's own security perimeter. In such cases, teams can use Philips's self-hosted GitHub runner, a Terraform module that spins up custom runners on AWS EC2 spot instances. The module also creates a set of Lambdas that handles lifecycle management (scaling up and down) for these runners. In our experience, this tool greatly simplifies the provisioning and management of self-hosted GitHub Actions runners. An alternative for teams that use Kubernetes is actions-runner-controller.

61. Pop

Trial

Pair programming continues to be an essential technique for us, because it helps improve code quality and spread knowledge within a team. Although it's best done in person, our distributed teams have explored tools to make remote pairing as pleasant and effective as possible, such as Tuple, Visual Studio Live Share, Code With Me and general-purpose chat and conferencing tools. The latest tool in the space that's caught our attention is Pop (formerly Screen). Coming from the founders of Screenhero, it supports multi-person screen sharing, annotations and high-quality audio/video. Some of our teams have used it extensively for pair programming and remote working sessions and report positively on their experience.

62. Renovate

Trial

Automatically monitoring and updating dependencies as part of the software build process has become standard practice across the industry. It takes the guesswork out of staying current with security updates to open-source packages as they're released. For many years, Dependabot has been the standard tool for this practice, but Renovate has become the preferred tool for many of our teams. They find that Renovate is more suitable to the modern software development environment where a deployable system relies not just on code and libraries but encompasses run-time tools, infrastructure and third-party services. Renovate covers dependencies on these ancillary artifacts in addition to code. Our teams also found that Renovate offers more flexibility through configuration and customization options. Although Dependabot remains a safe default choice and is conveniently integrated with GitHub, we'd recommend evaluating Renovate to see if it can further reduce the manual burden on developers to keep their application ecosystems safe and secure.

63. Terrascan

Trial

Terrascan is a static code analyzer for infrastructure as code (IaC) designed to detect security vulnerabilities and compliance issues before provisioning cloud-native infrastructure. It supports scanning for Terraform, Kubernetes (JSON/YAML), Helm, AWS CloudFormation, Azure Resource Manager, Dockerfiles and GitHub. The default policy pack covers all the major cloud providers, GitHub, Docker and Kubernetes. Our teams use Terrascan locally as a pre-commit hook and integrate it into CI pipelines to detect IaC vulnerabilities and violations.

64. Velero

Trial

Velero is an open-source tool for backing up and restoring Kubernetes resources and persistent volumes. It simplifies disaster recovery and cluster migrations by enabling on-demand and scheduled backups. Velero also allows finer-grained controls over which resources get backed up as well as over the backup/restore workflow (with hooks). Our teams appreciate its ease of use and its reliance on Kubernetes APIs instead of lower-level layers like etcd.

65. aider

Assess

aider is an open-source AI coding assistant. Like many open-source tools in this space, aider doesn't have direct IDE integration but is started as a CLI in the terminal. aider is interesting because it provides a chat interface with write access to the codebase across multiple files, whereas many of the coding assistant products today either only read the code or can change only one file at a time. This allows aider to help you implement concepts that stretch over multiple files (e.g., "add locators to my HTML and also use those in my functional test") and to create new files and folder structures in the codebase (e.g., "create a new component similar to the one in folder X"). As aider is open source and not a hosted product, you have to bring your own OpenAI or Azure OpenAI API key to use it. On the one hand, this can be great for occasional use because you only have to pay per use. On the other hand, aider does seem to be quite chatty in its interactions with the AI API, so keep an eye on request costs and rate limits when using it.

66. Akvorado

Assess

Akvorado is an open-source network monitoring and analysis tool. It captures the network flows, Netflow/IPFIX and sFlow, enriches them with interface names and geo information and then saves the updated flows in ClickHouse for future analysis. Although OpenTelemetry is gaining adoption for observing application-level traffic, we often come across challenges in the network layer that can be difficult to spot and troubleshoot. Tools like Akvorado are quite handy in such situations as they help you analyze the network flows across various devices in the network topology.

67. Baichuan 2

Assess

Baichuan 2 is part of a new generation of open-source large language models. It was trained on a high-quality corpus with 2.6 trillion tokens, achieving quite good performance for its size on Chinese, English and multi-language benchmarks. Baichuan has been trained on several domain-specific corpora, including healthcare and law data sets, which is why we prefer using it in these and related fields.

68. Cargo Lambda

Assess

The efficiency and performance of Rust make it a good fit for serverless computing. Another advantage is that Rust functions don't require a run time, which results in fast startup times. However, the developer experience for writing the functions in Rust wasn't great. That changed with Cargo Lambda. As a cargo subcommand, it integrates with the typical Rust workflow and allows you to run and test your AWS Lambda functions on the developer machine without needing Docker, VMs or other tools. Using a Zig toolchain, Cargo Lambda can cross-compile the functions on several operating systems for the Linux sandboxes used by AWS Lambda, and both Arm and Intel are supported as target architectures.

69. Codium AI

Assess

In the busy emerging space of AI coding assistants, some products, instead of competing with the strong incumbents, take a more focused approach. Codium AI is focused on test generation with AI. It works for all languages but has advanced support for common stacks, such as JavaScript and Python. We particularly like that the tool, rather than taking developers straight to the test code, offers a list of scenario descriptions in natural language for review. This makes it easier for developers to reason about the scenarios and decide which ones to turn into test code. To further improve the test generation for a particular codebase and use case, users can provide example tests and general instructions to enhance the information used by the AI to generate the tests.

70. Continue

Assess

Continue is an open-source autopilot for VS Code and JetBrains IDEs. We quite like it because it eliminates the pain of copying/pasting from a chat-based interface to large language models (LLMs) with a direct integration in the IDE. It supports several commercial and open-source models and makes it easy to try different LLM providers, including self-hosted LLMs. You can even run Continue without an internet connection.

71. Fern Docs

Assess

One hallmark of widely used REST APIs is that their contracts are thoroughly documented. Developers are more likely to adopt and use APIs whose behavior and syntax are described accurately in a structured, organized way. Keeping this documentation up to date as the contract evolves can be time-consuming and is a task that is easily overlooked. Fern Docs helps with this by reducing the toil involved in writing and maintaining API documentation. Fern automatically generates a website with attractive, usable documentation from a specification file that can be versioned alongside the API code. While our initial impressions of this product are positive, Fern does require you to maintain API information in a proprietary configuration file. While it provides a way to convert OpenAPI specs into its own configuration format, we'd prefer a tool that generates docs directly from annotated source code.

72. Granted

Assess

Given how common multi-account strategy is in organizations' AWS environments, engineers frequently have to switch between multiple accounts within a short period of time. Granted, a command-line tool that simplifies the opening of multiple accounts in the browser simultaneously, streamlines account switching. It leverages each browser's native features to isolate multiple identities, Firefox's Multi-Account Containers and Chromium's Profiles. If a specific service (such as S3) is specified as an argument, Granted will open the service's landing page. Granted currently only supports AWS. Notably, it stores AWS SSO's temporary credentials safely in the keychain rather than as plain text on the disk.

73. LinearB

Assess

LinearB is a platform designed to empower engineering leaders with data-driven insights for continuous improvement. It tackles three key areas: benchmarking, workflow automation and investment. Our experience with LinearB's metrics tooling highlights its potential to support a culture of continuous improvement. One of our teams leveraged the platform to track engineering metrics, identify and discuss improvement opportunities and define actionable steps based on data, leading to measurable progress. We're happy to see that this aligns with LinearB's core value proposition: benchmark, automate and improve. LinearB integrates with GitHub, GitLab, Bitbucket and Jira. It offers a comprehensive suite of preconfigured engineering metrics, with a strong focus on DORA metrics (deployment frequency, lead time, change failure rate and time to restore). As strong advocates of the four key metrics as defined by the DORA research, we appreciate LinearB's emphasis on measuring what truly matters for software delivery performance. Historically, gathering DORA-specific metrics has been a challenge. Teams have resorted to complex CD pipeline instrumentation, custom-built dashboards or rely on manual processes. Although our experience is limited to one team, LinearB seems to be a compelling alternative for gathering and tracking engineering metrics as well as fostering a data-driven approach to continuous improvement.

74. LLaVA

Assess

LLaVA (Large Language and Vision Assistant) is an open-source, large multimodal model that connects a vision encoder and LLM for general-purpose visual and language understanding. LLaVA's strong proficiency in instruction-following positions it as a highly competitive contender among multimodal AI models. The latest version, LLaVA-NeXT, allows for improved question answering. Among the open-source models for language and vision assistance, LLaVA is a promising option when compared to GPT-4 Vision. Our teams have been experimenting with it for visual question answering.

75. Marimo

Assess

Marimo offers a fresh take on Python notebooks by prioritizing reproducibility and interactivity. It addresses challenges with hidden state in traditional notebooks (like Jupyter) which can lead to unexpected behavior and hinder reproducibility. It does that by storing notebooks as plain Python files with no hidden state and using a deterministic execution order based on dependencies (when a variable changes, all affected cells are automatically run). Marimo also comes with interactive UI elements that similarly propagate value changes to cells that depend on them. As it can be deployed as a web app, it's also a useful tool for demos and prototyping purposes. Although we're excited for the potential of Marimo, in particular in terms of reproducibility for data exploration and analysis purposes, we continue to caution against productionizing notebooks.

76. Mixtral

Assess

Mixtral is part of the family of open-weight large language models Mistral released, that utilizes the sparse Mixture of Experts architecture. The family of models are available both in raw pretrained and fine-tuned forms in 7B and 8×7B parameter sizes. Its sizes, open-weight nature, performance in benchmarks and context length of 32,000 tokens make it a compelling option for self-hosted LLMs. Note that these open-weight models are not tuned for safety out of the box, and users need to refine moderation based on their own use cases. We have experience with this family of models in developing Aalap, a fine-tuned Mistral 7B model trained on data related to specific Indian legal tasks, which has performed reasonably well on an affordable cost basis.

77. NeMo Guardrails

Assess

NeMo Guardrails is an easy-to-use open-source toolkit from NVIDIA that empowers developers to implement guardrails for large language models (LLMs) used in conversational applications. Although LLMs hold immense potential in building interactive experiences, their inherent limitations around factual accuracy, bias and potential misuse necessitate safeguards. Guardrails offer a promising approach to ensure responsible and trustworthy LLMs. Although you have a choice when it comes to LLM guardrails, our teams have found NeMo Guardrails particularly useful because it supports programmable rules and run-time integration and can be applied to existing LLM applications without extensive code modifications.

78. Ollama

Assess

Ollama is an open-source tool for running and managing large language models (LLMs) on your local machine. Previously, we talked about the benefits of self-hosted LLMs, and we're pleased to see the ecosystem mature with tools like Ollama. It supports several popular models — including LLaMA-2, CodeLLaMA, Falcon and Mistral — that you can download and run locally. Once downloaded, you can use the CLI, API or SDK to interact with the model and execute your tasks. We're evaluating Ollama and are seeing early success as it improves the developer experience in working with LLMs locally.

79. OpenTofu

Assess

OpenTofu is a fork of Terraform made in response to a recent ambiguous license change by HashiCorp. It's open source and has been accepted by the Linux Foundation. It's backed by several organizations, including third-party vendors. The current version is compatible with the last open-source version of Terraform. Version 1.7 adds client-side encryption. The future of the OpenTofu project is unclear in terms of how closely it will support compatibility with future versions of Terraform. There are also questions around the long-term support by its current backers. We recommend keeping an eye on the project but remain cautious around usage, except for teams that have the capability to manage risks that may include being able to contribute to the codebase.

80. QAnything

Assess

Large language models (LLMs) and retrieval-augmented generation (RAG) techniques have greatly improved our ability to synthesize and extract information. We're seeing emerging tools taking advantage of this, and QAnything is one of them. QAnything is a knowledge management engine with a question-and-answer interface that can summarize and extract information from a wide range of file formats, including PDF, DOCX, PPTX, XLSX and MD files, among others. For data security concerns, QAnything also supports offline installation. Some of our teams use QAnything to build their team knowledge base. In GenAI scenarios with more industry depth (such as generating abstracts for investment reports), we also try to use this tool for proofs of concept before building real products and showing the potential of LLMs and RAG.

81. System Initiative

Assess

Little has emerged in recent years to challenge the dominance of Terraform as an infrastructure coding tool. Although alternatives such as Pulumi, CDK and, more recently, Wing have emerged, Terraform's modular, declarative paradigm has proven to be the most enduring. Indeed, all of these approaches share the common goal of modular code creating monolithic infrastructure. System Initiative is a new, experimental tool that represents a radical new direction for DevOps work. One way to view System Initiative is as a digital twin for your infrastructure. Interactive changes to the System Initiative state result in corresponding change sets that can be applied to the infrastructure itself. Likewise, changes to the infrastructure are reflected in the System Initiative state. One of the great advantages of this approach is the collaborative environment it creates for things like application deployment and observability. Engineers interact with System Initiative through a user interface that has a graphical representation of the entire environment. In addition to managing the cloud infrastructure, you can also use the tool to manage containers, scripts, tools and more. Although we're generally skeptical of these kinds of GUI tools, System Initiative can be extended to handle new assets or enforce policy via TypeScript code. 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 free and open source under an Apache 2.0 license and is currently in open beta. The maintainers themselves do not recommend the tool for production use yet, but we think it's worth checking out in its current state to experience a completely different approach to DevOps tooling.

82. Tetragon

Assess

Tetragon is an open-source eBPF-based security observability and run-time enforcement tool. We mentioned Falco for detecting security threats a while back in the Radar. Tetragon goes beyond threat detection by leveraging eBPF to enforce security policies at run time in the Linux kernel. You can use Tetragon either as a standalone tool on bare metal or inside the Kubernetes environment.

83. Winglang

Assess

We're seeing a lot of movement in the infrastructure-as-code (IaC) space with tools like Winglang emerging. Winglang takes a different approach to defining infrastructure and run-time behavior. It provides high-level abstractions over platform specifics provided by tools such as CloudFormation, Terraform, Pulumi and Kubernetes. With Winglang, you write code that runs at compile time to generate infrastructure configuration and then code that executes at run time for application behavior. It provides a simulation mode to run locally and has an integrated test framework. We're keeping an eye on this interesting tool; it's a potential preview of the future direction of IaC.

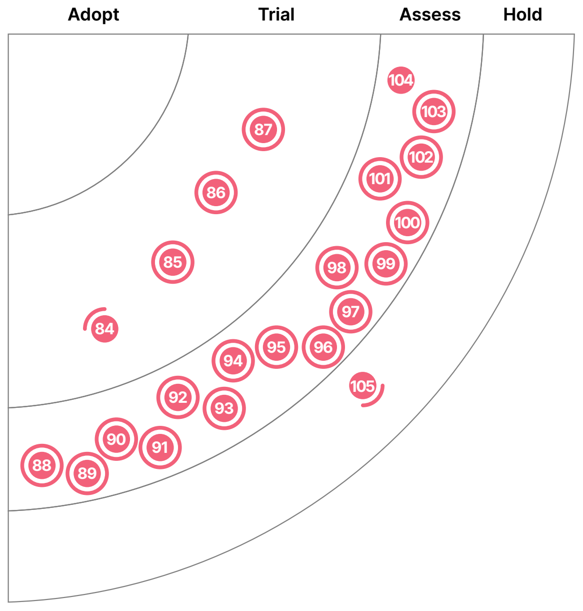
Languages and Frameworks

Adopt
—

Trial
84. Astro
85. DataComPy
86. Pinia
87. Ray

Assess
88. Android Adaptability
89. Concrete ML
90. Crabviz
91. Crux
92. Databricks Asset Bundles
93. Electric
94. LiteLLM
95. LLaMA-Factory
96. MLX
97. Mojo
98. Otter
99. Pkl
100. Rust for UI
101. vLLM
102. Voyager
103. WGPU
104. Zig

Hold
105. LangChain



◉ New ◔ Moved in/out ● No change

84. Astro

Trial

The Astro framework is gaining more popularity in the community. One of our teams has used Astro to build content-driven websites like blogs and marketing websites. Astro is a multi-page application framework that renders HTML on the server and minimizes the amount of JavaScript sent over the wire. We like that Astro supports — when appropriate — select active components written in the front-end JavaScript framework of your choice even though it encourages sending only HTML. It does this through its island architecture. Islands are regions of interactivity within a single page where the necessary JavaScript is downloaded only when needed. In this way, most areas of the site are converted to fast, static HTML, and the JavaScript parts are optimized for parallel loading. Our team likes both its page rendering performance as well as its build speed. The Astro component syntax is a simple extension of HTML, and the learning curve is quite gentle.

85. DataComPy

Trial

Comparing DataFrames is a common task in data engineering, often done to compare the output of two data transformation approaches to make sure no meaningful deviations or inconsistencies have occurred. DataComPy is a Python library that facilitates the comparison of two DataFrames in pandas, Spark and more. The library goes beyond basic equality checks by providing detailed insights into discrepancies at both row and column levels. DataComPy also has the ability to specify absolute or relative tolerance for comparison of numeric columns as well as known differences it need not highlight in its report. Some of our teams use it as part of their smoke testing suite; they find it efficient when comparing large and wide DataFrames and consider its reports easy to understand and act upon.

86. Pinia

Trial

Pinia is a store library and state management framework for Vue.js. It uses declarative syntax and offers its own state management API. Compared to Vuex, Pinia provides a simpler API with less ceremony, offers Composition-style APIs and, most importantly, has solid type inference support when used with TypeScript. Pinia is endorsed by the Vue.js team as a credible alternative to Vuex and is currently the official state management library for Vue.js. Our teams are leveraging Pinia for its simplicity and ease of implementation.

87. Ray

Trial

Today's machine learning (ML) workloads are increasingly compute-intensive. As convenient as they are, single-node development environments such as your laptop cannot scale to meet these demands. Ray is a unified framework for scaling AI and Python code from laptop to cluster. Ray is essentially a well-encapsulated distributed computing framework with a series of AI libraries to simplify ML work. By integrating with other frameworks (e.g., PyTorch and TensorFlow), it can be used to build large-scale ML platforms. Companies like OpenAI and Bytedance use Ray heavily for model training and inference. We also use its AI libraries to help with distributed training and hyperparameter tuning on our projects. We recommend you try Ray when building scalable ML projects.

88. Android Adaptability

Assess

Some mobile applications and games can be so demanding they cause thermal throttling within a few minutes. In this state, CPU and GPU frequency are reduced to help cool the device, but it also results in reduced frame rates in games. When the thermal situation improves, the frame rates increase again and the cycle repeats, leading to the software feeling janky. Android Adaptability, a new set of libraries, allows application developers to respond to changing performance and thermal situations. The Android Dynamic Performance Framework (ADPF) includes the Thermal API to provide information about the thermal state and the Hint API to help Android choose the optimal CPU operating point and core placement. Teams using Unity will find the Unity Adaptive Performance package helpful, as it works with both APIs.

89. Concrete ML

Assess

Previously, we blipped the Homomorphic Encryption technique that allows computations to be performed directly on encrypted data. Concrete ML is one such open-source tool that allows for privacy-preserving machine learning. Built on top of Concrete, it simplifies the use of fully homomorphic encryption (FHE) for data scientists to help them automatically turn machine learning models into their homomorphic equivalent. Concrete-ML's built-in models have APIs that are almost identical to their scikit-learn counterparts. You can also convert PyTorch networks to FHE with Concrete-ML's conversion APIs. Note, however, that FHE with Concrete-ML could be slow without tuned hardware.

90. Crabviz

Assess

Crabviz is a Visual Studio Code plug-in to create call graphs. The graphs are interactive, which is essential when working with even moderately large codebases such as a microservice. They show types, methods, functions and interfaces grouped by file and also display function calling relationships and interface implementation relationships. Because Crabviz is based on the Language Server Protocol, it supports any number of languages, as long as the corresponding language server is installed. This means, though, that Crabviz is limited to static code analysis, which might not be sufficient for some use cases. The plug-in is written in Rust and is available on the Visual Studio Code Marketplace.

91. Crux

Assess

Crux is an open-source cross-platform app development framework written in Rust. Inspired by the Elm architecture, Crux organizes business logic code at the core and UI layer in native frameworks like SwiftUI, Jetpack Compose, React/Vue or WebAssembly-based frameworks (like Yew). With Crux, you can write side effects–free behavior code in Rust and share it across iOS, Android and the web.

92. Databricks Asset Bundles

Assess

The recent public preview release of Databricks Asset Bundles (DABs), included with Databricks CLI version 0.205 and above, is becoming the officially recommended way to package Databricks assets for source control, testing and deployment. It has started to replace dbx among our teams. DABs supports packaging the configuration of workflows, jobs and tasks, as well as the code to be executed in those tasks, as a bundle that can be deployed to multiple environments. It comes with templates for common types of assets and supports custom templates. While DABs includes templates for notebooks and supports deploying them to production, we continue to recommend against productionizing notebooks and instead encourage intentionally writing production code with the engineering practices that support the maintainability, resiliency and scalability needs of such workloads.

93. Electric

Assess

Electric is a local-first sync framework for mobile and web applications. Local-first is a development paradigm where your application code talks directly to an embedded local database and data syncs in the background via an active-active database replication to the central database. With Electric, you have SQLite as the local embedded option and PostgreSQL for the central store. Although local-first greatly improves user experience, it is not without challenges, and the inventors of CRDT have worked on the Electric framework to ease the pain.

94. LiteLLM

Assess

LiteLLM is a library for seamless integration with various large language model (LLM) providers' APIs that standardizes interactions through an OpenAI API format. It supports an extensive array of providers and models and offers a unified interface for completion, embedding and image generation functionalities. LiteLLM simplifies integration by translating inputs to match each provider's specific endpoint requirements. This is particularly valuable in the current landscape, where a lack of standardized API specifications for LLM providers complicates the inclusion of multiple LLMs in projects. Our teams have leveraged LiteLLM to swap underlying models in LLM applications, addressing a significant integration challenge. However, it's crucial to acknowledge that model responses to identical prompts vary, indicating that a consistent invocation method alone may not fully optimize completion performance. Note that LiteLLM has several other features, such as proxy server, that are not in the purview of this blip.

95. LLaMA-Factory

Assess

We continue to caution against rushing to fine-tune large language models (LLMs) unless it's absolutely critical — it comes with a significant overhead in terms of costs and expertise. However, we think LLaMA-Factory can be useful when fine-tuning is needed. It's an open-source, easy-to-use fine-tuning and training framework for LLMs. With support for LLaMA, BLOOM, Mistral, Baichuan, Qwen and ChatGLM, it makes a complex concept like fine-tuning relatively accessible. Our teams used LLaMA-Factory's LoRA tuning for a LLaMA 7B model successfully, so, if you have a need for fine-tuning, this framework is worth assessing.

96. MLX

Assess

MLX is an open-source array framework designed for efficient and flexible machine learning on Apple silicon. It lets data scientists and machine learning (ML) engineers access the integrated GPU, allowing them to choose the hardware best suited for their needs. The design of MLX is inspired by frameworks like NumPy, PyTorch and Jax to name a few. One of the key differentiators is MLX's unified memory model, which eliminates the overhead of data transfers between the CPU and GPU, resulting in faster execution. This feature makes running the models on devices such as iPhones plausible, opening a huge opportunity for on-device AI applications. Although niche, this framework is worth pursuing for the ML developer community.

97. Mojo

Assess

Mojo is a new AI-first programming language. It aims to bridge the gap between research and production by combining the Python syntax and ecosystem with systems programming and metaprogramming features. It's the first language to take advantage of the new MLIR compiler backend and packs cool features like zero-cost abstraction, auto tuning, eager destruction, tail call optimization and better single instruction, multiple data (SIMD) ergonomics. We like Mojo a lot and encourage you to give it a try. The Mojo SDK is currently available for Ubuntu and macOS operating systems.

98. Otter

Assess

Otter is a contention-free cache library in Go. Although Go has several such libraries, we want to highlight Otter for two reasons: its excellent throughput and its clever implementation of the S3-FIFO algorithm for good cache hit ratio. Otter also supports generics, so you can use any comparable types as keys and any types as values.

99. Pkl

Assess

Pkl is a configuration language and tooling created for use internally by Apple and now open-sourced. The key feature of Pkl is its type and validation system, allowing configuration errors to be caught prior to deployment. It generates JSON, .plist, YAML and .properties files and has extensive IDE and language integration including code generation.

100. Rust for UI

Assess

The impact of Rust continues to grow, and many of the build and command-line tools we've covered recently are written in Rust. Now, we're seeing movement in using Rust for UI development as well. The majority of teams who prefer to use the same language for code running in the browser and on the server opt to use JavaScript or TypeScript. However, with WebAssembly you can use Rust in the browser, and this is becoming a little more common now. Frameworks like Leptos and sauron focus on web development, while Dioxus and several other frameworks support cross-platform desktop and mobile app development in addition to web development.

101. vLLM

Assess

vLLM is a high-throughput and memory-efficient inferencing and serving engine for large language models (LLMs) that's particularly effective thanks to its implementation of continuous batching for incoming requests. It supports several deployment options, including deployment of distributed tensor-parallel inference and serving with Ray run time, deployment in the cloud with SkyPilot and deployment with NVIDIA Triton, Docker and LangChain. Our teams have had good experience running dockerized vLLM workers in an on-prem virtual machine, integrating with OpenAI compatible API server — which, in turn, is leveraged by a range of applications, including IDE plugins for coding assistance and chatbots. Our teams leverage vLLM for running models such as CodeLlama 70B, CodeLlama 7B and Mixtral. Also notable is the engine's scaling capability: it only takes a couple of config changes to go from running a 7B to a 70B model. If you're looking to productionize LLMs, vLLM is worth exploring.

102. Voyager

Assess

Voyager is a navigation library built for Android's Jetpack Compose. It supports several navigation types, including Linear, BottomSheet, Tab and Nested, and its screen model integrates with popular frameworks like Koin and Hilt. When using Jetpack Compose in a multiplatform project, Voyager is a good choice to implement a common navigation pattern across all supported platforms. Development on Voyager has picked up again and the library reached version 1.0 in December 2023.

103. WGPU

Assess

wgpu is a graphics library for Rust based on the WebGPU API, notable for its capacity to handle general-purpose graphics and compute tasks on the GPU efficiently. wgpu aims to fill the gap left by the phasing out of older graphics standards such as OpenGL and WebGL. It introduces a modern approach to graphics development that spans both native applications and web-based projects. Its integration with WebAssembly further enables graphics and compute applications to run in the browser. wgpu represents a step forward in making advanced graphics programming more accessible to web developers with a range of applications, from gaming to creating sophisticated web animations, positioning wgpu as an exciting technology to assess.

104. Zig

Assess

Zig is a new language that shares many attributes with C but with stronger typing, easier memory allocation and support for namespacing, among a host of other features. Zig's aim is to provide a very simple language with straightforward compilation that minimizes side-effects and delivers predictable, easy-to-trace execution. Zig also provides simplified access to LLVM's cross-compilation capability. Some of our developers have found this feature so valuable they're using Zig as a cross-compiler, even though they're not writing Zig code. We see teams in the industry using Zig to help build C/C++ toolchains. Zig is a novel language and worth looking into for applications where C is being considered or already in use.

105. LangChain

Hold

We mentioned some of the emerging criticisms about LangChain in the previous Radar. Since then, we've become even more wary of it. While the framework offers a powerful set of features for building applications with large language models (LLMs), we've found it to be hard to use and overcomplicated. LangChain gained early popularity and attention in the space, which turned it into a default for many developers. However, as LangChain is trying to evolve and keep up with the fast pace of change, it has become harder and harder to navigate those changes of concepts and patterns as a developer. We've also found the API design to be inconsistent and verbose. As such, it often obscures what is actually going on under the hood, making it hard for developers to understand and control how LLMs and the various patterns around them actually work. We're moving LangChain to the Hold ring to reflect this. In many of our use cases, we've found that an implementation with minimum use of specialized frameworks is sufficient. Depending on your use case, you may also want to consider other frameworks such as Semantic Kernel, Haystack or LiteLLM.

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

Thoughtworks is a global technology consultancy that integrates strategy, design and engineering to drive digital innovation. We are 10,500+ people strong across 48 offices in 19 countries. Over the last 30 years, we've delivered extraordinary impact together with our clients by helping them solve complex business problems with technology as the differentiator.

/thoughtworks

Strategy. Design. Engineering.