技术雷达

针对当今科技领域发展的
前沿指南

Volume 31
2024年10月

/thoughtworks

Strategy. Design. Engineering.

关于技术雷达

Thoughtworker 酷爱技术。我们致力于建造技术，研究技术，测试技术，开源技术，书写技术，并不断改进技术。支持卓越软件并掀起 IT 革命是我们的使命，Thoughtworks 技术雷达就是为了完成这一使命。它由 Thoughtworks 中一群资深技术领导组成的技术顾问委员会，通过定期讨论 Thoughtworks 的全球技术战略以及对行业有重大影响的技术趋势而创建。

技术雷达以独特的形式记录技术顾问委员会的讨论结果，从首席技术官到开发人员，雷达将会为各路利益相关方提供价值。这些内容只是简要的总结。

我们建议您探索雷达中提到的内容以了解更多细节。技术雷达的本质是图形性质，把各种技术项目归类为技术、工具、平台和语言和框架。如果技术可以被归类到多个象限，我们选择看起来最合适的一个。我们还进一步将这些技术分为四个环以反映我们目前对其的态度。

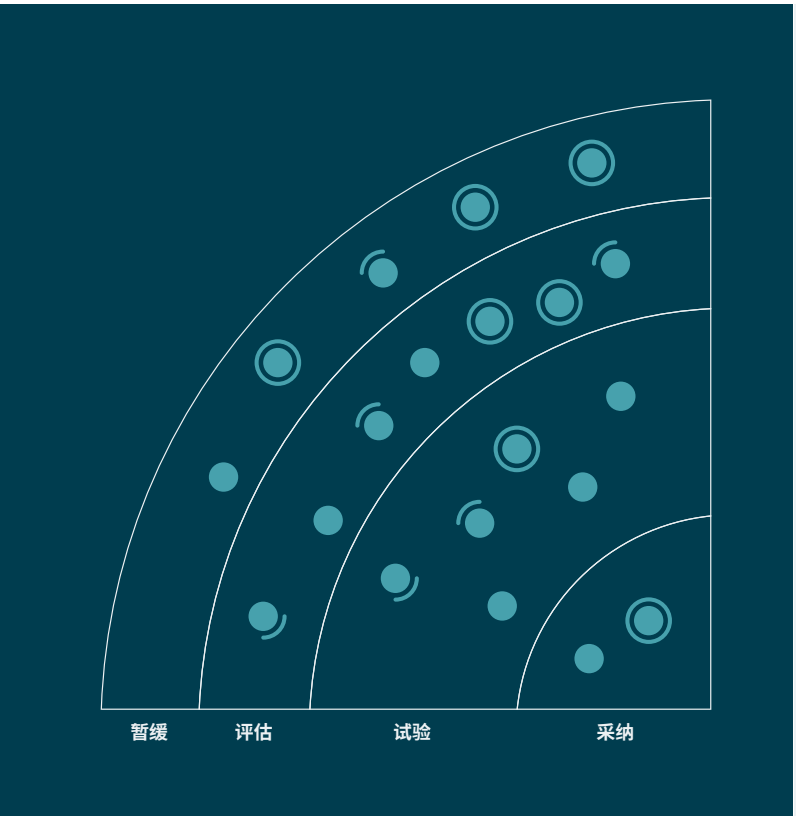
想要了解更多技术雷达相关信息，请点击：thoughtworks.com/cn/radar/faq

© Thoughtworks, Inc. All Rights Reserved.

雷达一览

技术雷达持续追踪有趣的技术是如何发展的，我们将其称之为条目。在技术雷达中，我们使用象限和环对其进行分类，不同象限代表不同种类的技术，而圆环则代表我们对它作出的成熟度评估。

软件领域瞬息万变，我们追踪的技术条目也如此，因此您会发现它们在雷达中的位置也会改变。



采纳：我们强烈主张业界采用这些技术。我们会在适当时候将其用于我们的项目。

试验：值得追求。重要的是理解如何建立这种能力，企业应该在风险可控的项目中尝试此技术。

评估：为了确认它将如何影响你所在的企业，值得作一番探究。

暂缓：谨慎推行。

新的　挪进 / 挪出　没有变化

技术雷达是具有前瞻性的。为了给新的技术条目腾出空间，我们挪出了近期没有发生太多变化的技术条目，但略去某项技术并不表示我们不再关心它。

© Thoughtworks, Inc. All Rights Reserved.

贡献者

技术顾问委员会（TAB）由 Thoughtworks 的 22 名高级技术专家组成。TAB 每年召开两次面对面会议，每两周召开一次视频会议。其主要职责是为 Thoughtworks 的首席技术官 Rachel Laycock 和名誉首席技术官 Rebecca Parsons 提供咨询建议。

作为一个综合型组织，TAB 能够审视影响 Thoughtworks 技术战略和技术人员的各种主题。本期技术雷达内容基于 2024 年 9 月技术委员会成员在纽约的面对面会议创建。


Rachel Laycock (CTO)


Martin Fowler (Chief Scientist)


Rebecca Parsons (CTO Emerita)


Bharani Subramaniam


Birgitta Böckeler


Camilla Falconi Crispim


Erik Dörnenburg


James Lewis


Ken Mugrage


Maya Ormaza


Mike Mason

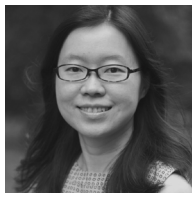

Neal Ford


Pawan Shah


Scott Shaw


Selvakumar Natesan


Shangqi Liu 刘尚奇


Sofia Tania


Thomas Squeo


Vanya Seth


Will Amaral

© Thoughtworks, Inc. All Rights Reserved.

本期主题

编码辅助反模式

毫不意外，生成式 AI 和大语言模型（LLM）成为了本期雷达中讨论的焦点话题，尤其是开发者们使用它们的新兴模式。伴随这些模式不可避免的也出现了对应的反模式——那些开发者应避免的情境化问题。我们在过度亢奋的 AI 领域中已经看到了一些反模式的出现，包括错误地认为 人类可以完全用 AI 取代结对编程伙伴 、过度依赖编码辅助建议、生成代码的质量问题以及代码库的快速膨胀。AI 往往通过蛮力而非抽象概念解决问题，比如使用大量嵌套条件语句而不是应用一个策略设计模式。代码质量问题尤其突显了开发者和架构师需要持续关注的领域，以确保他们不会在一堆“能用但很糟糕” 的代码中迷失。因此，团队成员应该 加强良好的工程实践 ，如单元测试、架构适应度函数以及其他经过验证的治理和检验技术，确保 AI 能够帮助你的工作，而不是通过复杂性加密你的代码库。

Rust 绝非“锈”铁

Rust 逐渐成为首选的系统编程语言。在每一次的雷达会议中，Rust 都反复出现在我们讨论的潜台词中；我们讨论的许多工具都是用 Rust 编写的。当替换旧的系统级实用工具时，它是首选语言，同时也是为了提高性能而重写生态系统某些部分的选择——基于 Rust 的工具最常见的描述是“极其快速”。例如，我们看到 Python 生态系统中有几种工具提供了基于 Rust 的替代方案，以支持明显更好的性能。语言设计者和社区成功创建了一个备受欢迎的核心 SDK、库和开发工具生态系统，同时提供了出色的执行速度，且比许多前辈语言的陷阱更少。我们团队中的许多人都是 Rust 的粉丝，而使用 Rust 的大多数开发者似乎也对其高度评价。

WASM 逐步崛起

WASM（WebAssembly）是一种用于基于栈的虚拟机的二进制指令格式，这听起来对大多数开发者来说太深奥和底层，直到人们看到其潜在的影响：能够在浏览器沙箱中运行复杂的应用程序。WASM 可以在现有的 JavaScript 虚拟机中运行，使得开发者过去只能在原生框架和扩展中实现的应用程序可以嵌入到浏览器中。现在，四大主要浏览器（Chrome、Firefox、Safari 和 Edge）都支持 WASM 1.0，这为复杂的可移植和跨平台开发开辟了令人兴奋的可能性。过去几年，我们对此标准一直保持高度关注，看到它开始展现作为一个合理部署目标的能力，我们感到非常高兴。

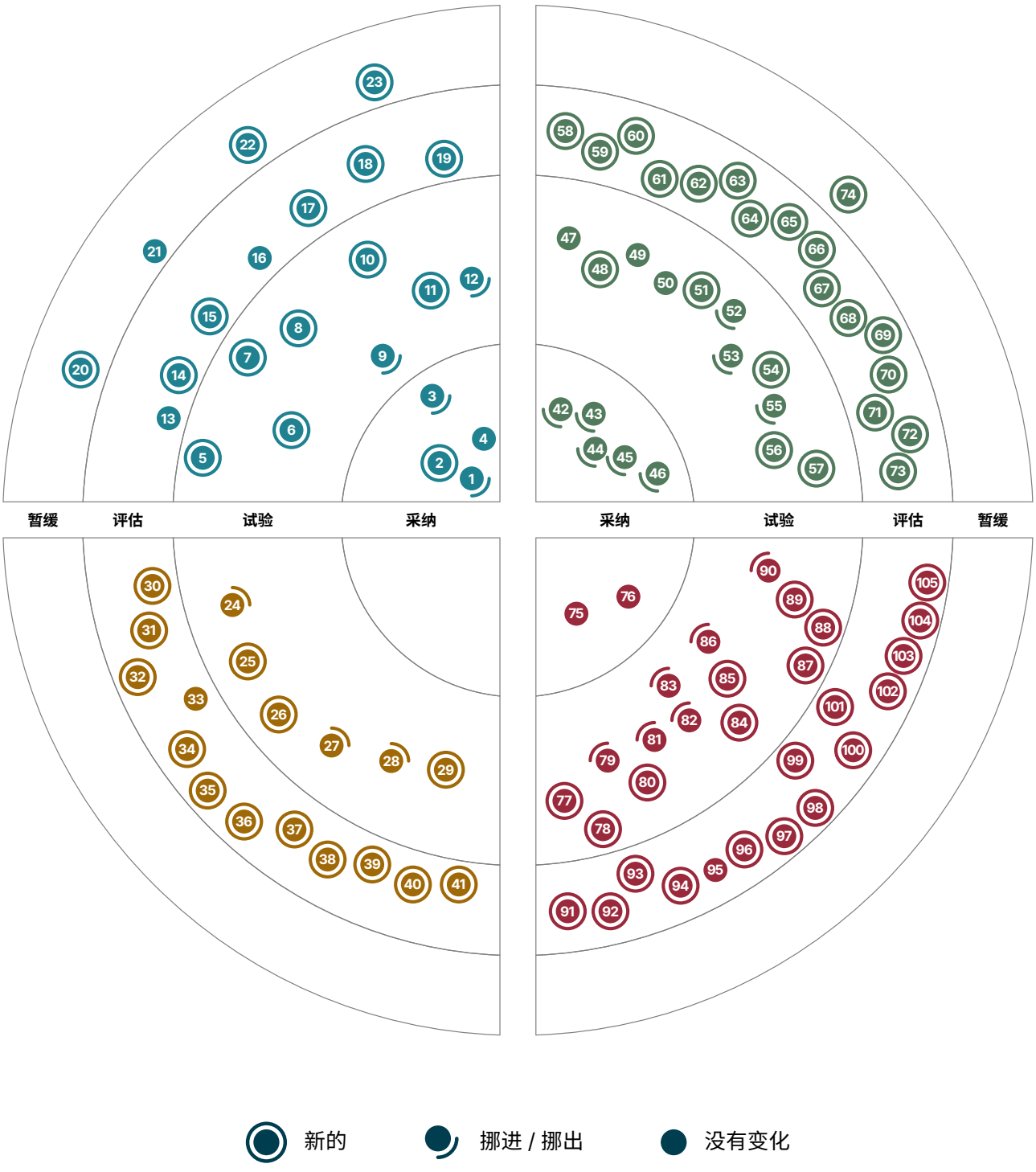
© Thoughtworks, Inc. All Rights Reserved.

生成式 AI 工具的寒武纪大爆发

根据过去几期雷达报告中设定的轨迹，我们预计生成式人工智能将在讨论中占据重要地位。然而，我们仍然对支持语言模型的技术生态的迅猛发展感到惊讶：包括安全边界、评估工具、构建智能体的工具、处理结构化输出的框架、向量数据库、云服务和可观察性工具。在许多方面，这种快速多样的增长是合理的：最初通过简单文本提示与语言模型互动的体验，如今已转变为软件产品的工程化。虽然这些产品可能无法实现人们在向 ChatGPT 发送首个提示词后所做的梦想和夸大宣传，但我们在许多客户中看到了生成式人工智能的合理高效的应用，这些工具、平台和框架在将基于 LLM 的解决方案投入生产中发挥了重要作用。就像 2015 年左右 JavaScript 生态系统的爆发一样，我们预计这种混乱的增长还将持续一段时间。

© Thoughtworks, Inc. All Rights Reserved.

本期雷达



© Thoughtworks, Inc. All Rights Reserved.

本期雷达

技术

采纳

1. 1% 金丝雀
2. 组件测试
3. 持续部署
4. 检索增强生成（RAG）

试验

5. 领域叙事（Domain storytelling）
6. 微调嵌入模型
7. 用 LLMs 进行函数调用
8. LLM as a judge
9. Passkeys
10. 小语言模型（SLMs）
11. 用于测试和训练模型的合成数据
12. 利用生成式 AI 理解遗留代码库

评估

13. AI 团队助手
14. 动态少样本提示
15. 用于数据产品的 GraphQL
16. LLM 驱动的自主代理
17. 可观察性 2.0
18. 本地设备上的大语言模型推理
19. 从 LLMs 获取结构化输出

暂缓

20. 自满于 AI 生成的代码
21. 企业范围的集成测试环境
22. 禁用大语言模型
23. 使用 AI 代替结对编程

平台

采纳

—

试验

24. Databricks Unity Catalog
25. FastChat
26. GCP Vertex AI Agent Builder
27. Langfuse
28. Qdrant
29. Vespa

评估

30. Azure AI Search
31. Databricks Delta Live Tables
32. Elastisys Compliant Kubernetes
33. FoundationDB
34. Golem
35. Iggy
36. Iroh
37. 大型视觉模型（LVM）平台
38. OpenBCI Galea
39. PGLite
40. SpinKube
41. Unblocked

暂缓

—

工具

采纳

42. Bruno
43. K9s
44. SOPS
45. 视觉回归测试工具
46. Wiz

试验

47. AWS Control Tower
48. CCMenu
49. ClickHouse
50. Devbox
51. Difftastic
52. LinearB
53. pgvector
54. Snapcraft 构建工具
55. Spinnaker
56. TypeScript OpenAPI
57. Unleash

评估

58. Astronomer Cosmos
59. ColPali
60. Cursor
61. Data Mesh Manager
62. GitButler
63. JetBrains AI Assistant
64. Mise
65. Mockoon
66. Raycast
67. ReadySet
68. Rspack
69. 语义路由
70. 软件工程代理
    Software engineering agents
71. uv
72. Warp
73. Zed

暂缓

74. CocoaPods

语言和框架

采纳

75. dbt
76. Testcontainers

试验

77. CAP
78. CARLA
79. Databricks Asset Bundles
80. Instructor
81. Kedro
82. LiteLLM
83. LlamaIndex
84. LLM Guardrails
85. Medusa
86. Pkl
87. ROS 2
88. seL4
89. SetFit
90. vLLM

评估

91. Apache XTable ™
92. dbldatagen
93. DeepEval
94. DSPy
95. Flutter for Web
96. kotaemon
97. Lenis
98. LLMLingua
99. Microsoft Autogen
100. Pingora
101. Ragas
102. Score
103. shadcn
104. Slint
105. SST

暂缓

—

© Thoughtworks, Inc. All Rights Reserved.

技术

采纳

1. 1% 金丝雀
2. 组件测试
3. 持续部署
4. 检索增强生成（RAG）

试验

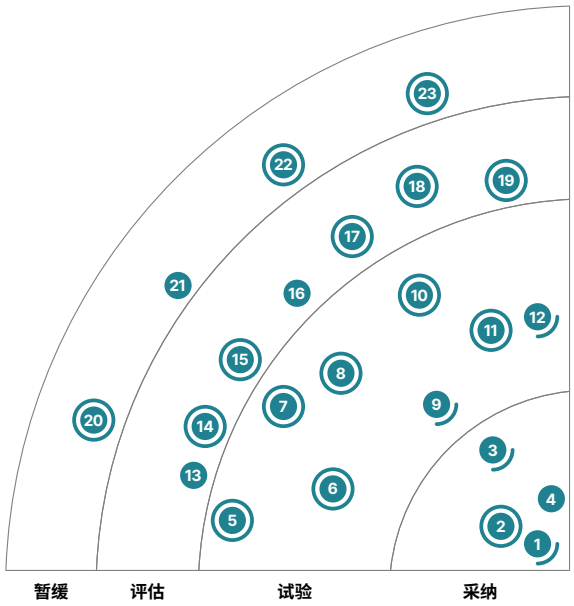
5. 领域叙事（Domain storytelling）
6. 微调嵌入模型
7. 用 LLMs 进行函数调用
8. LLM as a judge
9. Passkeys
10. 小语言模型 （SLMs）
11. 用于测试和训练模型的合成数据
12. 利用生成式 AI 理解遗留代码库

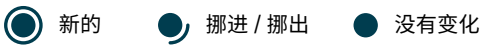
评估

13. AI 团队助手
14. 动态少样本提示
15. 用于数据产品的 GraphQL
16. LLM 驱动的自主代理
17. 可观察性 2.0
18. 本地设备上的大语言模型推理
19. 从 LLMs 获取结构化输出

暂缓

20. 自满于 AI 生成的代码
21. 企业范围的集成测试环境
22. 禁用大语言模型
23. 使用 AI 代替结对编程





1. 1%金丝雀

采纳

多年来，我们一直采用 金丝雀发布 的方法，鼓励用户对新软件版本提供早期反馈，同时通过逐步向选定用户推出更新来降低风险。1% 金丝雀是一种有效的技术，我们将新功能推出给一个非常小的用户群体（比如说 1%），这些用户是从不同类别中精心挑选的。这使团队能够快速获取用户反馈，并观察新版本对性能、稳定性等方面的影响，必要时进行学习和调整。这一技术在团队向移动应用程序或一系列设备（如边缘计算设备或软件定义汽车）推出软件更新时尤为重要。通过适当的可观察性和早期反馈，它能够在生产环境中遇到意外情况时限制影响范围。虽然金丝雀发布有助于更快地获取用户反馈，但我们认为从小部分用户开始测试是必要的，以降低大规模功能发布的风险并控制其影响。

2. 组件测试

采纳

自动化测试仍然是高效软件开发的基石。对于前端测试，我们可以讨论测试类型的分布是否应该采用经典的 测试金字塔 模型，还是应该采用 奖杯 形状。然而，无论是哪种模型，团队都应专注于组件测试，因为测试套件应该稳定且快速运行。相反，我们看到的是，团队放弃了对组件测试的掌握，而更倾向于端到端的基于浏览器的测试以及非常狭隘定义的单元测试。单元测试往往迫使组件暴露本应是纯内部的功能，而基于浏览器的测试则运行缓慢，更容易出错且更难调试。我们的建议是进行大量的组件测试，并使用类似 jsdom 这样的库在内存中运行组件测试。当然，像 Playwright 这样的浏览器工具仍然适用于端到端的测试，但不应用于组件测试。

3. 持续部署

采纳

我们认为，组织应该在可能的情况下采用持续部署实践。持续部署是指自动将每个通过自动化测试的更改部署到生产环境中的实践。这种做法是快速反馈循环的关键推动力，使组织能够更快速和高效地为客户提供价值。持续部署与持续交付的区别在于，它只要求代码“可以” 在任何时候进行部署；并不要求每个更改“实际被”部署到生产环境中。我们过去一直犹豫将持续部署移入“采纳” 环节，因为这是一种需要在软件交付其他领域具备高水平成熟度的实践，因此并不适合所有团队。然而，Thoughtworks 同事 Valentina Servile 最近出版的书籍《持续部署》提供了在组织中实施这一实践的全面指南。它为组织提供了一条实现所需成熟度以采用持续部署实践的路线图。

© Thoughtworks, Inc. All Rights Reserved.

4. 检索增强生成（RAG）

采纳

检索增强生成（RAG）是我们团队提高 LLM 生成响应质量的首选模式。我们已经在多个项目中成功应用了该技术，包括 Jugalbandi AI 平台。在 RAG 技术中，相关且可信的文档信息存储在数据库中。在给定的提示词（prompt）下，系统会查询数据库，检索相关文档，并将这些文档的内容与提示词结合，从而为 LLM 提供更丰富的上下文。这不仅提高了输出质量，还大大减少了幻觉现象。虽然随着新模型的出现，LLM 的上下文窗口（即 LLM 输入的最大大小）已经显著增长，但选择最相关的文档仍然是至关重要的一步。我们的经验表明，精心构建的较小上下文有时能比广泛的大上下文产生更好的效果。并且使用大上下文也会导致速度变慢且成本更高。我们过去依赖存储在向量数据库中的向量嵌入（embedding）来识别额外的上下文，但现在我们看到了重排序和混合搜索的趋势：搜索工具如 Elasticsearch Relevance Engine 以及诸如 GraphRAG 等利用 LLM 创建的知识图谱的方法开始被使用。在我们利用 生成式 AI 理解遗留代码库 的工作中，基于图谱的方法表现得特别出色。

5. 领域叙事（Domain storytelling）

试验

领域驱动设计（DDD）已成为我们开发软件的基础方法。我们利用它来建模事件、指导软件设计、在微服务周围建立上下文边界，以及阐述复杂的业务需求。DDD 建立了一种统一语言，非技术利益相关者和软件开发人员都可以用来有效沟通业务。一旦建立，领域模型会不断演变。但许多团队发现很难开始使用 DDD。构建初始领域模型没有一种通用的方法。我们最近遇到的一种前景看好的技术是 领域叙事（Domain storytelling）。领域叙事是一种引导技术，业务专家被提示描述业务中的活动。在专家的叙述过程中，主持人使用图示语言捕捉实体和参与者之间的关系和动作。使这些故事可视化的过程有助于澄清和发展参与者之间的共同理解。由于没有单一的最佳方法来开发领域模型，领域叙事提供了一个值得关注的替代方案，或者作为更全面的 DDD 方法，作为我们经常用来开始 DDD 的另一种技术 事件风暴 的补充。

6. 微调嵌入模型

试验

在构建基于 检索增强生成（RAG）的大语言模型应用时，嵌入的质量直接影响到相关文档的检索和响应质量。微调嵌入模型 可以提高特定任务或领域嵌入的准确性和相关性。我们的团队在开发领域特定的大语言模型应用时进行了嵌入微调，因为精确的信息提取至关重要。然而，当你 急于微调 嵌入模型之前，请仔细权衡这种方式。

© Thoughtworks, Inc. All Rights Reserved.

7. 用LLMs进行函数调用

试验

用 LLMs 进行函数调用是指通过根据给定查询和相关文档确定并调用适当的函数，将 LLM 与外部函数、API 或工具集成的能力。这将 LLM 的实用性扩展到文本生成之外，使它们能够执行特定任务，如信息检索、代码执行和 API 交互。通过触发外部函数或 API，LLM 可以执行之前超出其独立能力的操作。这一技术使 LLM 能够对其输出进行操作，有效地弥合了思想与行动之间的差距——就像人类使用工具来完成各种任务一样。通过引入函数调用，LLM 为生成过程增加了确定性和真实性，在创造力和逻辑之间达成了平衡。这种方法允许 LLM 连接到内部系统和数据库，甚至通过连接的浏览器进行互联网搜索。像 OpenAI 的 GPT 系列这样的模型支持函数调用，而像 Gorilla 这样的微调模型则专门针对从自然语言指令生成的可执行 API 调用，增强其准确性和一致性。作为一种技术，函数调用适用于 检索增强生成（RAG）和代理架构。它应被视为一种抽象的使用模式，强调其作为多种实现中基础工具的潜力，而非特定解决方案。

8. LLM as a judge

试验

许多我们构建的系统具有两个关键特征：一是能够根据大量数据集中的问题提供答案，二是几乎不可能追踪到该答案的得出过程。尽管这些系统具有不透明性，我们仍然希望评估并提高其响应质量。通过大语言模型（LLM）作为评判者的模式，我们可以使用一个 LLM 来评估另一个系统的响应，这个系统可能本身也是基于 LLM 的。我们看到这种模式用于评估产品目录中搜索结果的相关性，以及判断基于 LLM 的聊天机器人是否在合理地引导用户。当然，评估系统必须经过仔细设置和校准。这种方法能够显著提高效率，从而降低成本。这是一个正在进行的研究领域，其现状可以在 这篇文章 中找到总结。

9. Passkeys

试验

在 FIDO 联盟的引导下，并由 Apple、Google 和 Microsoft 支持，passkeys 正逐步接近主流可用性。使用 passkeys 设置新登录会生成一对密钥：网站接收公钥，而用户保留私钥。登录过程使用非对称加密，用户通过证明自己拥有私钥来进行身份验证，该私钥存储在用户设备上，永远不会发送到网站。访问 passkeys 通过生物识别或 PIN 码保护。passkeys 可以在大科技生态系统中存储和同步，如 Apple 的 iCloud 钥匙串、Google 密码管理器或 Windows Hello。对于跨平台用户，客户端到身份验证器协议（CTAP）使得 passkeys 可以存储在创建密钥或需要登录的设备之外的其他设备上。对 passkeys 最常见的反对意见是它对技术不太熟悉的用户来说是个挑战，而我们认为这种观点实际上是在自我否定。这类用户通常也不善于管理密码，因此他们最能从替代方法中受益。实际上，使用 passkeys 的系统在必要时可以回退到更传统的认证方式。

© Thoughtworks, Inc. All Rights Reserved.

10. 小语言模型（SLMs）

试验

大语言模型（LLMs）在许多应用领域中被证明是有用的，但它们的体积庞大可能会带来一些问题：响应一个提示需要大量计算资源，导致查询速度慢且成本高；这些模型是专有的，体积庞大，必须由第三方托管在云中，这可能对敏感数据造成问题；而且，在大多数情况下，训练一个模型的费用是非常高的。最后一个问题可以通过 RAG 模式来解决，该模式绕过了训练和微调基础模型的需求，但成本和隐私问题往往依然存在。为此，我们现在看到对小语言模型（SLMs）的兴趣日益增长。与更流行的 LLMs 相比，SLMs 的参数更少、精度较低，通常在 35 亿到 100 亿个参数之间。最近的研究 表明，在适当的上下文中，正确设置时，SLMs 可以执行甚至超越 LLMs。它们的体积也使得在 端侧设备 上运行成为可能。我们之前提到过谷歌的 Gemini Nano，但随着微软推出其 Phi-3 系列，该领域正在迅速发展。

11. 用于测试和训练模型的合成数据

试验

合成数据集创建涉及生成可以模拟现实世界场景的人工数据，而无需依赖敏感或有限访问的数据源。虽然合成数据在结构化数据集中的应用已得到广泛探索（例如，用于性能测试或隐私安全环境），但我们看到在非结构化数据中重新使用合成数据的趋势。企业通常面临领域特定数据缺乏标注的问题，尤其是在训练或微调大语言模型（LLMs）时。像 Bonito 和 Microsoft's AgentInstruct 这样的工具可以从原始数据源（如文本文档和代码文件）生成合成的指令调优数据。这有助于加速模型训练，同时降低成本和对手动数据管理的依赖。另一个重要的用例是生成合成数据来解决不平衡或稀疏数据的问题，这在欺诈检测或客户细分等任务中很常见。像 SMOTE 这样的技术通过人工创建少数类实例来帮助平衡数据集。同样，在金融等行业，生成对抗网络（GANs）用于模拟稀有交易，使模型在检测边缘案例方面更加稳健，从而提高整体性能。

12. 利用生成式AI理解遗留代码库

试验

生成式 AI（GenAI）和大型语言模型（LLMs）能帮助开发人员编写和理解代码。尤其是在处理遗留代码库时，这种帮助显得尤为有用，特别是当文档质量差、过时或具有误导性时。自我们上次讨论此话题以来， 利用生成式 AI 理解遗留代码库的技术和产品得到了进一步的发展，我们已经成功实践了一些方法，尤其是在 帮助大型机现代化改造的逆向工程 中。我们使用的一个特别有前景的技术是 retrieval-augmented generation（RAG）方法，其中信息检索基于代码库的知识图谱。知识图谱可以保留代码库的结构化信息，而这往往超出了 LLM 从文本代码中推导的内容。这对于那些不具备自我描述性和一致性的遗留代码库尤其有帮助。另一个提升代码理解的机会是，图谱可以通过现有的或 AI 生成的文档、外部依赖关系、业务领域知识等内容进一步丰富，从而让 AI 的工作更加轻松。

© Thoughtworks, Inc. All Rights Reserved.

13. AI 团队助手

评估

AI 编码辅助工具通常是在帮助和增强个人贡献者的工作方面被讨论的。然而，软件交付始终是团队合作的过程，因此您应该寻找创建 AI 团队助手 的方法，以帮助形成 10 倍效能的团队，而不是一群 AI 辅助孤立的 10x 工程师。幸运的是，最近工具市场的发展让我们更接近于实现这一目标。Unblocked 是一个将团队的所有知识来源汇聚在一起，并智能地整合到团队成员工具中的平台。而 Atlassian 的 Rovo 则将 AI 引入最广泛使用的团队协作平台，为团队提供新的搜索类型和访问其文档的方式，并通过 Rovo 代理解锁新的自动化和软件实践支持方式。在我们等待市场在这个领域进一步发展的同时，我们自己也在探索 AI 在知识增强和团队实践支持方面的潜力：我们开源了我们的 Haiven 团队助手，并开始收集与 非编码任务（如需求分析）相关的 AI 辅助 的学习经验。

14. 动态少样本提示

评估

动态少样本提示 构建在 少样本提示（few-shot prompting）的基础上，通过动态地在提示中包含特定的示例来引导模型的响应。调整这些示例的数量和相关性可以优化上下文长度和相关性，从而提升模型的效率和性能。像 scikit-llm 这样的库实现了这一技术，它使用最近邻搜索来获取与用户查询最相关的示例。该 技术 可以更好地利用模型有限的上下文窗口，并减少 token 消耗。开源 SQL 生成器 vanna 就是利用动态少样本提示来提高响应的准确性的。

15. 用于数据产品的 GraphQL

评估

用于数据产品的 GraphQL 是指使用 GraphQL 作为数据产品的输出端口，供客户端消费产品的技术。我们之前讨论过 GraphQL 作为一种 API 协议，它使开发者能够创建一个统一的 API 层，抽象底层数据的复杂性，为客户端提供一个更加一致且易于管理的接口。将 GraphQL 用于数据产品，使得消费者可以通过 GraphQL 结构无缝地发现数据格式和关系，并使用熟悉的客户端工具进行各种操作。我们的团队正在探索将这一技术应用于特定场景，例如通过 talk-to-data 来探索和发现大数据洞察。在此场景下，GraphQL 查询由大语言模型（LLM）根据用户提示词构建，同时还可以在提示词中引入 GraphQL 结构以便于 LLM 参考以生成准确的查询语句。

16. LLM 驱动的自主代理

评估

随着诸如 Autogen 和 CrewAI 等框架的出现，LLM 驱动的自主代理正在超越单一代理和静态的多代理系统。这项技术允许开发人员将复杂的任务分解为多个小任务，再交由不同角色的代理完成。开发人员可以使用预配置的工具来执行任务，代理之间通过对话来协调任务流程。该技术仍处于早期开发阶段。在我们目前的实验中，团队遇到了一些问题，比如说代理陷入持续循环和行为失控。像 LangGraph 这样的库提供了更大的代理交互控制能力，能够以图的形式定义流程。如果你使用这项技术，我们建议实施一些安全机制，包括超时处理和人工监控。

© Thoughtworks, Inc. All Rights Reserved.

17. 可观察性 2.0

评估

可观察性 2.0 代表了一种从传统的、分散的监控工具向统一方法的转变，它利用结构化的、高基数的事件数据存储在单一数据存储中。该模型捕捉丰富的、带有详细元数据的原始事件，为综合分析提供单一的真实来源。通过以原始形式存储事件，这种方法简化了关联性分析，支持实时和取证分析，并能够深入了解复杂的分布式系统。它允许高分辨率的监控和动态调查能力。可观察性 2.0 优先捕捉高基数和高维度数据，能够进行详细的检查而不会产生性能瓶颈。统一的数据存储减少了复杂性，提供了系统行为的连贯视图，并使可观察性实践与软件开发生命周期更加紧密地结合。

18. 本地设备上的大语言模型推理

评估

大语言模型（LLMs）现在可以在网络浏览器和智能手机、笔记本电脑等边缘设备上运行，这使得本地 AI 应用成为可能。这允许在不传输到云端的情况下安全处理敏感数据，为边缘计算、实时图像或视频处理等任务提供极低的延迟，通过本地计算降低成本，并在网络连接不稳定或不可用时依然能够正常工作。这是一个活跃的研究和开发领域。我们之前曾提到过 MLX，这是一个在 Apple Silicon 上高效执行机器学习的开源框架。其他新兴工具包括 Transformers.js 和 Chatty 。Transformers.js 允许你通过 ONNX Runtime 在浏览器中运行 transformers，支持从 PyTorch、TensorFlow 和 JAX 转换的模型。Chatty 利用 WebGPU 在浏览器中本地和私密地运行 LLMs，提供了功能丰富的浏览器内 AI 体验。

19. 从LLMs 获取结构化输出

评估

从 LLMs 获取结构化输出 是指通过定义的结构模式来约束语言模型的响应。这可以通过指示通用模型以特定格式响应，或者通过微调模型使其“原生” 输出例如 JSON 的结构化数据来实现。OpenAI 现在支持结构化输出，允许开发人员提供 JSON Schema、pydantic 或 Zod 对象来约束模型响应。这种能力在函数调用、API 交互和外部集成中尤其有价值，因为这些场景中格式的准确性和一致性至关重要。结构化输出不仅提升了 LLMs 与代码交互的方式，还支持更广泛的使用场景，例如生成用于呈现图表的标记语言。此外，结构化输出已被证明可以减少模型输出中的幻觉现象。

© Thoughtworks, Inc. All Rights Reserved.

20. 自满于 AI 生成的代码

暂缓

AI 编程助手，如 GitHub Copilot 和 Tabnine，已经变得非常受欢迎。根据 StackOverflow 2024 年开发者调查 的数据，“72% 的受访者对开发中的 AI 工具持赞成或非常赞成的态度”。尽管我们也看到了这些工具的好处，但我们对它们在中长期对代码质量的影响持谨慎态度，并提醒开发者警惕 自满于 AI 生成的代码 。在经历了几次积极的 AI 辅助体验后，很容易在审查 AI 建议时变得不够谨慎。像 GitClear 的这项研究 显示了代码库快速增长的趋势，我们怀疑这与更大的 Pull Request 有关。还有 GitHub 的这项研究 让我们开始思考，提到的 15% 的 Pull Request 合并率的增加是否真的是好事，还是人们因为过于信任 AI 的结果而更快地合并了更大的请求。我们仍在使用 一年多前提供的基本 “入门建议”，也就是要警惕自动化偏见、沉没成本谬误、锚定偏见和审查疲劳。我们还建议程序员建立一个良好的 在何时何地不使用和信任 AI 心理框架 。

21. 企业范围的集成测试环境

暂缓

创建企业范围的集成测试环境是一种常见且浪费的做法，会拖慢整个开发流程。这些环境往往成为难以复制的珍贵资源，成为开发的瓶颈。由于不同环境之间不可避免的数据和配置开销差异，它们还会提供一种虚假的安全感。讽刺的是，常见的对替代方案（如临时环境或多个本地测试环境）的反对意见是成本问题。然而，这种观点未能考虑到企业级集成测试环境所造成的延迟成本，因为开发团队往往需要等待其他团队完成任务或等待依赖系统的新版本部署。相反，团队应使用临时环境，最好是开发团队拥有的一系列测试，这些测试可以快速创建和销毁，使用替身（fake stubs）而不是实际的副本。关于支持这种替代方案的其他技术，可以查看 契约测试 、将部署和发布解耦 、关注平均恢复时间 和 在生产环境测试 。

22. 禁用大语言模型

暂缓

与其在工作场所全面禁用大语言模型，组织应专注于提供一套经过批准的 AI 工具。禁令只会促使员工寻找未经批准且可能不安全的替代方案，带来不必要的风险。正如个人计算机早期一样，人们会使用他们认为有效的工具来完成工作，无论是否存在障碍。如果公司不提供安全且获得认可的替代方案，员工可能会使用未经批准的大语言模型，这会带来知识产权、数据泄露和法律责任的风险。相反，提供安全、经企业批准的大语言模型或 AI 工具可以确保安全性和生产力的双赢。一个良好管理的方案能够帮助组织管理数据隐私、安全性、合规性和成本问题，同时赋能员工利用大语言模型的功能。最理想的情况下，对 AI 工具的妥善管理访问可以加速组织学习如何在工作场所中最好地利用 AI。

© Thoughtworks, Inc. All Rights Reserved.

23. 使用AI代替结对编程

暂缓

当人们谈论编码助手的时候，关于 结对编程 的话题就会不可避免地被提及。我们所处的行业对于它爱恨交织：有的同行信任它，有的同行讨厌它。现在大家都会对编码助手们提出同样的问题：一个程序员可以选择与人工智能，而不是另外一个程序员，进行结对编程，从而达到同样的团队产出吗？ Github Copilot 甚至自称为“你的 AI 结对程序员”。然而当大家都认为编程助手可以在结对编程方面带来好处时，我们还是不建议完全 使用 AI 代替结对编程。把编码助手当做结对编程者忽略了结对编程的一个关键收益：它可以让团队而不只是个人变得更好。在帮助解决难题，学习新技术，引导新人，或者提高技术任务的效率从而让团队更关注战略性设计等这些方面，使用编程助手确实大有裨益。但在诸如将正在进行的工作的数量控制在低水平，减少团队交接与重复学习，让持续集成成为可能，或者改善集体代码所有权等等这些团队合作相关的层面，它没有带来什么好处。

© Thoughtworks, Inc. All Rights Reserved.

平台

采纳

—

试验

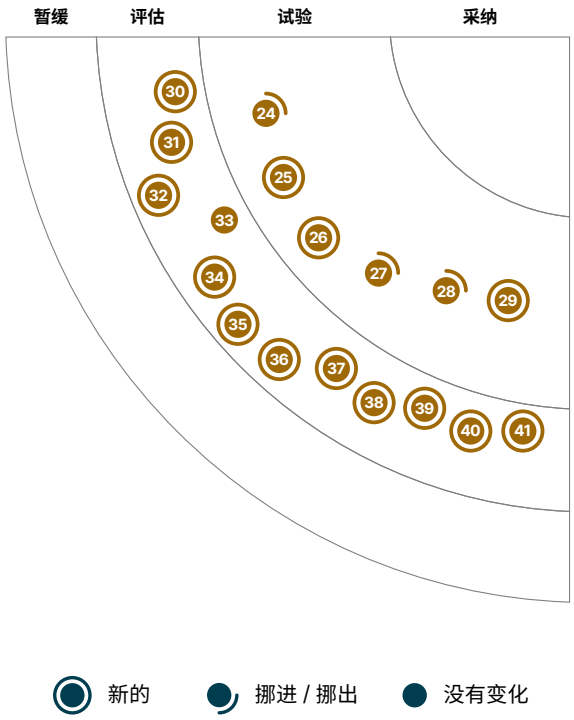
24. Databricks Unity Catalog
25. FastChat
26. GCP Vertex AI Agent Builder
27. Langfuse
28. Qdrant
29. Vespa

评估

30. Azure AI Search
31. Databricks Delta Live Tables
32. Elastisys Compliant Kubernetes
33. FoundationDB
34. Golem
35. Iggy
36. Iroh
37. 大型视觉模型（LVM）平台
38. OpenBCI Galea
39. PGLite
40. SpinKube
41. Unblocked

暂缓

—



24. Databricks Unity Catalog

试验

Databricks Unity Catalog 是一种用于数据治理的解决方案，适用于在 lakehouse 中的资产，例如文件、表或机器学习模型。它是开源 Unity Catalog 的托管版本，可用于管理和查询存储在外部存储或由 Databricks 管理的数据。过去，我们的团队使用了多种数据管理解决方案，如 Hive metastore 或 Microsoft Purview。然而，Unity Catalog 对治理、元数据存储管理和数据发现的综合支持，使其颇具吸引力，因为它减少了管理多个工具的需求。我们团队发现的一个问题是 Databricks 托管的 Unity Catalog 缺乏自动灾难恢复功能。虽然他们能够自行配置备份和恢复功能，但由 Databricks 提供的解决方案会更加便捷。需要注意的是，虽然这些治理平台通常会实施集中化的解决方案，以确保工作空间和工作负载之间的一致性，但可以通过让各个团队管理自己的资产，将治理责任下放，从而实现联邦化的治理模式。

25. FastChat

试验

FastChat 是一个开放平台，用于训练、服务和评估大型语言模型。我们的团队利用其模型服务能力来托管多个模型 — Llama 3.1（8B and 70B）、Mistral 7B 和 Llama-SQL — 出于不同的目的，所有模型均以一致的 OpenAI API 格式运行。FastChat 采用控制器 - 工作者架构，允许多个工作者托管不同的模型。它支持不同类型的工作者，如 vLLM、LiteLLM 和 MLX 。我们选择使用 vLLM 模型工作者，以利用其在高吞吐量的优势。根据使用案例的不同（比如延迟或吞吐量），可以创建和扩展不同类型的 FastChat 模型工作者。例如，用于开发者 IDE 中代码建议的模型需要低延迟，这就可以通过多个 FastChat 工作者进行扩展，以有效处理并发请求。相反，用于 Text-to-SQL 的模型由于需求较低或性能要求不同，则不需要多个工作者。我们的团队利用 FastChat 的扩展能力进行 A/B 测试。我们用相同的模型但不同的超参数（Hyperparameter）值配置 FastChat 工作者，并向每个工作者提出相同的问题，从而识别最佳的超参数（Hyperparameter）值。在在线服务中切换模型时，我们进行 A/B 测试以确保平滑迁移。 例如，我们最近将代码建议的模型从 CodeLlama 70B 迁移到 Llama 3.1 70B 。通过同时运行这两个模型并比较输出，我们验证了新模型在性能上达到了或超过了之前的模型，同时没有打断开发者的使用体验。

26. GCP Vertex AI Agent Builder

试验

GCP Vertex AI Agent Builder 提供了一个灵活的平台，可以通过自然语言或代码优先的方式创建 AI 代理。该工具通过第三方连接器无缝集成企业数据，并且拥有构建、原型设计和部署 AI 代理所需的全部工具。随着对 AI 代理需求的增加，许多团队在理解其优势和实施上面临困难。GCP Vertex AI Agent Builder 使开发者能够更快地进行代理的原型设计，并以最小的设置处理复杂的数据任务。我们的开发者发现它特别适用于构建基于代理的系统，例如知识库或自动化支持系统，这些系统可以高效地管理结构化和非结构化数据。因此，这是一款开发 AI 驱动解决方案的有价值工具。

© Thoughtworks, Inc. All Rights Reserved.

27. Langfuse

试验

LLM（大型语言模型）像黑箱一样运作，非常难以确定它的行为。可观察性对于打开这个黑箱并理解 LLM 应用程序在生产环境中的运作至关重要。我们团队在使用 Langfuse 方面有过积极的体验，我们曾用它来观察、监控和评估基于 LLM 的应用程序。它的追踪、分析和评估能力使我们能够分析完成性能和准确性，管理成本和延迟，并理解生产使用模式，从而促进持续的数据驱动改进。仪器数据提供了请求 - 响应流和中间步骤的完整可追溯性，这可以作为测试数据，在部署新变更之前验证应用程序。我们已将 Langfuse 与 RAG（检索增强生成）等 LLM 架构，以及 大语言模型驱动的自主代理 一起使用。例如，在基于 RAG 的应用程序中，分析低评分的对话追踪有助于识别架构的哪个部分（如预检索、检索或生成）需要改进。当然，在这一领域，另一个值得考虑的选项是 Langsmith 。

28. Qdrant

试验

Qdrant 是一个开源的向量相似度搜索引擎和数据库，使用 Rust 编写。它支持多重文本和多模态密集 向量嵌入模型 。我们的团队使用了诸如 MiniLM-v6 和 BGE 等开源嵌入模型，将其应用于多的产品知识库中。我们把 Qdrant 作为一个企业级 Vector Store 使用，利用 多租户 技术将向量嵌入存储为独立的集合，从而隔离不同产品的知识库。用户访问策略则在应用层中进行管理。

29. Vespa

试验

Vespa 是一个开源搜索引擎和大数据处理平台。它特别适合于需要低延迟和高吞吐量的应用。我们的团队喜欢 Vespa 实现混合搜索的能力，能够使用多种检索技术高效过滤和排序多种类型的元数据，实施多阶段排名，针对每个文档索引多个向量（例如，对于每个数据块），而无需将所有元数据复制到单独索引的文档中，并能够同时从多个索引字段中检索数据。

30. Azure AI Search

评估

Azure AI Search，前称 Cognitive Search，是一个云端搜索服务，专为处理结构化和非结构化数据而设计，适用于知识库等应用场景，尤其适用于检索 增强生成（RAG）设置。它支持多种搜索方式，包括关键字搜索、向量搜索和混合搜索，我们认为这些功能将变得越来越重要。该服务可以自动导入常见的非结构化数据格式，例如 PDF、DOC 和 PPT，从而简化创建可搜索内容的流程。此外，它还与其他 Azure 服务集成，如 Azure OpenAI，使用户能够以最少的手动集成工作构建应用程序。根据我们的经验，Azure AI Search 性能可靠，非常适合在 Azure 环境中托管的项目。通过其 定制技能，用户还可以定义特定的数据处理步骤。总体而言，如果您在 Azure 生态系统中工作并需要为 RAG 应用构建强大的搜索解决方案，Azure AI Search 值得考虑。

© Thoughtworks, Inc. All Rights Reserved.

31. Databricks Delta Live Tables

评估

Databricks Delta Live Tables 是一个声明式框架，旨在构建可靠、可维护和可测试的数据处理管道。它允许数据工程师使用声明式方法定义数据转换，并自动管理底层基础设施和数据流。Delta Live Tables 的一个突出特点是其强大的监控能力。它提供了整个数据管道的有向无环图（DAG），直观地表示从源到最终表的数据移动。这种可见性对于复杂的流水线至关重要，帮助数据工程师和数据科学家跟踪数据血缘和依赖关系。Delta Live Tables 深度集成到 Databricks 生态系统中，这也带来了一些定制接口的挑战。我们建议团队在使用 Delta Live Tables 之前仔细评估输入和输出接口的兼容性。

32. Elastisys Compliant Kubernetes

评估

Elastisys Compliant Kubernetes 是一个专门设计的 Kubernetes 发行版，旨在满足严格的监管和合规要求，尤其适用于医疗、金融和政府等高度监管的行业。它自动化了安全流程，支持多云和本地部署，并建立在零信任安全架构之上。由于其内置符合 GDPR 和 HIPAA 等法律法规的要求，并具备 ISO27001 等控制措施，对于需要安全、合规且可靠的 Kubernetes 环境的公司来说，这是一个具有吸引力的选择。

33. FoundationDB

评估

FoundationDB 是一个多模型数据库，2015 年被苹果公司收购，并于 2018 年 4 月开源。FoundationDB 的核心是一个分布式键值存储，提供严格的可序列化事务。自从我们在技术雷达中首次提到它以来，它已经有了显著的改进，包括智能数据分布以避免写入热点、新的存储引擎、性能优化以及 多区域复制 支持。我们正在一个正在进行的项目中使用 FoundationDB，并对其 解耦架构 印象深刻。该架构允许我们独立扩展集群的不同部分。例如，我们可以根据特定的工作负载和硬件调整事务日志、存储服务器和代理的数量。尽管功能丰富，FoundationDB 仍然非常易于运行和操作大型集群。

34. Golem

评估

持久计算（Durable computing）是分布式计算中的一个新兴运动，它使用显式状态机的架构风格来持久化无服务器计算的内存，以提高容错性和恢复能力。Golem 是推动这一运动的倡导者之一。该概念在某些场景下有效，例如长时间运行的微服务 saga 或 AI 代理的长时间工作流。在之前的雷达报告中，我们评估了 Temporal 以实现类似目的，而 Golem 是另一个选择。使用 Golem，您可以用除 Golem 之外的任何支持语言编写 WebAssembly 组件，同时 Golem 具有确定性并支持快速启动时间。我们认为 Golem 是一个值得评估的令人兴奋的平台。

© Thoughtworks, Inc. All Rights Reserved.

35. Iggy

评估

Iggy 是一个用 Rust 编写的持久化消息流平台，虽然是一个相对较新的项目，但其功能非常出色。它已经支持多个流、主题和分区、至多一次投递、消息过期，另外它还通过 QUIC、TCP 和 HTTP 协议支持 TLS。Iggy 以单服务器的方式运行， 目前在读写操作中都能实现高吞吐量。随着即将引入的集群和 io_uring 支持，Iggy 很有可能成为 Kafka 的一个替代方案。

36. Iroh

评估

Iroh 是一个相对较新的分布式文件存储和内容交付系统，旨在作为现有去中心化系统，如 IPFS (InterPlanetary File System) 的演进。Iroh 和 IPFS 都可以用来创建去中心化网络，以存储、共享和访问使用不透明内容标识符寻址的内容。然而，Iroh 去掉了 IPFS 的一些 实现限制，例如没有最大区块大小，并通过对文档进行 基于范围的集合重整 提供数据同步机制。该项目的路线图包括通过 WASM 将技术带入浏览器，这为在 Web 应用程序中构建去中心化带来了令人兴奋的可能性。如果您不想自己托管 Iroh 节点，可以使用其云服务 iroh.network。目前已经有多个 SDK 可供多种语言使用，其目标是比替代的 IPFS 系统更具用户友好性和易用性。尽管 Iroh 仍处于早期阶段，但值得关注，因为它有可能成为去中心化存储领域的重要参与者。

37. 大型视觉模型（LVM）平台

评估

大语言模型 (LLMs) 在当前吸引了如此多的关注，以至于我们往往忽略了大型视觉模型 (LVMs) 的持续发展。这些模型可用于分割、合成、重建和分析视频流和图像，有时还结合了扩散模型或标准卷积神经网络。尽管 LVMs 有潜力彻底改变我们处理视觉数据的方式，但在生产环境中适应和应用这些模型仍面临重大挑战。例如，视频数据在收集训练数据、分割和标注对象、微调模型以及部署和监控这些模型时，带来了独特的工程挑战。与 LLMs 更适合简单的聊天界面或纯文本 API 不同，计算机视觉工程师或数据科学家必须管理、版本化、注释和分析大量的视频流数据，这项工作需要一个可视化界面。大型视觉模型 (LVM) 平台是新兴的一类工具和服务，其中包括 V7、 Nvidia Deepstream SDK 和 Roboflow，这些平台正在解决这些挑战。Deepstream 和 Roboflow 对我们特别有吸引力，因为它们结合了用于管理和标注视频流的集成 GUI 开发环境，同时提供了 Python、C++ 或 REST API 以便从应用代码中调用模型。

© Thoughtworks, Inc. All Rights Reserved.

38. OpenBCI Galea

评估

对脑机接口（BCI）及其在辅助技术中的潜在应用的兴趣正在不断增长。使用脑电图（EEG）和其他电生理信号的非侵入性技术为恢复中的患者提供了一种比脑植入物更低风险的替代方案。现在，研究人员和企业家可以在新兴的平台上构建创新应用，而不必担心低级信号处理和集成挑战。这样的平台的例子包括 Emotive 和 OpenBCI，它们提供开源硬件和软件用于构建 BCI 应用程序。OpenBCI 的最新产品 OpenBCI Galea 将 BCI 与 VR 头显的功能结合在一起。它为开发人员提供了一系列时间锁定的生理数据流，以及空间定位传感器和眼动追踪。这些传感器数据可以用于控制各种物理和数字设备。该 SDK 支持多种语言，并在 Unity 或 Unreal 中提供传感器数据。我们很高兴看到这一能力在开源平台上提供，这样研究人员就能获得他们在这一领域创新所需的工具和数据。

39. PGLite

评估

PGLite 是一个构建于 WASM 的 PostgreSQL 数据库。与之前需要 Linux 虚拟机的尝试不同，PGLite 直接将 PostgreSQL 构建为 WASM，允许你完全在 web 浏览器中运行它。你可以在内存中创建一个临时数据库，或者通过 indexedDB 将其持久化到磁盘中。自从我们在 Radar 中上次提到 本地优先应用 以来，工具集已经有了显著发展。借助 Electric 和 PGLite，你现在可以基于 PostgreSQL 构建响应式的本地优先应用。

40. SpinKube

评估

SpinKube 是一个在 Kubernetes 上运行的开源无服务器（serverless）WebAssembly 运行时。虽然 Kubernetes 提供了强大的自动扩展能力，但容器的冷启动时间仍然可能需要预先配置以应对高峰负载。我们认为，WebAssembly 的毫秒级启动时间为按需工作负载提供了更加动态和灵活的无服务器（serverless）解决方案。自从我们上次讨论 Spin 以来， WebAssembly 生态系统已经取得了显著的进展。我们很高兴推荐 SpinKube 这一平台，它简化了在 Kubernetes 上基于 WebAssembly 工作负载的开发和部署过程。

41. Unblocked

评估

Unblocked 提供软件开发生命周期（SDLC）资产和工件的发现功能。它与常见的应用程序生命周期管理（ALM）和协作工具集成，帮助团队理解代码库及相关资源。通过提供代码的即时相关上下文，Unblocked 改善了代码理解，使导航和理解复杂系统变得更加容易。工程团队可以安全合规地访问与其工作相关的讨论、资产和文档。Unblocked 还捕捉并分享了通常掌握在经验丰富团队成员手中的本地知识，使宝贵的见解能够为所有人访问，不论经验水平如何。

© Thoughtworks, Inc. All Rights Reserved.

工具

采纳

42. Bruno
43. K9s
44. SOPS
45. 视觉回归测试工具
46. Wiz

试验

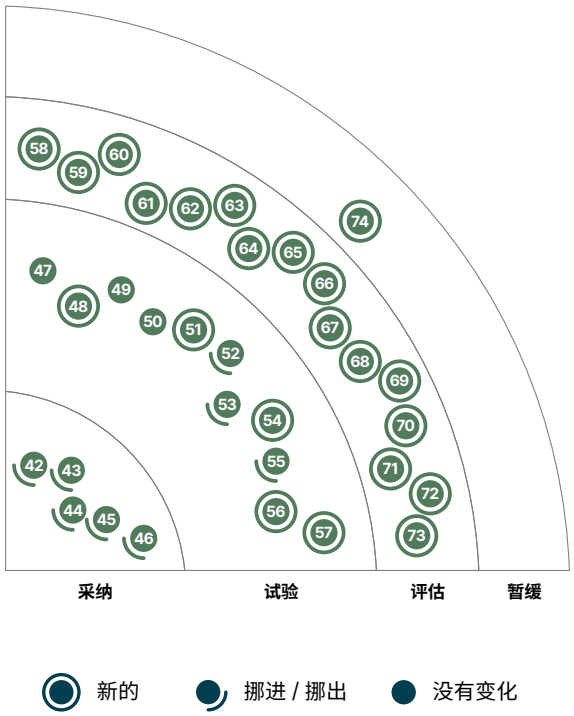
47. AWS Control Tower
48. CCMenu
49. ClickHouse
50. Devbox
51. Difftastic
52. LinearB
53. pgvector
54. Snapcraft 构建工具
55. Spinnaker
56. TypeScript OpenAPI
57. Unleash

评估

58. Astronomer Cosmos
59. ColPali
60. Cursor
61. Data Mesh Manager
62. GitButler
63. JetBrains AI Assistant
64. Mise
65. Mockoon
66. Raycast
67. ReadySet
68. Rspack
69. 语义路由
70. 软件工程代理 Software engineering agents
71. uv
72. Warp
73. Zed

暂缓

74. CocoaPods



42. Bruno

采纳

Bruno 是一个用于 API 测试、开发和调试的开源桌面工具，类似于 Postman 和 Insomnia 。它旨在通过简单的离线设计提供更卓越的协作、隐私和安全性。集合直接存储在您的文件系统中，采用自定义的纯文本标记语言 Bru Lang 编写，可以通过 Git 或您选择的版本控制工具进行共享以便协作。Bruno 既可作为桌面应用程序使用，也可作为 CLI tool。它还提供了官方的 VS Code 扩展，并计划支持其他 IDE。Bruno 已成为多支 Thoughtworks 团队的默认选择，但我们也建议团队在 VPN 和代理环境下工作时保持警惕，因为在这些情况下发出的请求 可能会意外失败。

43. K9s

采纳

K9s 通过集成更详细的图表和视图，显著地提升了其可视化能力。它现在提供了更友好的日志和指标展示功能，并更加灵活地支持自定义资源（CRDs）的显示。对于 Pods 的操作也得到了扩展，包括与调试工具（如 kubectl debug）更深入的集成，以及对多集群环境的增强支持。对于 CRD 的支持有显著地提升，现在提供了更好的资源导航和管理能力，并且可以更加流畅地与自定义资源进行交互。快捷键面板也得到了增强，以便于对 kubectl 不太熟悉的开发者使用。K9s 最初主要专注于 DevOps 团队，所以这次改进可以看作是它一次重大的改进。

44. SOPS

采纳

SOPS 是一个加密文件编辑器，支持多种文件格式的加密，并与密钥管理服务（KMS）兼容。我们在处理秘密管理时的建议一直是要将其 与源代码解耦。然而，当面临在完全自动化（符合 基础设施即代码 ）的精神下）和一些手动步骤（使用像 vaults 这样的工具）之间进行选择以管理、种子和轮换种子机密时，团队常常需要权衡。例如，我们的团队使用 SOPS 来管理引导基础设施的种子凭证。然而，在某些情况下，无法从遗留代码库中移除 Secret。在这些情况下，我们使用 SOPS 来加密文本文件中的 Secret。SOPS 可以与云管理的密钥存储（如 AWS 和 GCP 密钥管理服务（KMS）或 Azure Key Vault）集成，作为加密密钥的来源。它还支持跨平台使用，并支持 PGP 密钥。我们的几个团队在需要管理代码库中的秘密时默认使用 SOPS。

45. 视觉回归测试工具

采纳

我们之前强调过视觉回归测试工具 ，并观察到它们的算法从原始的像素级比较进化到更复杂的模式匹配和光学字符识别（OCR）。早期的视觉回归工具产生了很多误报，只有在界面稳定的后期开发阶段才有用。BackstopJS 通过配置选择器和视口来定位页面上的特定元素进行视觉测试，避免了这个问题。但机器学习使得检测和比较视觉元素更加准确，即使在这些元素移动或包含动态内容的情况下。这些工具变得越来越有用，并且具备利用 AI 和机器学习最新进展的优势。现在，几个商业工具，如 Applitools 和 Percy ，声称在其视觉回归测试中使

© Thoughtworks, Inc. All Rights Reserved.

用了 AI。我们的一支团队广泛使用了 Applitools Eyes，并对结果感到满意。尽管视觉回归测试不能替代编写良好的端到端功能测试，但它们是测试工具箱中的一个宝贵补充。我们正积极推动它们的采用，因为它们已经成为全面 UI 测试策略中的一个安全默认选项。

46. Wiz

采纳

Wiz 已成为我们多个项目中的云安全平台首选。我们的团队喜欢它能够比类似工具更早地检测到风险和威胁，因为它能够持续扫描变更。Wiz 可以检测并警报尚未部署到生产环境的工件（如容器镜像、基础设施代码）以及生产环境中的工作负载（如容器、虚拟机和云服务）的错误配置、漏洞和泄露的机密。我们也非常欣赏它为开发团队和领导团队提供的强大报告功能。该分析帮助我们了解漏洞如何影响特定服务，从而能够在该上下文中解决问题。

47. AWS Control Tower

试验

AWS Control Tower 仍然是我们在多团队环境中管理 AWS 账户的首选工具。它提供了一个便捷的机制，可以预配置安全和合规控制，这些控制将自动应用于新的着陆区。这是 在变更点合规 的一个实例，因为这些控制在创建新基础设施时被应用和验证，消除了后续进行手动合规检查的需求。AWS Control Tower Account Factory for Terraform（AFT） 自我们上次使用以来不断发展，现在已在更多的 AWS 区域提供。AFT 允许通过 基础设施即代码 流水线来创建 Control Tower 账户。我们喜欢 AFT 的定制化能力，它可以通过发送 webhooks 或采取特定操作，安全地与外部工具 GitHub Actions 集成。我们的团队报告说使用 AWS Control Tower 管理账户效果很好，但我也希望 AWS 能接受社区对该项目的贡献，尤其是在我们发现改进机会时。

48. CCMenu

试验

对于实践持续集成（CI）的团队来说，了解中央构建在 CI 中的状态非常重要。在疫情之前，团队会议室中的大屏幕 仪表盘 可以让人一目了然地获取这些信息。随着远程工作成为常态，团队需要一种适用于开发者个体工作站的解决方案。在 Mac 上，CCMenu 就是一款这样的工具，这是一个由 Thoughtworks 员工编写的小应用程序。最初它是 CruiseControl 的一部分，适用于所有可以提供 cctray 格式信息的服务器，包括 Jenkins 和 TeamCity。最近的重写增加了对 GitHub Actions 的支持，并为更深入集成更多 CI 服务器和身份验证方式奠定了基础。

© Thoughtworks, Inc. All Rights Reserved.

49. ClickHouse

试验

ClickHouse 是一个开源的列式在线分析处理（OLAP）数据库，用于实时分析。它于 2009 年作为一个实验项目启动，之后发展成为一个高性能且线性可扩展的分析数据库。其高效的查询处理引擎结合数据压缩，使其适合在不进行预聚合的情况下运行交互式查询。ClickHouse 也是 OpenTelemetry 数据的优秀存储选择。它与 Jaeger 的集成允许您存储大量的追踪数据并高效分析。

50. Devbox

试验

尽管开发工具不断进步，保持一致的本地开发环境仍然是许多团队面临的挑战。为新工程师进行入职设置通常需要运行命令或自定义脚本，而这些操作可能会在不同机器上不可预测地失败，导致环境不一致。为了解决这个问题，我们的团队越来越依赖 Devbox。Devbox 是一个命令行工具，提供了简洁的界面，用于创建可复现的、按项目定义的本地开发环境，它利用了 Nix 包管理器，但不使用虚拟机或容器。Devbox 极大地简化了团队的入职流程，因为一旦为代码库配置好环境，在新设备上只需一个 CLI 命令（devbox shell）就能复现已定义的环境。Devbox 支持 shell 钩子、自定义脚本以及生成 devcontainer.json，以便与 VSCode 集成。

51. Difftastic

试验

Difftastic 是一种用于在语法感知的基础上高亮显示代码文件差异的工具，和传统的文本 diff 工具（例如经典的 Unixdiff 命令）有很大不同。例如，在像 Java 或 TypeScript 这样的以分号分隔的语言中，Difftastic 会忽略为了分割长语句而插入的换行符。该工具仅突出显示对程序语法有影响的更改。它首先将文件解析为抽象语法树，然后使用 Dijkstra 算法计算它们之间的距离。我们发现，Difftastic 在审查大型代码库时特别有用。只要编程语言有解析器，它就可以用于任何编程语言，并且开箱即用地支持 50 多种编程语言以及 CSS、HTML 等结构化文本格式。尽管这不是一个新工具，但在大语言模型（LLM）代码助手时代，人工审查越来越庞大的代码库变得至关重要，我们认为有必要强调这一工具的价值。

52. LinearB

试验

LinearB 是一个软件工程智能平台，为我们的工程领导者提供数据驱动的洞察，以支持持续改进。它对关键领域进行对齐，如基准测试、工作流自动化以及增强开发者体验和生产力的针对性投资。我们对 LinearB 的体验表明，它能够在工程团队中培育改进和效率的文化。我们的团队使用该平台来跟踪关键的工程指标，识别需要改进的领域，并实施基于证据的行动。这些功能与 LinearB 的核心价值主张高度一致：基准测试、自动化收集指标，并实现数据驱动的改进。LinearB 集成了源代码、应用生命周期、CI/CD 和沟通工具，使用预配置和自定义的工程指标，提供有关开发者体验、生产力和团队绩效的全面定量洞察。作为 交付核心四指标 的支持者，

© Thoughtworks, Inc. All Rights Reserved.

我们特别欣赏 LinearB 对这些特定指标的强烈关注，以及其衡量软件交付性能关键方面的能力，这对于提升效率至关重要。历史上，团队在收集交付核心四指标特定指标时会面临挑战，往往依赖复杂的自定义仪表盘或手动过程。LinearB 持续提供一个引人注目的解决方案，能够自动跟踪这些指标，并提供实时数据，以支持围绕开发者体验、生产力和可预测性进行的主动决策。

53. pgvector

试验

pgvector 是一个开源的 PostgreSQL 扩展，用于进行向量相似性搜索，允许将向量与结构化数据一起存储在单一且成熟的数据库中。虽然它缺少一些专用向量数据库的高级功能，但它受益于 PostgreSQL 的 ACID 合规性、时间点恢复等强大功能。随着生成式 AI 驱动的应用程序的兴起，我们看到越来越多的模式是存储并有效搜索嵌入向量以进行相似性匹配，pgvector 有效地解决了这一需求。pgvector 在生产环境中的使用日益增多，尤其是在团队已经使用云提供商管理的 PostgreSQL 时，它表现出能够满足常见向量搜索需求，而无需单独的向量存储库。我们的团队在对比结构化和非结构化数据的项目中发现了它的价值，展示了其广泛采用的潜力，因此我们将其移动到试验环（Trial ring）。

54. Snapcraft 构建工具

试验

Snapcraft 是一个开源的命令行工具，用于在 Ubuntu、其他 Linux 发行版和 macOS 上构建和打包名为 snaps 的自包含应用程序。Snaps 可以在包括 Linux 机器、虚拟环境和车辆车载计算机系统在内的硬件平台上轻松部署和维护。虽然 Snapcraft 提供了一个公共的 应用商店 用于发布 snaps，但我们的团队使用这个构建工具将自动驾驶系统打包为 snap，而不将其发布到公共应用商店。这使我们能够在本地构建、测试和调试嵌入式软件系统，同时将其发布到内部工件库。

55. Spinnaker

试验

Spinnaker 是由 Netflix 创建的一个开源持续交付平台。它将集群管理和云端烘焙镜像的部署作为核心功能。我们喜欢 Spinnaker 对微服务部署的特定做法。在之前的版本中，我们提到了缺乏将流水线配置为代码的能力，但这一问题已通过添加 spin CLI 得到解决。尽管我们不建议在简单的持续交付场景中使用 Spinnaker，但在复杂的情况和同样复杂的部署管道中，它已经成为许多人的首选工具。

© Thoughtworks, Inc. All Rights Reserved.

56. TypeScript OpenAPI

试验

TypeScript OpenAPI （或称 tsoa）是 Swagger 生成 OpenAPI 规范的一个替代方案，用于从代码中直接生成 API 规范。它采用代码优先的方式，将 TypeScript 控制器和模型作为唯一的真实数据来源，并使用 TypeScript 注解或装饰器，而不像使用 OpenAPI 工具时需要复杂的文件和配置。它能够生成 2.0 和 3.0 的 API 规范，并且支持为 Express、Hapi 和 Koa 生成路由。如果你在使用 TypeScript 编写 API，值得看看这个项目。

57. Unleash

试验

尽管我们仍然推荐使用 最简特性开关，但随着团队的扩展和开发速度的加快，管理手工制作的开关变得更加复杂。现在我们团队广泛使用 Unleash，它能够应对这种复杂性并支持 CI/CD。它既可以作为服务使用，也可以自托管。Unleash 提供了多个语言的 SDK，拥有良好的开发者体验和友好的管理界面。尽管目前还没有对 OpenFeature 规范 的官方支持，但你可以找到由社区维护的 Go 和 Java providers。Unleash 既可以用于简单特性开关，也支持分组和渐进式发布，使其成为适合大规模功能管理的选择。

58. Astronomer Cosmos

评估

Astronomer Cosmos 是一个为 Airflow 设计的插件，旨在为 dbt core 工作流提供更原生的支持。安装该插件后，当使用 DbtDag 包装 dbt 工作流时，它将 dbt 节点转换为 Airflow 任务 / 任务组，使工程师能够直接在 Airflow UI 中可视化 dbt 依赖图及其执行进度。它还支持使用 Airflow 连接代替 dbt 配置文件，从而可能减少配置扩散。我们正在试验该工具，探索它在 Airflow 中与 dbt 更无缝集成的潜力。

59. ColPali

评估

ColPali 是一款新兴工具，利用 视觉语言模型 实现 PDF 文档检索，旨在解决从包含图像、图表和表格的多媒体文档中提取数据的难题，这对于构建强大的 检索增强生成（RAG）应用至关重要。与依赖文本嵌入或光学字符识别（OCR）技术的传统方法不同，ColPali 处理整页 PDF 文档，使用视觉 Transformer 创建嵌入，综合考虑文本和视觉内容。这种整体方法不仅提高了文档检索的效果，还增强了对为何检索到特定文档的推理能力，大大提升了 RAG 在数据丰富的 PDF 文档中的表现。我们已经在多个客户项目中测试了 ColPali，结果显示出很大的潜力，但该技术仍处于早期阶段。对于拥有复杂视觉文档数据的组织来说，值得考虑进行评估。

© Thoughtworks, Inc. All Rights Reserved.

60. Cursor

评估

AI 辅助编程工具的竞赛仍在继续，而其中最引人注目的一个就是 Cursor。Cursor 是一个以 AI 为核心的代码编辑器，旨在通过深度整合 AI 到编码工作流中来提升开发者的生产力。虽然我们在之前的雷达评估中已经关注到它，但显然，Cursor 近期的持续改进已为其带来了质的飞跃。在我们的使用中，Cursor 展现了基于现有代码库的强大上下文推理能力。尽管其他 AI 代码工具如 GitHub Copilot 已经可以围绕代码片段进行代码生成或协作，Cursor 的多行和多文件编辑操作让它脱颖而出。Cursor 是基于 VSCode 代码库分叉开发的，提供了一种符合开发者直觉的快速且直观的交互方式。通过快捷键 ctrl/cmd+K 和 ctrl/cmd+L 即可完成强大的操作。Cursor 在 AI 编程工具的竞赛中引领了新一轮的竞争，尤其是在开发者交互和代码库理解方面更为突出。

61. Data Mesh Manager

评估

Data Mesh Manager 提供了典型 data mesh 平台的元数据层。它特别关注数据产品的定义以及使用 OpenContract 倡议规范数据契约，并可以通过相关的 DataContract CLI 集成到构建管道中。该应用还提供了数据目录，用于发现和探索数据产品及其元数据，并允许进行联邦治理，包括定义数据质量指标和管理数据质量规则。作为该领域的首批原生工具之一，它不仅仅是试图将现有平台改造为数据网格范式。

62. GitButler

评估

尽管 Git 的 功能强大且实用，但其命令行界面在管理多个分支和提交暂存方面以复杂性著称。GitButler 是一个 Git 客户端，它提供了图形界面，旨在简化这一过程。GitButler 通过独立于 Git 跟踪未提交的文件更改，并将这些更改暂存到虚拟分支中来实现这一目标。有人可能会认为这是对不应存在的问题的解决方案；如果你经常进行小规模更改并频繁推送到主干，就不需要多个分支。然而，当你的工作流程涉及 pull request（PR）时，分支结构可能会变得复杂，尤其是在 PR 合并之前有较长的审查周期时。为了解决这个问题，GitButler 还与 GitHub 集成，允许你选择性地将更改分组为 pull request，并直接从该工具发出。GitButler 是旨在管理 PR 流程中固有复杂性的工具中又一新增选项。

63. JetBrains AI Assistant

评估

JetBrains AI Assistant 是一款为所有 JetBrains IDE 提供支持的编码助手，旨在顺畅集成以支持代码补全、测试生成和风格指南遵循。它基于 OpenAI 和 Google Gemini 等模型，因其能够记住编码风格并在后续会话中保持一致性输出而脱颖而出。我们的开发人员发现其测试生成功能特别有用，并指出它在处理较长输出时没有稳定性问题。然而，与一些竞争对手不同，JetBrains 没有托管自己的模型，这对于担心第三方数据处理的客户可能并不适用。尽管如此，该工具与 JetBrains IDE 的集成使其成为探索 AI 驱动编码助手的团队的一个有前景的选择。

© Thoughtworks, Inc. All Rights Reserved.

64. Mise

评估

在多语言环境中工作的开发人员经常发现自己需要管理多个不同语言和工具的版本。mise 旨在解决这个问题，提供一个工具来替代 nvm、pyenv、pyenv、rbenv、rustup 等工具，并且可以作为 asdf 的直接替代品。Mise 是用 Rust 编写的，以提高 shell 交互的速度；与使用基于 shell 的 shim 的 asdf 不同，mise 预先修改 PATH 环境变量，从而直接调用工具运行时。这也是 mise 比 asdf 更快的部分原因。对于那些已经熟悉 asdf 的开发人员，mise 提供了相同的功能，但有一些关键区别。由于是用 Rust 编写的，mise 更快，并且拥有一些 asdf 所没有的功能，例如能够同时安装同一工具的多个版本，以及更宽容的命令，包括模糊匹配。它还提供了一个集成的任务运行器，方便执行代码检查、测试、构建、服务器和其他与项目相关的任务。如果您厌倦了使用多个工具来管理开发环境，并且对其他工具有时笨拙的语法感到不满，那么 mise 绝对值得一试。

65. Mockoon

评估

Mockoon 是一个开源的 API 模拟工具。它具有直观的界面、可自定义的路由和动态响应功能，还可以自动创建模拟数据集。Mockoon 兼容 OpenAPI，允许生成不同的场景，能够在本地进行测试并与开发流水线集成。你还可以创建“部分模拟”，通过拦截请求，仅模拟 Mockoon 中定义的调用。这部分模拟有助于模拟特定的 API 路由或端点，并将其他请求转发到实际的服务器。部分模拟在某些场景下非常有用，但存在滥用风险，可能导致不必要的复杂性。除此之外，Mockoon 仍然是快速搭建模拟 API、改进和自动化开发流程的宝贵工具。

66. Raycast

评估

Raycast 是一款适用于 macOS 的启动器，允许你通过键盘快速启动应用程序、运行命令、搜索文件并自动化任务。我们的团队认为其针对开发者的开箱即用功能非常有价值，并且它的扩展非常简单，允许与第三方应用和服务，如 VSCode、Slack、Jira 和 Google 等，进行交互。Raycast 专为提高生产力设计，减少上下文切换，是希望简化日常任务的用户的有用工具。专业版用户还可以使用 Raycast AI，这是一款专门的 AI 驱动搜索助手。

67. ReadySet

评估

ReadySet 是一个适用于 MySQL 和 PostgreSQL 的缓存层。与依赖手动失效的传统缓存解决方案不同，ReadySet 利用数据库复制流来增量更新其缓存。通过 部分视图物化，ReadySet 实现了比传统只读副本更低的尾延迟。ReadySet 与 MySQL 和 PostgreSQL 在协议上兼容，因此可以将其部署在数据库前，以横向扩展读取工作负载，而无需更改应用程序。

© Thoughtworks, Inc. All Rights Reserved.

68. Rspack

评估

许多我们团队在开发基于 Web 的前端时，已经从旧的打包工具（比如 Webpack）转向了 Vite。该领域的新进者是 Rspack，经过 1.5 年的开发，刚刚发布了 1.0 release。Rspack 被设计为 Webpack 的直接替代品，兼容 Webpack 生态系统中的插件和加载器。这在迁移复杂的 Webpack 设置时，相较于 Vite 可能具有一定优势。我们团队迁移到 Vite 和 Rspack 等新工具的主要原因是开发者体验，特别是速度。没有什么比在获取上次代码更改反馈前需要等待一两分钟更能打断开发流程的了。Rspack 使用 Rust 编写，提供的性能显著快于 Webpack，在许多情况下甚至比 Vite 更快。

69. 语义路由

评估

在构建基于 LLM 的应用时，将请求路由到特定代理或触发某一流程之前，确定用户意图是至关重要的。Semantic Router 充当 LLM 和代理之间的快速决策层，基于语义意义进行高效且可靠的请求路由。通过使用向量嵌入推断意图，Semantic Router 减少了不必要的 LLM 调用，提供了一种更加简洁、具有低成本的用户意图理解方式。它的潜力不仅限于意图推断，还可以作为各种语义任务的多功能构建模块。它的响应速度和灵活性使其在需要避免 LLM 带来的额外开销，快速实时决策的环境中成为强有力的竞争者。

70. 软件工程代理（software engineering agents）

评估

目前在生成式 AI（GenAI）领域最热门的话题之一是软件工程代理（software engineering agents）的概念。这些编程辅助工具不仅仅是在代码片段上帮助工程师，它们的目标是扩大解决问题的范围，理想情况下能够自主完成任务，且减少人为干预。其理念是，这些工具能够接收 GitHub issue 或 Jira ticket，提出计划并进行代码更改，甚至创建供人类审查的 pull request。这是提升 AI 编程辅助工具影响力的下一步逻辑，但想要实现覆盖广泛编码任务的通用代理的目标仍然非常雄心勃勃，目前的工具尚未令人信服地展示出这一点。然而，我们认为对于范围较小、较简单的任务，这种工具将很快起到作用，帮助开发人员腾出时间处理更复杂的问题。正在发布和推广测试版代理的工具包括 GitHub Copilot Workspace、qodo flow、Tabnine's 的 JIRA 代理，以及 Amazon Q Developer。SWE Bench 基准测试列出了更多此类工具，但我们建议对 AI 领域的基准测试保持一定的谨慎态度。

71. uv

评估

Rust 因为其启动性能非常适合编写命令行工具，我们看到一些工具链正在用它重写。我们在之前的雷达报告中提到过 Ruff，这是一个用 Rust 编写的 Python linter。在本期报告中，我们评估了 uv，这是一个用 Rust 编写的 Python 包管理工具。uv 的价值主张是“超快”，在基准测试中，它的性能大幅超过其他 Python 包管理工具。然而，在我们的雷达评估中，我们讨论了在构建工具中优化几秒钟是否真的算是一个显著的提升。相比于

© Thoughtworks, Inc. All Rights Reserved.

性能，对于一个包管理系统来说，更重要的是生态系统、成熟的社区和长期的支持。尽管如此，我们项目团队的反馈表明，这一小幅速度提升可能会极大改善反馈周期和整体开发者体验——我们通常手动使 CI/CD 缓存变得非常复杂，以实现这微小的性能提升。uv 简化了我们的 Python 环境管理。考虑到在 Python 开发的包和环境管理方面仍有很大的改进空间，我们认为 uv 是一个值得评估的选择。

72. Warp

评估

Warp 是一款适用于 macOS 和 Linux 的终端工具，它将命令输出分割为块以提高可读性。Warp 提供了 AI 驱动的能力，如智能命令建议和自然语言处理。它还包括笔记本功能，允许用户组织命令和输出，并添加注释和文档。你可以利用这些功能创建 README 文件或入职材料，以结构化和互动的方式呈现和管理终端工作流。Warp 还可以轻松集成 Starship，一个灵活的跨终端提示工具，允许你自定义终端体验，并检索有关正在运行的进程、所使用工具的特定版本、Git 详细信息或当前 Git 用户等信息。

73. Zed

评估

在 Atom 文本编辑器项目关闭后，其创建者构建了一个名为 Zed 的新编辑器。Zed 使用 Rust 编写，并经过优化以充分利用现代硬件，给人感觉非常快速。它具备我们对现代编辑器的所有期望功能：支持多种编程语言、内置终端以及多缓冲编辑等。通过与多个 LLM 提供商的集成，Zed 还提供 AI 辅助编码功能。作为结对编程的热衷者，我们对 Zed 内置的远程 协作特性 很感兴趣。开发者可以通过他们的 GitHub ID 找到彼此，然后实时协作于同一工作区。虽然现在还无法判断开发团队是否能够并愿意摆脱 Visual Studio Code 生态系统的吸引，但 Zed 是一个值得探索的替代方案。

74. CocoaPods

暂缓

CocoaPods 一直是 Swift 和 Objective-C Cocoa 项目中广受欢迎的依赖管理工具。然而，CocoaPods 团队宣布，该项目在作为 iOS 和 macOS 开发者关键工具超过十年后，进入了维护模式。尽管工具和其资源仍将继续可用，但将不再进行主动开发。其鼓励开发者们转向 Swift Package Manager，它与 Xcode 原生集成，并且获得了来自苹果的更好的长期支持。

© Thoughtworks, Inc. All Rights Reserved.

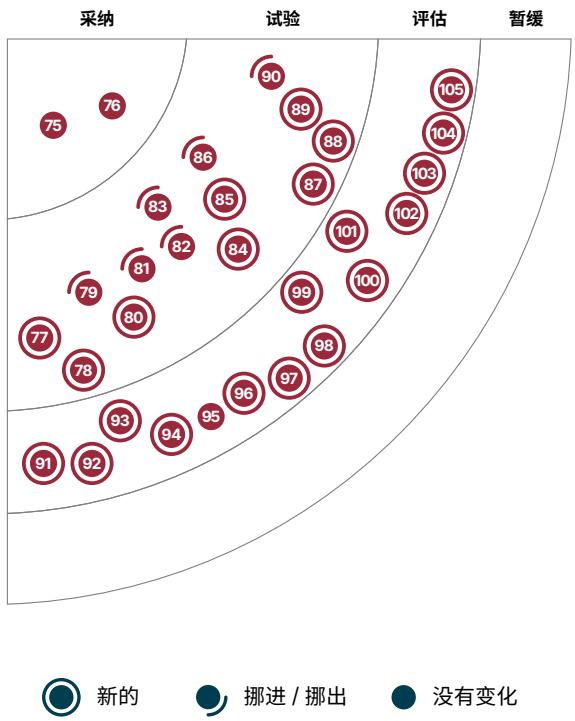
语言和框架

采纳
75. dbt
76. Testcontainers

试验
77. CAP
78. CARLA
79. Databricks Asset Bundles
80. Instructor
81. Kedro
82. LiteLLM
83. LlamaIndex
84. LLM Guardrails
85. Medusa
86. Pkl
87. ROS 2
88. seL4
89. SetFit
90. vLLM

评估
91. Apache XTable ™
92. dbldatagen
93. DeepEval
94. DSPy
95. Flutter for Web
96. kotaemon
97. Lenis
98. LLMLingua
99. Microsoft Autogen
100. Pingora
101. Ragas
102. Score
103. shadcn
104. Slint
105. SST

暂缓
—



75. dbt

采纳

我们仍然认为 dbt 是在 ELT 数据管道中实施数据转换的强大且合理的选择。我们喜欢它能够引入工程上的严谨性，并支持模块化、可测试性和 SQL 转换的可重用性等实践。dbt 集成了 很多云数据仓库、数据湖仓和数据库，包括 Snowflake、BigQuery、Redshift、Databricks 和 Postgres，并且拥有健康的社区包生态系统。最近在 dbt core 1.8+ 和全新的 dbt Cloud 无版本体验中引入的原生单元测试支持，进一步巩固了其在我们工具箱中的地位。我们的团队很欣赏这个新的单元测试功能，它使他们能够轻松定义静态测试数据，设置输出预期，并测试管道的增量和完全刷新模式。在许多情况下，这使得团队能够淘汰自制脚本，同时保持相同的质量水平。

76. Testcontainers

采纳

在我们的经验中，Testcontainers 是创建可靠测试环境的一个有效默认选项。它是一个移植到多种语言的库，可以将常见的测试依赖项进行 Docker 化——包括各种类型的数据库、队列技术、云服务以及像网页浏览器这样的 UI 测试依赖项，并能够在需要时运行自定义 Dockerfile。最近发布了一个 桌面版本，允许对测试会话进行可视化管理，并能够处理更复杂的场景，这对我们的团队非常有用。

77. CAP

试验

CAP 是一个实现了 Outbox 模式 的 .NET 库 在使用 RabbitMQ 或 Kafka 等分布式消息系统时，我们经常面临确保数据库更新和事件发布按原子性执行的挑战。CAP 通过在引发事件的同一个数据库事务中记录事件发布的意图，解决了这一问题。我们发现 CAP 非常实用，它支持多种数据库和消息平台，并确保至少一次的消息投递。

78. CARLA

试验

CARLA 是一个开源的自动驾驶模拟器，它可用于在生产部署之前测试自动驾驶系统。它在创建和重用车辆、地形、人类、动物等 3D 模型上具有很强的灵活性，这使得它可以用来模拟各种场景，比如模拟行人走上马路或遇到迎面而来的特定速度的车辆。那些待测试的自动驾驶系统必须能够识别这些动态参与者并采取适当的行动，例如刹车。我们的团队使用 CARLA 进行自动驾驶系统的持续开发和测试。

© Thoughtworks, Inc. All Rights Reserved.

79. Databricks Asset Bundles

试验

Databricks Asset Bundles（DABs），于 2024 年 4 月 实现基本可用，正逐渐成为打包和部署 Databricks 资产的首选工具，帮助我们的数据团队应用软件工程实践。DABs 支持将工作流和任务的配置以及要在这些任务中执行的代码打包成一个 bundle，并通过 CI/CD 管道部署到多个环境中。它提供了常见资产类型的模板，并支持自定义模板，这使得能够为数据工程和机器学习项目创建 定制服务模版。我们的团队越来越多地将其作为工程工作流中的关键部分。尽管 DABs 包含了针对 notebooks 的模板，并支持将 notebooks 部署到生产环境中，我们并不推荐 生产化 notebooks，而是鼓励有意地编写符合生产标准的代码，以支持此类工作负载的可维护性、可靠性和可扩展性需求。

80. Instructor

试验

当我们作为终端用户使用大语言模型（LLM）聊天机器人时，它们通常会返回给我们一个非结构化的自然语言答案。在构建超越聊天机器人的生成 AI 应用时，请求 LLM 以 JSON、YAML 或其他格式返回结构化答案并在应用中解析和使用该响应可能非常有用。然而，由于 LLM 是非确定性的，它们可能并不总是按照我们要求的方式执行。Instructor 是一个可以帮助我们请求 从 LLMs 获取结构化输出 的库。您可以定义预期的输出结构，并在 LLM 未返回您要求的结构时配置重试。由于与 LLM 进行交互时，最佳的终端用户体验往往是将结果流式传输给他们，而不是等待完整响应，Instructor 还可以处理从流中解析部分结构的任务。

81. Kedro

试验

Kedro 作为 MLOps 工具有了显著的改善，并始终保持对模块化和工程实践的关注，这一点是我们从一开始就非常喜欢的。突出其模块化的一步是推出了独立的 kedro-datasets 包，该包将代码与数据解耦。Kedro 在其命令行接口、起始项目模板和遥测功能方面进行了增强。此外，最近发布的 VS Code 扩展对开发者体验也是一个很好的提升。

82. LiteLLM

试验

LiteLLM 是一个用于无缝集成各种大语言模型（LLM）提供商 API 的库，通过 OpenAI API 格式 交互。它支持多种 提供方和模型，并为文本生成、嵌入和图像生成提供统一的接口。LiteLLM 简化了集成过程，通过匹配每个提供商的特定端点要求来翻译输入。它还提供了实现生产应用中所需的操作功能的框架，如缓存、日志记录、速率限制和负载均衡，从而确保不同 LLM 的一致操作。我们的团队使用 LiteLLM 来轻松切换各种模型，这在模型快速演变的今天尤为必要。然而，值得注意的是，不同模型在相同提示下的响应会有所不同，这表明仅一致的调用方式可能不足以优化生成性能。此外，每个模型对附加功能的实现方式也各不相同，单一接口可能无法满足所有需求。例如，我们的一个团队在通过 LiteLLM 代理 AWS Bedrock 模型时，难以充分利用其函数调用功能。

© Thoughtworks, Inc. All Rights Reserved.

83. LlamaIndex

试验

LLamaIndex 包含能够设计特定领域、上下文增强的 LLM 应用程序的引擎，并支持数据摄取、向量索引和文档上的自然语言问答等任务。我们的团队使用 LlamaIndex 构建了一个 检索增强生成（RAG）流水线，自动化文档摄取，索引文档嵌入，并根据用户输入查询这些嵌入。使用 LlamaHub，您可以扩展或自定义 LlamaIndex 模块以满足您的需求，例如构建使用您首选的 LLM、嵌入和向量存储提供者的 LLM 应用程序。

84. LLM Guardrails

试验

LLM Guardrails 是一套用于防止大语言模型（LLMs）生成有害、使人误解或不相关内容的指南、政策或过滤器。Guardrails 也可用于保护 LLM 应用免受恶意用户通过操纵输入等技术对其滥用。它们通过为模型设定边界来作为安全网，确保内容的处理和生成在可控范围内。在这一领域中，诸如 NeMo Guardrails 、Guardrails AI 和 Aporia Guardrails 等框架已经逐渐崭露头角，并被我们的团队认为非常有用。我们建议每个 LLM 应用都应设置相应的安全护栏，并且不断改进其规则和政策。这对于构建负责任和值得信赖的 LLM 聊天应用至关重要。

85. Medusa

试验

根据我们的经验，大多数用于构建购物网站的电子商务解决方案通常会陷入 80/20 陷阱，即我们可以轻松构建出 80% 的需求，但对于剩下的 20% 却无能为力。Medusa 提供了一个良好的平衡。它是一个高度可定制的开源商业平台，允许开发人员创建独特且量身定制的购物体验，可以自我托管或运行在 Medusa 的平台上。Medusa 基于 Next.js 和 PostgreSQL 构建，通过提供从基本购物车和订单管理到高级功能（如礼品卡模块和不同地区的税收计算）的全面模块，加快了开发过程。我们发现 Medusa 是一个有价值的框架，并在几个项目中应用了它。

86. Pkl

试验

Pkl 是一种开源的配置语言及工具，最初由苹果公司内部使用而创建。它的主要特点是其类型和验证系统，能够在部署之前捕捉配置错误。Pkl 帮助我们的团队减少了代码重复（例如环境覆盖的情况），并且能够在配置更改应用到生产环境之前进行验证。它可以生成 JSON、PLIST、YAML 和 .properties 文件，并且具有包括代码生成在内的广泛集成开发环境（IDE）和语言支持。

© Thoughtworks, Inc. All Rights Reserved.

87. ROS 2

试验

ROS 2 是一个为开发机器人系统设计的开源框架。它提供了一套用于模块化实现应用程序的库和工具，涵盖了诸如进程间通信、多线程执行和服务质量等功能。ROS 2 在其 前任 的基础上进行了改进，提供了更好的实时性能、更高的模块化、对多种平台的支持以及合理的默认设置。ROS 2 在汽车行业正在获得越来越多的关注；它基于节点的架构和基于主题的通信模型，特别适合那些具有复杂且不断发展的车载应用（如自动驾驶功能）的制造商。

88. seL4

试验

在软件定义汽车 （SDV）或其他安全关键场景中，操作系统的实时稳定性至关重要。由于该领域的高准入门槛，少数公司垄断了这一领域，因此像 seL4 这样的开源解决方案显得尤为珍贵。seL4 是一个高保障、高性能的操作系统微内核。它使用 形式化验证 方法来“数学上”确保操作系统的行为符合规范。其微内核架构还将核心职责最小化，以确保系统的稳定性。我们已经看到像蔚来汽车（NIO）这样的电动汽车公司参与 seL4 生态系统，未来在这一领域可能会有更多的发展。

89. SetFit

试验

当前大多数基于 AI 的工具都是生成式的——它们生成文本和图像，使用生成式预训练模型（GPTs）来完成这些任务。而对于需要处理现有文本的用例——例如文本分类或意图识别——sentence transformers 是首选工具。在这个领域，SetFit 是一个用于微调 sentence transformers 的框架。我们喜欢 SetFit 的原因是它使用对比学习来区分不同的意图类别，通常只需非常少量的样本（甚至少于 25 个）就能实现清晰的分类。sentence transformers 在生成式 AI 系统中也可以发挥作用。我们成功地在一个使用大语言模型（LLM）的客户聊天机器人系统中使用 SetFit 进行意图检测。尽管我们知晓 OpenAI 的内容审核 API，我们仍选择基于 SetFit 的分类器进行额外的微调，以实现更严格的过滤。

90. vLLM

试验

vLLM 是一个高吞吐量、内存高效的 LLM 推理引擎，既可以在云环境中运行，也可以在本地部署。它无缝支持多种 模型架构 和流行的开源模型。我们的团队在 NVIDIA DGX 和 Intel HPC 等 GPU 平台上部署了容器化的 vLLM 工作节点，托管模型如 Llama 3.1（8B and 70B）、Mistral 7B 和 Llama-SQL，用于开发者编码辅助、知识搜索和自然语言数据库交互。vLLM 兼容 OpenAI SDK 标准，促进了一致的模型服务。Azure 的 AI 模型目录 使用自定义推理容器来提升模型服务性能，vLLM 由于其高吞吐量和高效的内存管理，成为默认的推理引擎。vLLM 框架正在成为大规模模型部署的默认选择。

© Thoughtworks, Inc. All Rights Reserved.

91. Apache XTable™

评估

在可用的开放表格格式中，能够支持 数据湖仓一体（lakehouses）架构 ——例如 Apache Iceberg、Delta 和 Hudi ——尚未出现明显的赢家。相反，我们看到了一些工具正在促进这些格式之间的互操作性。例如，Delta UniForm 通过允许 Hudi 和 Iceberg 客户端读取 Delta 表，实现了单向互操作性。另一个新进入这个领域的是 Apache XTable ™，这是一个 Apache 孵化器项目，旨在实现 Hudi、Delta 和 Iceberg 之间的全向互操作性。与 UniForm 类似，XTable 在不创建底层数据副本的情况下，能够在这些格式之间转换元数据。XTable 对于那些在多个表格格式之间进行实验的团队可能会很有用。然而，考虑到这些格式之间功能的差异，长期依赖全向互操作性可能会导致团队只能使用功能的“最小公倍数”。

92. dbldatagen

评估

为数据工程准备测试数据是一个重大挑战。从生产环境转移数据到测试环境存在风险，因此团队通常依赖于编造数据或合成数据。在本期雷达中，我们探讨了诸如 通过大模型生成合成数据 等新方法。但大多数情况下，成本较低的程序生成已经足够用。dbldatagen （Databricks Labs Data Generator）就是这样一个工具；它是一个用于在 Databricks 环境中生成合成数据的 Python 库，适用于测试、基准测试、演示等多种用途。dbldatagen 可以在短时间内生成规模达数十亿行的合成数据，支持多表、变更数据捕获和合并 / 连接操作等各种场景。它能够很好地处理 Spark SQL 的基本类型，生成范围和离散值，并应用指定的分布。在 Databricks 生态系统中创建合成数据时，dbldatagen 是一个值得评估的选项。

93. DeepEval

评估

DeepEval 是一个基于 Python 的开源评估框架，用于评估大语言模型（LLM）的性能。你可以使用它评估使用流行框架（如 LlamaIndex 或 LangChain 构建的检索增强生成（RAG）和其他类型的应用程序，也可以用于基准测试和对比不同模型，以满足你的需求。DeepEval 提供了一个全面的指标和功能套件，用于评估 LLM 的表现，包括幻觉检测、答案相关性和超参数优化。它支持与 pytest 的集成，结合其断言功能，你可以轻松地将测试套件集成到持续集成（CI）管道中。如果你正在使用 LLM，建议尝试 DeepEval 来改进测试流程，确保你的应用程序的可靠性。

© Thoughtworks, Inc. All Rights Reserved.

94. DSPy

评估

如今，大多数基于语言模型的应用程序依赖于为特定任务手工调整的提示词模板。DSPy 是一个用于开发此类应用程序的框架，采用了一种不同的方式，摒弃了直接的提示词工程。相反，它引入了围绕程序流程的更高级抽象（通过可以彼此叠加的 modules），以及优化的指标和用于训练 / 测试的数据。生成的代码库类似于使用 PyTorch 进行神经网络训练。我们认为这种方法很有新意，值得尝试。

95. Flutter for Web

评估

Flutter 以其对 iOS 和 Android 应用程序的跨平台支持而闻名。现在，它已经扩展到更多的平台。我们之前评估过 Flutter for Web — 它允许我们从同一代码库构建适用于 iOS、Android 和浏览器的应用程序。并不是每个网页应用都适合使用 Flutter，但我们认为 Flutter 特别适合诸如 progressive web apps、single-page apps 和将现有 Flutter 移动应用转换为网页的情况。Flutter 已经支持相反，它引入了围绕程序流程的更高级抽象（通过可以彼此叠加的 modules），以及优化的指标和用于训练 / 测试的数据。WebAssembly（WASM）作为其实验性频道中的编译目标，这意味着它正在积极开发中。最近的版本已使其潜在漏洞和性能风险较为稳定。编译到 WASM 目标的 Flutter 网页应用的性能远超其 JavaScript 编译目标。不同平台上的近原生性能也是许多开发者最初选择 Flutter 的原因之一。

96. kotaemon

评估

kotaemon 是一个基于检索增强生成（RAG）的开源工具和框架，用于构建针对知识库文档的问答应用程序。它可以理解多种文档格式，包括 PDF 和 DOC，并提供基于 Gradio 的 Web 用户界面，用户可以通过聊天界面组织和互动知识库。kotaemon 具有内置的 RAG 流水线，并集成了向量存储，且可以通过 SDK 进行扩展。它的回答中还引用了来源文档，提供网页内联预览和相关性评分。对于想要构建基于 RAG 的文档问答应用程序的用户来说，这个可定制的框架是一个非常好的起点。

97. Lenis

评估

Lenis 是一个为现代浏览器设计的轻量且强大的平滑滚动库。 它能够实现流畅的滚动体验，例如 WebGL 滚动同步和视差效果，这使其非常适合那些需要构建流畅、无缝滚动交互页面的团队。我们的开发人员发现 Lenis 的使用非常简单，它提供了一种精简高效的方式来创建平滑滚动效果。然而，该库在无障碍性方面可能存在一些问题，特别是在处理垂直和水平滚动交互时，可能会让残障用户感到困惑。虽然它在视觉上很吸引人，但在实现时仍然需要仔细考虑无障碍性。

© Thoughtworks, Inc. All Rights Reserved.

98. LLMLingua

评估

LLMLingua 通过使用小型语言模型压缩提示，去除非必要的 token，从而提高大语言模型（LLM）的效率，并在性能损失最小的情况下实现这一目标。 这种 方法 使大语言模型（LLM）能够在有效处理较长提示的同时，保持推理和上下文学习能力，解决了成本效率、推理延迟和上下文处理等挑战。LLMLingua 与各种大语言模型兼容，无需额外训练，并支持如 LLamaIndex 等框架，它非常适合优化大语言模型的推理性能。

99. Microsoft Autogen

评估

Microsoft Autogen 是一个开源框架，旨在简化 AI 代理的创建和编排，支持多个代理协作解决复杂任务。它支持自动化和人机协作的工作流程，并兼容多种大语言模型（LLMs）和代理交互工具。我们的一个团队曾在为客户构建的 AI 驱动平台中使用 Autogen，每个代理代表一种特定技能，如代码生成、代码审查或文档摘要。该框架使团队能够通过定义正确的模型和工作流程，轻松且一致地创建新代理。他们还利用 LlamaIndex 编排工作流程，使代理能够有效管理产品搜索和代码建议等任务。尽管 Autogen 在生产环境中展示了潜力，尤其在代理的扩展性和复杂性管理上仍有待评估，以确保其在扩展代理系统时的长期可行性。

100. Pingora

评估

Pingora 是一个 Rust 搭建框架，旨在构建快速可靠可编程的网络服务。它最开始由 Cloudflare 开发用以 解决 Nginx 的不足，Pingora 已展现出巨大的潜力，像 River 这样的新型代理正是以 Pingora 为基础构建的。虽然我们大多数人无需面对 Cloudflare 级别的规模，但我们确实会遇到需要灵活的应用层路由的网络服务的场景。Pingora 的架构可以让我们在这些场景下充分利用 Rust 的强大能力，同时不牺牲安全性或性能。

101. Ragas

评估

Ragas 是一个框架，旨在评估 检索增强生成（RAG） 流水线的性能，解决了评估这些系统中检索和生成组件的挑战。它提供了结构化的指标，如可靠性、答案相关性和上下文利用率，这些指标有助于评估基于 RAG 系统的有效性。我们的开发者发现，它在运行定期评估以微调参数（如 top-k 检索和嵌入模型）时非常有用。一些团队将 Ragas 集成到每天运行的流水线中，以便在提示模板或模型发生变化时进行评估。虽然它的指标提供了可靠的见解，但我们担心该框架可能无法捕捉复杂 RAG 流水线的所有细微差别和复杂交互，因此建议考虑额外的评估框架。尽管如此，Ragas 在生产环境中简化 RAG 评估的能力使其脱颖而出，为数据驱动的改进提供了宝贵的支持。

© Thoughtworks, Inc. All Rights Reserved.

102. Score

评估

许多实施自己内部开发平台的组织倾向于创建自己的 平台编排 系统，以在开发人员与其平台托管团队之间强制执行组织标准。然而，针对安全、一致和合规地托管容器工作负载的铺路平台的基本功能在不同组织之间是相似的。如果我们有一种共享语言来指定这些要求，那该多好呢？ Score 在这一领域显示出了成为标准的潜力。它是一种以 YAML 形式编写的声明性语言，描述了容器化工作负载的部署方式，以及其运行所需的特定服务和参数。Score 最初由 Humanitec 开发，作为其 平台编排者 产品的配置语言，但现在作为一个开源项目由云原生计算基金会 （CNCF）进行管理。在 CNCF 的支持下，Score 有潜力在 Humanitec 产品之外得到更广泛的使用。它已经发布了两个参考实现：Kubernetes 和 Docker Compose。Score 的可扩展性希望能够促使社区对其他平台的贡献。Score 确实与 Kubevela 的 开放应用模型 （OAM）规范相似，但它更专注于容器工作负载的部署，而不是整个应用程序。与 SST 也有一些重叠，但 SSI 更关注直接部署到云基础设施，而不是内部工程平台。我们对 Score 的发展保持关注。

103. shadcn

评估

shadcn 通过提供可重用的、可直接复制粘贴的组件，挑战了传统组件库的概念。这种方法让团队拥有完整的控制权和所有权，更加容易进行定制和扩展，而这是更受欢迎的传统库如 MUI 和 Chakra UI 往往不足的地方。shadcn 使用 Radix UI 和 Tailwind CSS 构建，能够无缝集成到任何基于 React 的应用程序中，非常适合优先考虑控制和可扩展性的项目。它还提供一个 CLI，帮助将组件复制并粘贴到项目中。其优点还包括减少隐藏依赖关系，避免紧耦合的实现，因此对于寻求更加自主且可适应的前端开发方法的团队来说，shadcn 正在成为一个引人注目的替代方案。

104. Slint

评估

Slint 是一个声明式的 GUI 框架，用于为 Rust、C++ 或 JavaScript 应用程序构建原生用户界面。尽管它是一个多平台的 UI 框架，拥有实时预览、响应式 UI 设计、VS Code 集成 和原生用户体验等重要特性，但我们特别想强调它在嵌入式系统中的实用性。开发嵌入式应用程序的团队经常会面临有限的 UI 开发选项，并且每个选项都有其权衡之处。Slint 在开发者体验和性能之间提供了完美的平衡，它使用类似 HTML 的易用标记语言，并可以直接编译为机器代码。在运行时，它还具有低资源占用的优势，这对于嵌入式系统至关重要。简而言之，我们喜欢 Slint，因为它将 web 和移动开发中经过验证的实践引入到了嵌入式生态系统中。

© Thoughtworks, Inc. All Rights Reserved.

105. SST

评估

SST 是一个框架，用于将应用程序部署到云环境中，并同时配置应用程序运行所需的所有服务。SST 不仅是一个 基础设施即代码 工具，它还是一个提供 TypeScript API 的框架，可以定义应用程序环境，在 Git 推送时触发应用程序部署服务，还配备了一个 GUI 控制台，用于管理生成的应用程序并调用 SST 管理功能。虽然 SST 最初是基于 AWS Cloud Formation 和 CDK，但其最新版本已经在 Terraform 和 Pulumi 之上实现，因此理论上它是云平台无关的。SST 原生支持部署几种标准的 Web 应用程序框架，包括 Next.js 和 Remix，但也支持无界面 API 应用程序。SST 似乎处于一个独特的类别中。虽然它与 平台编排 工具如 Kubevela 有一些相似之处，但它还为开发者提供了便利功能，如实时模式，将 AWS Lambda 调用代理回开发者本地运行的函数。目前，SST 仍然有些新颖，但随着它的演进，这个项目及其工具类别值得持续关注。

© Thoughtworks, Inc. All Rights Reserved.

想要了解技术雷达最新的新闻和洞见?

点击订阅，以接收每两个月一次、来自 Thoughtworks 的技术洞察和未来趋势探索邮件。

现在订阅

Thoughtworks 是一家全球性软件及技术咨询公司，集战略、设计和工程技术咨询服务于一体，致力于推动数字创新。我们在 19 个国家 / 地区的 48 个办公室拥有超过 10,500 名员工。 在过去的 30 年里，我们为全球各地的众多合作伙伴倾力服务，与客户一起创造了非凡的影响力，帮助他们以技术为优势解决复杂的业务问题。

/thoughtworks

Strategy. Design. Engineering.