/thoughtworks

Strategy. Design. Engineering.

第 29 期 | 2023 年 9 月

技术雷达

针对当今科技领域发展的前沿指南

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。

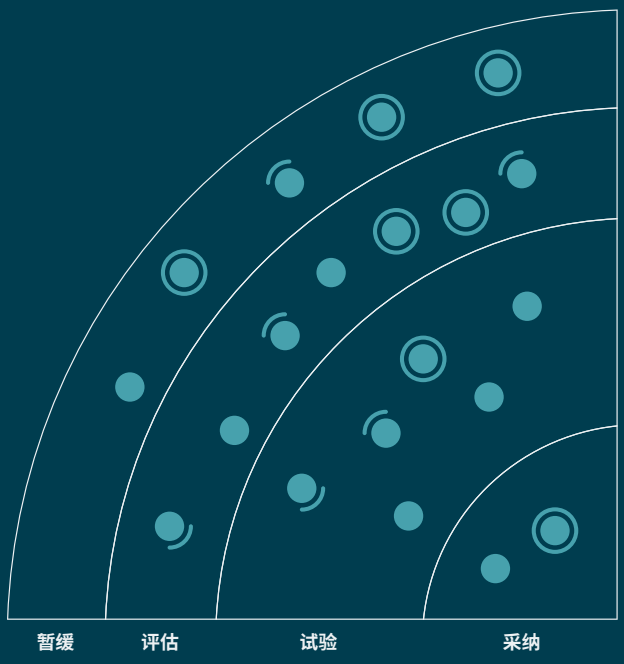
我们建议您探索雷达中提到的内容以了解更多细节。技术雷达的本质是图形性质，把各种技术项目归类为技术、工具、平台和语言和框架。如果技术可以被归类到多个象限，我们选择看起来最合适的一个。我们还进一步将这些技术分为四个环以反映我们目前对其的态度。

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

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。

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 2023 年 8 月的 TAB 线上会议创建。


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


Ian Cartwright


James Lewis


Marisa Hoenig


Maya Ormaza


Mike Mason

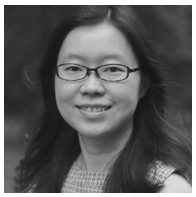

Neal Ford


Pawan Shah


Scott Shaw


Selvakumar Natesan


Shangqi Liu


Sofia Tania


Vanya Seth

中国区技术雷达汉化组：

边晓琳、陈亮、程显通、樊田、樊卓文、冯炜、符雨菡、高晓、管英杰、何蜜、何蔚、何向东、纪扬、郑李鹏、蒋亦雄、李辉、李天舒、李妍、李昱桦、林晨、刘钊、秦睿、孙郁俨、童圣、王鹏熹、王芹芹、熊晓荟、杨琛、杨阳、姚瑶、于海源、余琦、余亚彬、院阳、张丽、张旭海、赵泽鑫、郑茗蔓

本期主题

AI 辅助软件开发

毫无意外，本期技术雷达主要围绕 AI 相关话题展开讨论。这是有史以来第一次，我们需要一个可视化指南来理清不同 AI 的类别和功能（即使在 JavaScript 生态系统十分混乱的时期，我们也从未采取过这样的做法）。作为一家开创 CI、CD 等突破性工程实践历史的软件咨询公司，我们对于使用 AI 辅助软件开发特别感兴趣。因此，本期技术雷达讨论了许多代码辅助工具，如 GitHub Copilot、Tabnine 和 Codeium。我们兴奋于 open-source LLMs for coding 在工具领域可能带来的变革，并且我们看到了在编码之外的辅助领域中工具和能力的爆炸式增长，如用户故事编写辅助、用户研究、电梯演讲和其他基于语言的任务。同时，我们希望开发人员能够负责任地使用所有这些工具，并且始终掌控主导权，比如 hallucinated dependencies 就是其中一个需要注意的安全和质量风险。

衡量生产力有多有效

对于非技术人员来说，软件开发有时似乎很神奇，这导致管理者需要努力衡量开发人员在完成其神秘任务时的生产效率。我们的首席科学家 Martin Fowler 早在 2003 年就撰写了有关此主题的文章，但问题并没有消失。在这期雷达中，我们讨论了许多现代工具和技术，它们采用更加细致入微的方法来衡量软件的创造过程，但这仍然不够。幸运的是，业界已经不再使用代码行数作为产出衡量标准。然而，衡量框架 SPACE 中 A（Activity，活动）的替代方法，例如拉取请求的数量或已解决的问题的数量，仍然不足以成为衡量生产力的良好指标。相反，行业已经开始关注“工程效能”：我们不应该衡量生产力，而应该衡量我们知道对流程有贡献或有损害的事物。我们不应该专注于个体的活动，而应该关注系统中的浪费来源以及可以从经验上证明导致开发人员对“生产力”感知产生影响的条件。新的工具，比如 DX DevEx 360，通过关注开发者体验而不是一些虚假的产出衡量标准解决了这个问题。然而，许多领导人仍然以模糊的、定性的方式衡量开发者的“生产力”。我们怀疑，这种兴趣的复苏至少有一部分原因是受到了人工智能辅助软件开发的影响，这不可避免地引发了一个问题：它是否产生了积极的影响？虽然衡量标准可能变得更加细致入微，但真正的生产力衡量仍然难以捉摸。

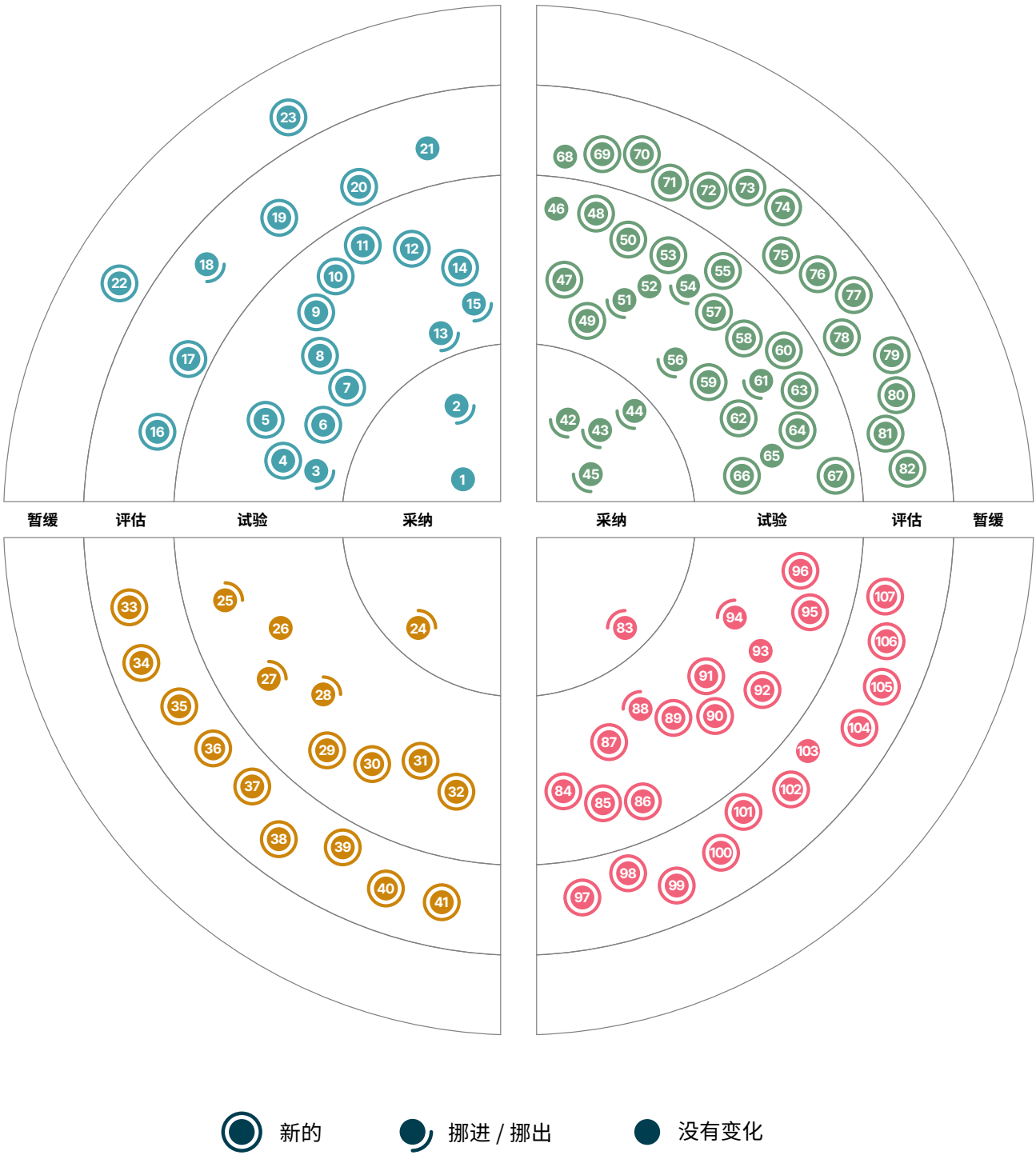
众多大语言模型

大语言模型（LLMs）为现今人工智能的许多重要突破奠定了基础。目前的应用多使用类似聊天的界面进行交互，例如 ChatGPT 或 Google Bard。生态中的主要竞争者（例如 OpenAI 的 ChatGPT，Google Bard，Meta 的 LLaMA 以及亚马逊的 Bedrock 等）在我们的讨论中占据重要地位。更广泛来说，大语言模型可以应用于从内容生成（文本、图片和视频）、代码生成到总结概述和翻译等各种问题。通过自然语言的抽象层，这些大模型成为了强大的工具库，被诸多信息工作者广泛使用。我们讨论了大语言模型的各个方面，包括自托管式大语言模型，相较云托管的大语言模型，它支持更多的定制和管控。随着大语言模型日益复杂，我们正在深思如何在小型设备上运行大语言模型，特别是在边缘设备和资源受限的环境中。我们还提到有望提高性能的 ReAct 提示工程，以及利用大语言模型驱动的自主代理开发远超简单的问答交互的动态应用。我们也提到一些向量数据库（包括 Pinecone）由于大语言模型而重新流行起来。大语言模型的底层能力，包括更专业化和自行托管的能力，将继续呈爆发性增长。

远程交付解决方案日臻成熟

尽管远程软件开发团队多年来利用技术克服地理限制，但疫情的影响进一步推动了这一领域的创新，巩固了向完全远程或混合工作演进的趋势。在本期技术雷达中，我们讨论了远程软件开发实践和工具的成熟，和团队们如何继续以有效协作为重点，不断突破界限，在一个更加分散和动态的环境中进行工作。一些团队利用新的协作工具不断提出创新解决方案。其他团队则继续调整和改进现有的面对面实践，例如实时结对编程或集体编程、分布式工作坊（例如 远程事件风暴）以及异步和同步沟通。远程工作提供了许多好处（包括更多样化的人才储备），但面对面交流的价值是显而易见的。团队不应中断重要的反馈循环，并且需要意识到在转向远程工作时所做的取舍。

本期雷达



本期雷达

技术

采纳

1. 设计系统
2. 轻量级的 RFCs 方法

试验

3. 具有可访问性意识的组件测试设计
4. 攻击路径分析
5. 自动合并依赖项更新 PR
6. 针对 FAIR 数据的数据产品思维
7. OIDC for GitHub Actions
8. 使用 Terraform 创建监控和告警
9. ReAct 提示工程
10. 检索增强生成
11. 基于风险的故障建模
12. 大语言模型半结构化自然语言输入
13. 追踪健康债务状况
14. 对告警规则的单元测试
15. CI/CD 的零信任保护

评估

16. 通过依赖健康检查化解包幻觉风险
17. 设计系统决策记录
18. GitOps
19. 大语言模型驱动的自主代理
20. 平台编排
21. 自托管式大语言模型

暂缓

22. 忽略 OWASP 十大安全风险榜单
23. 用于服务端渲染（SSR）web 应用的 web 组件

平台

采纳

24. Colima

试验

25. CloudEvents
26. DataOps.live
27. Google Cloud Vertex AI
28. Immuta
29. Lokalise
30. Orca
31. Trino
32. Wiz

评估

33. ActivityPub
34. Azure Container Apps
35. Azure OpenAI Service
36. ChatGLM
37. Chroma
38. Kraftful
39. pgvector
40. Pinecone
41. wazero

暂缓

—

工具

采纳

42. dbt
43. Mermaid
44. Ruff
45. Snyk

试验

46. AWS Control Tower
47. Bloc
48. cdk-nag
49. Checkov
50. Chromatic
51. Cilium
52. 云服务的碳足迹
53. 容器结构测试
54. Devbox
55. DX DevEx 360
56. GitHub Copilot
57. Insomnia
58. IntelliJ HTTP 客户端插件
59. KEDA
60. Kubeconform
61. mob
62. MobSF
63. Mocks Server
64. Prisma 运行时防护
65. Terratest
66. Thanos
67. Yalc

评估

68. ChatGPT
69. Codeium
70. GitHub 合并队列
71. Google Bard
72. Google Cloud 工作站
73. Gradio
74. KWOK
75. Llama 2
76. Maestro
77. Open-source LLMs for coding
78. OpenCost
79. OpenRewrite
80. OrbStack
81. Pixie
82. Tabnine

暂缓

—

语言和框架

采纳

83. Playwright

试验

84. .NET Minimal API
85. Ajv
86. Armeria
87. AWS SAM
88. Dart
89. fast-check
90. Kotlin with Spring
91. Mockery
92. Netflix DGS
93. OpenTelemetry
94. Polars
95. Pushpin
96. Snowpark

评估

97. 基准配置文件
98. GGML
99. GPTCache
100. 语法性别 API
101. htmx
102. Kotlin Kover
103. LangChain
104. LlamaIndex
105. promptfoo
106. Semantic Kernel
107. Spring Modulith

暂缓

—

技术

采纳

1. 设计系统
2. 轻量级的 RFCs 方法

试验

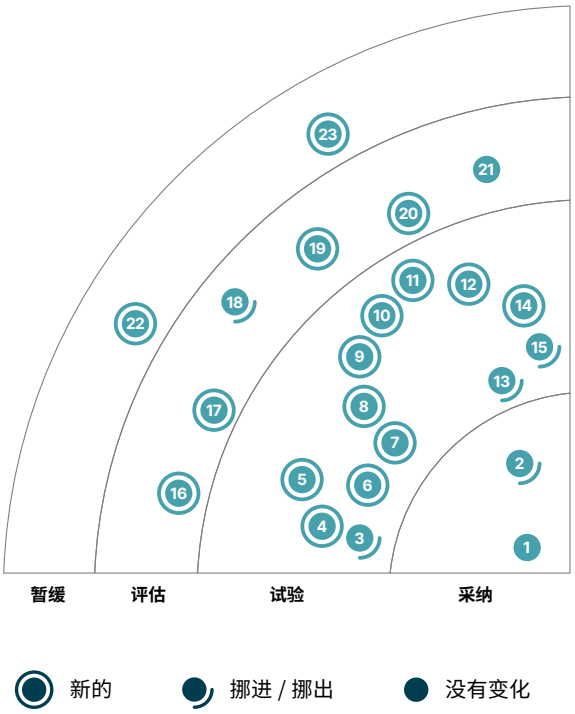
3. 具有可访问性意识的组件测试设计
4. 攻击路径分析
5. 自动合并依赖项更新 PR
6. 针对 FAIR 数据的数据产品思维
7. OIDC for GitHub Actions
8. 使用 Terraform 创建监控和告警
9. ReAct 提示工程
10. 检索增强生成
11. 基于风险的故障建模
12. 大语言模型半结构化自然语言输入
13. 追踪健康债务状况
14. 对告警规则的单元测试
15. CI/CD 的零信任保护

评估

16. 通过依赖健康检查化解包幻觉风险
17. 设计系统决策记录
18. GitOps
19. 大语言模型驱动的自主代理
20. 平台编排
21. 自托管式大语言模型

暂缓

22. 忽略 OWASP 十大安全风险榜单
23. 用于服务端渲染（SSR）web 应用的 web 组件



1. 设计系统

采纳

随着应用开发变得越来越动态和复杂，交付风格一致且好用的产品成为了一项挑战，尤其是在有多个团队参与不同产品开发的大型组织中。设计系统定义了一系列的设计模式、组件库以及良好的设计和工程实践，以确保数字产品的一致性。设计系统从过去的企业风格指南演变而来，提供易于查找和使用的共享组件库和文档。通常，设计系统的风格指南以代码的形式记录并进行版本控制，比简单的文档记录更加清晰且易于维护。设计系统已经成为跨团队和学科进行产品开发时的标准方法，每当需要新的视觉组件时，团队不用重新发明轮子，因此能够集中精力，专注解决产品本身的种种挑战。

我们的经验表明，团队在构建设计系统时很少采用产品为中心的思维方式。共享组件库和文档的主要消费者是产品开发团队。在使用产品为中心的思维方式时，设计系统所有者应该与消费者（开发团队）合作，建立共情。我们发现，许多组件库之所以受到批评，是因为所有者团队无法快速响应消费者的需求，并且无法接受来自外部的贡献。产品为中心的思维方式还要求组织思考是否应该允许和怎样向设计系统做出贡献，以及如何管理这些贡献——在这个话题上，我们推荐采用设计系统决策记录。对我们来说，维护一个良好的设计系统或组件库不光是技术工作，也同样是社交工作。

2. 轻量级的 RFCs 方法

采纳

Request for Comments（RFC）是一种正式文档，其包含与上下文相关的设计和架构思想，以促进团队协作和决策。几乎所有数字原生和快速扩张的组织都使用 RFCs 来记录围绕设计、架构、技术和团队协作方式的决策。成熟的组织已经在自治团队中，特别是在跨团队相关的决策中使用 RFCs 来推动更好的沟通和协作。它通常被用作架构决策记录的审查和批准过程。即让受决策影响的人有机会在决策获得批准之前，参与讨论并提供意见，这一个透明的协作过程。快节奏的环境往往会导致设计决策的推理过程丢失，从而让负责实施决策的团队感到困惑。RFCs 提供了一个决策审计记录，有利于未来的团队成员查看，与此同时记录了组织技术和业务的演进过程。RFCs 可以成为促进演化架构的宝贵工具。不过，为了获得最佳效果，我们建议组织采用轻量级的 RFCs 方法。如果不限定范围并明确要点，这些文件往往会随着时间的推移而变得越来越长，类似于传统的解决方案架构文件一样最终被归档和遗忘。

3. 具有可访问性意识的组件测试设计

试验

在软件交付进程中，可访问性要求是 Web 组件测试阶段的一种考察指标。尽管诸如 chai-a11y-axe 的测试框架插件 API 已提供了基础的可访问性断言，具有可访问性意识的组件测试设计依然能够帮助测试进一步检验屏幕阅读器和其他辅助技术所需的全量语义元素。

首先，在测试验证元素时，通过 ARIA 角色或者元素的其它语义化属性查找元素，而不采用元素的 test id 或 class 属性。像 Testing Library 的一些测试库甚至已经在文档中推荐了这一实践。其次，不要仅仅测试点击交互，还要考虑不能使用鼠标或看不到屏幕的人，并考虑增加针对键盘和其他交互方式的额外测试。在我们的团队中，上述测试设计实践已十分成熟，并且我们已在不久前将其纳入测试闭环中。

4. 攻击路径分析

试验

攻击路径分析是一种分析和评估潜在攻击路径的安全分析方式，黑客可能按照这些来自组织内系统网络的潜在攻击路径进行攻击。此前的多数安全分析策略或工具主要聚焦在特定分线领域，例如错误的配置，脆弱的容器，和常见漏洞上。这些孤立的方法意味着团队们不能看到这些风险与技术栈上其他层的弱点组合产生的危险攻击路径。尽管这一技术已提出一段时间，但是近期安全分析工具的进展能使安全团队更易使用这项技术。Orca 和 Wiz 是两个此类工具。我们建议管理复杂基础设施的团队在为组织设计安全策略或选择安全分析工具时考虑这项技术。

5. 自动合并依赖项更新 PR

试验

软件供应链的复杂性是一个重大风险，我们已经在一些文章中进行过讨论，例如 SBOM 与 SLSA。对于大多数团队来说，致命弱点仍然是依赖项中存在漏洞，通常是来自于多层的间接依赖项。Dependabot 等工具可以通过创建拉取请求（PR）来更新依赖项。不过，团队仍然需要制定工程纪律，以确保及时处理这些 PR，尤其是对长时间不活跃的应用程序或服务提交的 PR。

如果系统具有广泛的测试覆盖范围——不仅有完善的单元测试，还包括有功能和性能测试，并且构建流水线必须运行所有这些测试以及安全扫描，我们更提倡自动合并依赖项更新 PR。 简而言之，团队必须完全相信，流水线运行成功后，软件就可以投入生产。在这种情况下，依赖项更新 PR，即使它们在间接依赖项中包含主要版本更新，也应该自动合并。

6. 针对 FAIR 数据的数据产品思维

试验

数据产品思维重视将数据消费者视为客户，确保他们在数据价值链中的无缝体验。这包括易用的数据发现、理解、信任、访问和消费。“产品思维”不是一个新概念，过去我们在运维中施用了这一概念，建立了运维产品和微服务。它伴随着构建长期的跨功能的团队在组织中拥有并分享他们的数据，通过结合数据和产品思维，我们相信组织能够使用 FAIR（可发现，可访问，可互通且可复用）原则进行数据运营。我们的团队使用如 Collibra 和 DataHub 的数据目录实现数据产品的可发现性，为了建立信任，我们发布数据质量和服务等级指标，比如数据产品的及时性、完整性和一致性，并使用 Soda Core 和 Great Expectations 等工具自动化数据质量检查。 数据可观测性可同时通过 Monte Carlo 等平台实现。

我们已经看到数据产品随着时间的推移，演变为多个用例的可重用构建块。随着我们在识别和构建价值驱动的数据产品上的进展，后续用例的上市时间也随之加快。因此，我们的建议是，采纳针对 FAIR 数据的数据产品思维。

7. OIDC for GitHub Actions

试验

推荐实现 CI/CD 的零信任安全的技术之一是通过使用 OpenID Connect（OIDC）等联合身份机制对流水线进行身份验证，以访问云服务。这一重要的技术仍未被充分利用在 GitHub Actions 中，因此推荐 OIDC for GitHub Actions。通过这种方式，可以避免存储长期的访问令牌来访问云资源，同时确保流水线无法直接访问机密信息。然而，请务必谨慎地限制访问权限，以确保操作以最低权限运行。

8. 使用 Terraform 创建监控和告警

试验

基础设施及代码（IaC）已经是一种被广泛采纳用于定义和创建托管环境的方法。尽管这个领域的工具和技术不断发展，但 Terraform 仍然是 IaC 方式管理云原生资源的主要工具。然而，当下大多数托管环境都是云供应商原生服务、第三方服务和自定义代码的复杂组合。在这些环境中，我们发现工程师通常会使用 Terraform 处理云资源，又使用自定义脚本处理其他资源。这可能导致资源创建过程缺乏一致性和可重复性。事实上，在托管环境中常用的许多第三方服务 Terraform 都提供了相应的支持程序，可以用来创建和配置这些服务，例如 Splunk、Datadog、PagerDuty 和 New Relic。因此，我们建议团队除了云资源外，还应使用 Terraform 创建监控和告警。这将实现更模块化的 IaC，更易于理解和维护。与所有 IaC 一样，同时使用多种方式进行配置变更，会带来不一致的风险。所以，我们建议禁用通过用户界面和 API 的方式处理配置变更，确保 Terraform 代码始终是唯一的真实生效的版本。

9. ReAct 提示工程

试验

ReAct 提示工程是一种用于提示大语言模型的方法，相较于思维链（CoT）等竞争方法，ReAct 旨在提高大语言模型的响应准确性。这一方法在一份 2022 年的论文中首次提出，其原理是将推理和行动结合起来（因此称为 ReAct）。这种方法有助于使大语言模型的响应更具解释性，相对于思维链减少了虚构性内容，从而提高了提示者获得他们想要的内容的机会。最初，LangChain 是为支持这种提示方式而开发的。基于 ReAct 的自主代理已被证明是我们团队构建的大语言模型应用中使用最广泛的一种。最近，OpenAI 在其 API 中引入了函数调用以使 ReAct 和类似的提示风格更容易实现，而无需依赖像 LangChain 这样的外部工具。我们仍然处于定义这一学科的早期阶段，但到目前为止，ReAct 及其后继方法已指引出大语言模型最令人兴奋的一些应用领域。

10. 检索增强生成

试验

检索增强生成（RAG）是一种结合预训练参数和非参数记忆的文本生成技术。它使你能够通过你的领域内特有的包含上下文的知识，来强化预训练模型中的现有知识。使用 RAG，你会先从非参数记忆中去检索相关文档集（一般是通过在向量数据库中的相似性搜索），再使用 LLM 中的参数记忆生成与检索出的文档一致的输出。我们发现 RAG 对各种需要大量知识的 NLP 任务十分有用，包括问答，总结和故事生成。

11. 基于风险的故障建模

试验

基于风险的故障建模是一种用于了解系统发生故障的可能性、潜在影响和检测手段的方法。交付团队逐渐开始使用这种方法来设计和评估预防故障所需的控制措施。该方法源自故障模式与影响分析（FMEA）的实践。FMEA 是一种诞生于上世纪 40 年代的风险评分技术，成功运用于航空航天和汽车等建造复杂物理系统的行业中。与这些行业一样，软件故障也可能产生严重后果，例如危及人类健康和隐私。这就是为什么我们越来越需要对系统进行严格分析的原因。该方法首先识别可能的故障模式，然后对根本原因进行分析，并根据故障发生的可能性、影响的大小以及发现根本原因的概率给出评分。我们发现，当跨职能团队在系统演进过程中迭代使用该方法时效果最好。在安全方面，基于风险的故障建模可以作为威胁建模和攻击路径分析的有益补充。

12. 大语言模型半结构化自然语言输入

试验

在使用大语言模型的各种应用中，我们在半结构化自然语言输入方面取得了成功。结构化输入，如 JSON 文档，清晰而精确，为模型提供了所寻求响应类型的指示。以这种方式限制响应有助于缩小问题空间，并且可以产生更准确的响应，特别是当结构符合领域特定语言（DSL）的语法或模式情况下。我们还发现，将结构化输入与自然语言注释或标记结合使用，比仅使用自然语言或结构化输入产生更好的响应。通常，在构建提示时，自然语言会与结构化内容简单地交织在一起。与许多大语言模型行为一样，我们不知道为什么这样做有效，但我们的经验表明，在人工编写的代码中加入自然语言注释也会改善基于大语言模型的编码助手的输出质量。

13. 追踪健康债务状况

试验

通过将健康度评级与其他服务级目标（SLO）同等对待，并据此确定增强的优先级，而不是仅仅关注跟踪技术债务，我们不断体验到团队对其生态系统的改进。通过有效分配资源来解决与健康状况相关的最有影响的问题，团队和组织可以降低长期维护成本，更高效地发展产品。这种方法还能加强技术和非技术利益相关者之间的沟通，促进对系统状态的共同理解。尽管不同组织的衡量标准可能有所不同（请参阅本博文中的示例），但它们最终都有助于实现长期可持续性，并确保软件保持适应性和竞争力。在瞬息万变的数字环境中，专注于跟踪系统的健康状况与债务，可为维护和增强系统提供结构化的循证战略。

14. 对告警规则的单元测试

试验

可观测性和监控对于软件团队至关重要。鉴于特定事件的不可预测性，创建具有复杂规则的准确告警机制至关重要。然而，只有当事件真实出现时，这些规则才能得到真正的验证。对告警规则的单元测试让团队通过预先、主动地测试和完善规则，来更好地定义规则，从而增加对规则的信心。这有助于减少误报，并确保报告真正的事件。Prometheus 等工具支持对规则进行单元测试。 我们的团队报告它的确可以在现实环境中起到帮助作用。

15. CI/CD 的零信任保护

试验

如果没有得到正确的安全配置，运行构建和交付流水线的基础设施和工具可能成为一个大隐患。流水线需要访问关键数据和系统，如源代码、凭据和机密数据，去构建和部署软件。这让这些系统对恶意攻击者充满了吸引力。因此，我们强烈推荐为 CI/CD 流水线和基础设施引入零信任安全机制——尽可能少地信赖它们。这项机制包含一系列技术：如果可行，使用云供应商提供的联合身份校验机制，如 OIDC，来验证流水线，而不是赋予它们直接访问机密数据的权限。实行最小权限原则去最小化个人用户和执行器账户的权限，而不是使用具有无限访问权限的万能账户。使用一次性执行器替代重复使用执行器，来减少暴露先前任务的机密数据或在受到攻击的运行器上运行任务的风险。将执行代理和执行器上的软件更新到最新版本。像监控你的生产软件一样去监控你的 CI/CD 系统的完整性、保密性和可用性。

我们不断见到有团队忘记这些实践，特别是当他们使用在内部网络中自我管理的 CI/CD 基础设施的时候。所有这些实践不仅在内部网络中很重要，当使用托管服务时，因为扩大了攻击面和影响范围，这些实践会变得更加关键。

16. 通过依赖健康检查化解包幻觉风险

评估

确保软件供应链的安全已成为交付团队普遍关心的问题，这也反映在该领域的工具和技术数量不断增加，而一些工具和技术我们在之前的雷达中也进行了介绍。在软件开发过程中使用基于 GenAI 的工具日益普及，这也引发了一种新的软件供应链攻击媒介：包幻觉。我们认为在开发过程中使用 GenAI 工具的团队需要重视这类风险。团队可以通过对依赖进行健康检查化解包幻觉风险：在选择依赖之前查看它的创建日期、下载数量、github 评论及星标数、贡献者数量、活动历史记录等。一些依赖健康检查可以在包存储仓库和 GitHub 上执行，而像 deps.dev 和 Snyk advisor 等工具也可以提供帮助。尽管依赖健康不是一项新技术，但随着团队在软件开发过程中越来越多地尝试 GenAI 工具，该实践正在获得新的关注。

17. 设计系统决策记录

评估

在迭代速度快、用户需求不断演进的产品开发环境中，设计是一个不断变化的领域。这意味着对设计决策输入的需求会一直持续下去。我们借鉴了用 ADR（架构决策记录）记录软件架构决策的思路，采用类似的格式，以设计系统决策记录来记录设计系统决策以及相应的依据、研究洞见和实验结果，这有效地传达了设计系统决策似乎已成为产品开发团队新的需求。这种轻量级的方式也被 zeroheight 推荐。这一方法让我们减少了新人上手时间，推动了讨论的进行，并帮助拉通共用同一个设计系统的多个开发团队。

18. GitOps

评估

GitOps 是一项通过控制回路模式进行应用部署的技术。Operator 能够将已部署的应用和配置（通常是 Git 仓库）保持同步。当我们上次写到 GitOps 的时候，社区对此术语的定义未能形成共识。当时，我们对该技术的常

见解读抱有疑虑，因为部分解读包含不恰当的做法。例如，使用“环境分支”就可能导致雪花即代码的出现。此外，“GitOps 是持续交付的一种替代方案”这个说法也令人困惑。在那之后，四个 GitOps 原则澄清了该技术的范围和性质。当拨开炒作和混乱的迷雾，你会发现 GitOps 是一项基于 Kubernetes 集群功能的有用技术，为分离介于配置应用和实施部署流程的关注点创造了机会。我们的一些团队在他们的持续交付设置中实施了 GitOps，并取得了良好的体验。所以我们推荐大家去评估这项技术。

19. 大语言模型驱动的自主代理

评估

随着大语言模型的持续发展，构建自主人工智能代理的兴趣日益浓厚。AutoGPT、GPT-Engineer 和 BabyAGI 都是大语言模型驱动的自主代理的示例，它们朝着底层大语言模型理解所获得的目标方向努力。这些代理会记住目标的进展程度，使用大语言模型来思考接下来该做什么，然后采取行动，并理解何时已经实现了目标。这通常被称为思维链推理，而且实际上是可行的。我们的团队实现了一个作为自主代理的客户服务聊天机器人。如果机器人无法达成客户的目标，它会认识到自己的限制并将客户引导到人工处理。这种方法显然仍处于早期发展阶段：自主代理通常存在高失败率和高昂的 AI 服务费用，至少有一家 AI 初创公司已经从代理为基础的方法转向其他方向。

20. 平台编排

评估

随着平台工程的广泛采纳，我们看到了新一代的工具，它们超越了传统的平台即服务（PaaS）模型，为开发人员和平台团队之间提供了公开的合约。这个合约可能涉及在不同环境中提供云环境、数据库、监控、身份验证等功能。这些工具强制执行组织标准，同时允许开发人员通过配置自主访问多种环境。这些平台编排系统的案例包括 Kratix 和 Humanitec Platform Orchestrator。我们建议平台团队考虑这些工具，作为自己的脚本、本地工具和基础设施即代码（infrastructure as code，IaC）的独特集合替代方案。我们还注意到，与开放应用模型（OAM）及其参考编排器 KubeVela 有相似之处，尽管 OAM 声称更加面向应用程序而不是工作负载为中心。

21. 自托管式大语言模型

评估

大语言模型（LLMs）通常需要大量的 GPU 基础设施才能运行，但目前有强烈的推动力使它们可以在更简单的硬件上运行。对大语言模型进行量化可以减少内存需求，使高保真度模型可以在成本更低廉的硬件甚至是 CPU 上运行。像 llama.cpp 这样的工作使大语言模型可以在包括树莓派、笔记本电脑和通用服务器在内的硬件上运行成为可能。

许多组织正在部署自托管式大语言模型。这往往是出于安全或隐私方面的考虑，有时是因为需要在边缘设备上运行模型。开源示例包括 GPT-J、GPT-JT 和 Llama。这种方法提供了更好的模型控制，以进行特定用途的微调，提高了安全性和隐私性，以及离线访问的可能性。尽管我们已经帮助一些客户自托管开源大语言模型用于代码生成，但我们建议在决定自托管之前仔细评估组织的能力和运行这类大语言模型的成本。

22. 忽略 OWASP 十大安全风险榜单

暂缓

OWASP 十大安全风险榜单长期以来一直是 Web 应用程序最关键的安全风险参考。尽管众所周知，我们曾写过它在软件开发过程中未得到充分利用，并警告不要忽略 OWASP 十大安全风险榜单。

但鲜为人知的是 OWASP 也在其他领域发布了类似的十大榜单。在八月初发表了第一个主要版本的 OWASP LLM 十大安全风险榜单 强调了提示注入、不安全的输出处理、训练数据投毒以及其他个人和团队构建 LLM 应用程序时最好注意的风险。OWASP 近期也发布了 OWASP API 十大安全风险榜单的第二版。鉴于 OWASP 十大安全风险榜单的覆盖范围（Web 应用程序、API、LLM 及其他）、质量以及与持续变化的安全形势的相关性，我们继续向团队警告不要忽略 OWASP 十大安全风险榜单。

23. 用于服务端渲染（SSR）web 应用的 web 组件

暂缓

自从我们在 2014 年首次提到它们以来，Web 组件已经变得流行起来，总体而言，我们对其的看法是积极的。同样地，我们通过对采用默认选择 SPA 发出警告以及将如 Next.js 和 HTMX 等框架与传统的服务器端框架一起列入，来表达对在服务器上来渲染 HTML 的支持。然而，尽管可以将两者结合使用，也还是可能造成深层次的问题；这就是为什么我们建议避免服务器端渲染（SSR）的 Web 应用中使用 Web 组件。作为一种浏览器技术，要在服务器上使用 Web 组件并不容易。已经出现了一些框架来简化这一过程，甚至一些框架中还使用了浏览器引擎，但复杂性仍然存在。比开发人员体验更糟糕的是用户体验：当必须在浏览器中加载和构建自定义 Web 组件时，页面加载性能会受到影响，即使在预渲染和精心调整组件的情况下，也几乎无法避免“无样式内容闪烁”或某些布局变化。放弃使用 Web 组件的决定可能会产生深远的影响，正如我们的一个团队曾经不得不将其基于 Web 组件的设计系统 Stencil 进行迁移。

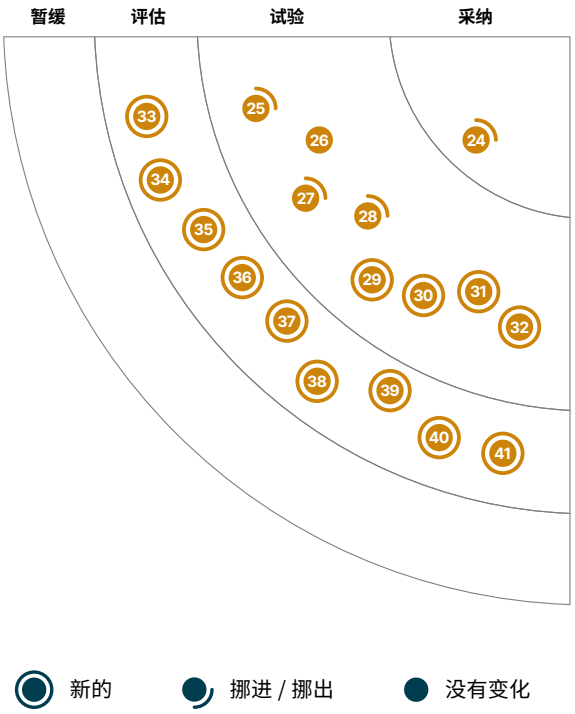
平台

采纳
24. Colima

试验
25. CloudEvents
26. DataOps.live
27. Google Cloud Vertex AI
28. Immuta
29. Lokalise
30. Orca
31. Trino
32. Wiz

评估
33. ActivityPub
34. Azure Container Apps
35. Azure OpenAI Service
36. ChatGLM
37. Chroma
38. Kraftful
39. pgvector
40. Pinecone
41. wazero

暂缓
—



24. Colima

采纳

Colima 现在是我们在 macOS 上替代 Docker Desktop 的首选方案。我们持续在几个项目中使用它来提供 Docker 容器运行时的 Lima VM，在 macOS 上配置 Docker CLI，并处理端口转发和挂载卷。Colima 可以配置为使用 containerd 作为其运行时，这也是大多数托管的 Kubernetes 服务上的运行时，可以提高重要的开发到生产环境的一致性。

25. CloudEvents

试验

事件是事件驱动架构或无服务器应用中常见的机制。然而，生产者或云提供商通常以不同形式支持它们，这阻碍了跨平台和基础架构的互操作性。CloudEvents 是一个描述事件数据的通用格式的规范，旨在提供服务、平台和系统之间的互操作性。它提供了多种编程语言的 SDK，因此您可以将规范嵌入到应用程序或工具链中。我们的团队不仅将其用于跨云平台的目的，还用于领域事件规范等其他场景。CloudEvents 由云原生计算基金会（CNCF）托管，现在作为孵化项目运行，已经获得了越来越多的行业关注。

26. DataOps.live

试验

DataOps.live 是一个自动化 Snowflake 环境的数据平台。受 DevOps 实践启发，DataOps.live 可以像在其他网络平台一样在数据平台中实施持续集成和持续交付（CI/CD），自动化测试，可观测性和代码管理。我们的团队正在用它来管理数据产品的全生命周期，包括代码和数据的开发、分支、部署。通过它的自动化环境管理，能够轻易建立、修改、自动销毁基于特征分支的环境。它的声明式标准（SOLE）能力也值得关注，因其可以优化开发者体验。它能使团队构建数据产品的时间从几个月变为几天。我们的团队成功将 DataOps.live 用于生产环境，这也是我们推荐在使用 Snowflake 时使用这一平台的原因。

27. Google Cloud Vertex AI

试验

自从我们第一次提出 Google Cloud Vertex AI 以来，AI 领域已经发生了重大进展。自 2023 年 5 月以来，Google 推出了多项服务和功能来丰富这一领域。这些新增功能包括 Model Garden，一个拥有 100 多个预训练模型的仓库；Generative AI Studio，一个旨在快速探索和原型生成 AI 模型的控制台；以及 Vertex AI Extensions，提供完全托管的开发人员工具，通过 API 连接 AI 模型和实时数据或操作。 该平台已经发展到提供 GenAI 模型和集成支持，我们非常期待能更广泛地使用它。

28. Immuta

试验

自从我们上次介绍了 Immuta 以来，我们的团队在使用这个数据安全平台方面已经积累了丰富的经验。它的亮点包括能够将订阅和数据策略定义为代码、版本控制以及自动部署这些策略到更高的环境中。它基于属性的访

问控制（ABAC）允许我们将标签关联到数据源；如果用户与相同的标签关联，就会获得访问权限。通过利用 Immuta 和 Snowflake 的集成，我们已经能够以自助方式自动授权对数据产品或数据集的访问。当“用户”请求访问数据产品或数据集时，一旦获得批准，数据产品标签将被关联到“用户”作为属性。由于“用户”的属性与数据源上的标签匹配，因此根据 Immuta 的全局订阅策略，访问权限将自动授予。值得一提的是 Immuta 的数据掩码策略，它通过对个人身份信息（PII）进行掩码和限制来保护数据隐私。可以使用行级安全策略来定义对更细粒度的敏感信息的访问，以确保用户只能访问他们被授权查看的特定数据。我们对 Immuta 非常满意，这也是为什么我们将其列入“试验”的原因：它提供了良好的开发者体验，使大型组织更容易管理数据策略。

29. Lokalise

试验

Lokalise 是一个全自动的本地化平台，它支持特定上下文的翻译。我们的团队在 ETL 流程或开发工作流中使用 Lokalise API 来翻译可本地化的内容。Lokalise 支持多种文件格式的可本地化字符串。一个值得强调的方面是它支持上传整个文件，其中每个键 - 值对都被视为单独的记录并被翻译。在底层，我们利用了 Lokalise 与 Google MT 的集成来处理翻译。 Lokalise 的 Web 界面提供了便捷的访问方式，供人工审阅员验证翻译结果，或者根据需要简化、重新表达翻译内容。在过去，我们曾介绍过类似的工具，例如 Phrase。我们的团队在使用 Lokalise 方面有很好的体验，建议您评估该平台是否适用于协作翻译工作流程。

30. Orca

试验

Orca 是一个专有的云安全平台，用于识别、优先级排序和修复安全风险和合规问题。它支持主流的云提供商和混合设置。Orca 拥有广泛的安全查询和规则，以持续监控已部署的工作负载，检测配置错误、漏洞和合规性问题。它支持云虚拟机、无服务器函数、容器以及已部署工作负载的 Kubernetes 上部署的应用。这些内置的安全规则会定期更新，以跟上不断演进的合规标准和威胁向量。由于 Orca 无需代理，因此提供了良好的开发者体验，并且易于设置。另一个显著的特点是它促进了安全的左移。我们的团队使用 Orca CLI 来扫描容器镜像和 IaC 模板，以检测漏洞和配置错误，作为预提交钩子或 CI/CD 工作流的一部分。它还持续监控和扫描容器仓库（如 AWS ECR），以查找已发布镜像中易受攻击的基础镜像或脆弱的操作系统依赖项。根据我们团队的经验，Orca 提供了从开发到生产的安全状态的统一视图，因此我们将其放入试验阶段。

31. Trino

试验

Trino 以前被称之为 PrestoSQL，是一个专为面向大数据交互式分析查询而设计的开源分布式 SQL 查询引擎。经过优化后，它可以在本地或者云上环境运行，并支持对 Hive、Cassandra、关系型数据库、甚至专有数据存储等多种不同的数据源进行查询。它支持基于密码的认证、LDAP 和 OAuth 的身份验证机制，同时具备在 catalog、schema 和 table 级别授予权限和访问控制的能力。我们的团队根据可视化、报告或机器学习用例等消费模式，使用资源组进行管理和限制资源分配。基于 JMX 的监控提供了丰富的指标集，帮助实现在查询或用户级别进行成本分配。我们的团队将 Trino 用作跨各种数据源的数据访问网关，当涉及到查询极大规模的数据时，Trino 对

我们的团队来说是一个可靠的选择。Trino 起源于 2015 年 11 月首次在 Radar 上亮相的 Presto 项目，具体可参考 Facebook project 。

32. Wiz

试验

Wiz 是日渐成熟的云安全平台领域里又一竞争者，它能让用户在一个平台上预防、检测和应对安全风险和威胁。Wiz 能对尚未部署到生产环境的构建产物（容器镜像、基础设施代码）以及生产工作负载（容器、虚拟机和云服务）的错误配置、漏洞和泄漏的机密数据进行检测并发出警报。 它还能将发现的问题置于特定客户的云环境的上下文中，使响应团队能够更好地了解问题并确定修复优先级。我们的团队在使用 Wiz 时获得了良好的体验。他们发现 Wiz 正在快速发展并不断增加新的功能。值得称赞的是，由于 Wiz 能持续扫描变化，团队因此能够比使用一些其他类似工具更快地发现风险和威胁。

33. ActivityPub

评估

随着微型博客平台领域的剧变，ActivityPub 协议逐渐名声鹊起。ActivityPub 是一个用于分享诸如帖子、出版物和日期等信息的开放协议。它可以用来实现一个社交媒体平台，但其关键优势在于能够实现不同社交媒体平台之间的协同工作能力。我们预计 ActivityPub 将在社交媒体领域扮演重要角色，但更加对其在其他领域可能发挥的作用感到好奇。一个例子就是最近 GitLab 提出对合并请求增加 ActivityPub 支持。

34. Azure Container Apps

评估

Azure 容器应用是一种 Kubernetes 命名空间托管服务。该服务通过消除 Kubernetes 集群和底层基础架构组件的复杂维护需求来简化容器化工作负载的部署，从而减轻了运维和管理负担。然而，在考虑使用该工具时需要小心谨慎：当前处于开发阶段，它在 Azure 门户中展示的功能有些不一致；在与标准 Terraform Azure 插件集成时遇到了困难，该插件在匹配 Azure 容器应用的功能方面进展缓慢。综上所述，我们建议仔细评估这个工具。

35. Azure OpenAI Service

评估

伴随对生成式 AI 的巨大关注，许多访问主流模型的解决方案应运而生。如果正在考虑或正在使用 Azure，那么我们推荐评估 Azure OpenAI 服务。它通过 REST API 、Python SDK 以及基于 Web 的界面提供对 OpenAI 的 GPT-4、GPT-35-Turbo 和嵌入模型的访问。这些模型可以适应如内容生成、汇总、语义搜索和自然语言到代码的转换的任务，也可以通过少量学习和超参数的定制进行微调。与 OpenAI 自己的 API 相比，Azure OpenAI 服务受益于 Azure 企业级的安全性和合规性，同时也在更多的区域可用，哪怕每个较大的地理区域的可用性是有限的。

36. ChatGLM

评估

在英语世界中，有许多新兴的大语言模型（LLM）。虽然这些模型通常经过多种语言的预训练，但它们在其他语言中的表现可能不如英语。清华大学开发的 ChatGLM 是一个开放的双语语言模型，基于通用语言模型架构，针对中文会话进行了优化。由于中文在词语划分和语法方面较英语更为复杂，因此拥有一个针对中文进行优化的 LLM 非常重要。我们的团队在为呼叫中心开发中文情感检测应用时发现，ChatGLM 在准确性和鲁棒性方面都优于其他 LLM。考虑到许多 LLM 因授权或地区限制而无法在中国使用，ChatGLM 成为了为数不多的开源选择之一。

37. Chroma

评估

Chroma 是一个开源的向量存储和嵌入数据库，可用于增强由大语言模型（LLMs）驱动的应用程序。通过促进 LLMs 中的领域知识的存储和利用，Chroma 弥补了 LLMs 通常缺乏内部存储器的不足。特别是在文本到文本应用中，Chroma 可以自动生成单词嵌入并分析它们与查询嵌入之间的相似性，从而大大简化操作。它还提供了存储自定义嵌入的选项，促进了自动化和定制化的融合。鉴于 Chroma 能够增强由 LLM 驱动的应用程序的功能，我们建议团队对 Chroma 进行评估，挖掘其潜力，改进将领域知识集成到此类应用程序中的方式。

38. Kraftful

评估

用户体验（UX）研究平台，比如 Dovetail，为组织提供了一个工具，帮助他们了解和改善客户体验。通过这个工具，企业能够迅速、轻松地通过收集和分析来自客户反馈、调查、访谈等渠道的数据，深入洞察客户的需求、偏好和行为。情感分析、客户分割、市场研究、数据分析和洞见生成是产品开发中有价值的任务，而这些正好是大语言模型擅长的领域，因此我们看到了它在产品开发领域存在巨大的颠覆潜力。

Kraftful 是一个自诩为产品构建者副驾驶的工具，现在已经处于领先地位。它目前仅处于测试阶段，您需要提供邮箱才能访问其功能。我们已经试用过它并取得了很好的效果。您可以将 30 多种用户反馈来源连接到这个平台，它可以分析数据并识别功能请求、常见投诉、用户喜欢的产品特点，甚至列出您的竞争对手。为了获取更多细节，您可以像向 ChatGPT 或 Google Bard 提问一样，这里的好处是它针对您的数据进行了优化。一旦您确定了要从用户反馈中解决的问题，Kraftful 会基于所有基础数据（包括验收标准）为您生成用户故事，即使对经验丰富的产品经理和业务分析师来说它也是一个出色的助手。

39. pgvector

评估

随着生成式人工智能应用的兴起，我们观察了一种储存和有效搜索嵌入（embeddings）向量相似性的模式。pgvector 是一个用于 PostgreSQL 的开源向量相似性搜索插件。我们非常喜欢它，因为它能够让我们在 PostgreSQL 中搜索 embeddings，而无需仅为了相似性搜索而将数据转移到另一个存储中。尽管目前已有几个专门的向量搜索引擎，但我们还是希望你能评估一下 pgvector。

40. Pinecone

评估

Pinecone 是一个完全托管的、对开发人员友好的、云原生的向量数据库。它提供了简单的 API，而无需处理基础设施方面的繁琐工作。Pinecone 可在数十亿向量数据的规模上，提供过滤后的查询结果，且延迟较低。我们的团队发现，对于存储团队知识库或服务台门户内容等使用场景，相较于针对对复杂的 LLM 进行微调，使用数据库提供商，特别是 Pinecone 会更便利，且上手很快。

41. wazero

评估

wazero 是使用 Go 编写的一个零依赖的 WebAssembly（WASM）运行时。尽管运行时本身与语言无关，我们仍想对 Go 开发者们强调 wazero，因为它提供了一种很方便的方式，使用任何 符合标准的语言 编写的 wasm 模块来扩展你的 Go 应用程序。它不依赖于 CGO，所以你可以很容易地将你的 Go 应用程序交叉编译到其他平台。尽管在选择 WASM 运行时 的时候你有很多候选项，但我们仍认为 wazero 值得评估。

工具

采纳

42. dbt
43. Mermaid
44. Ruff
45. Snyk

试验

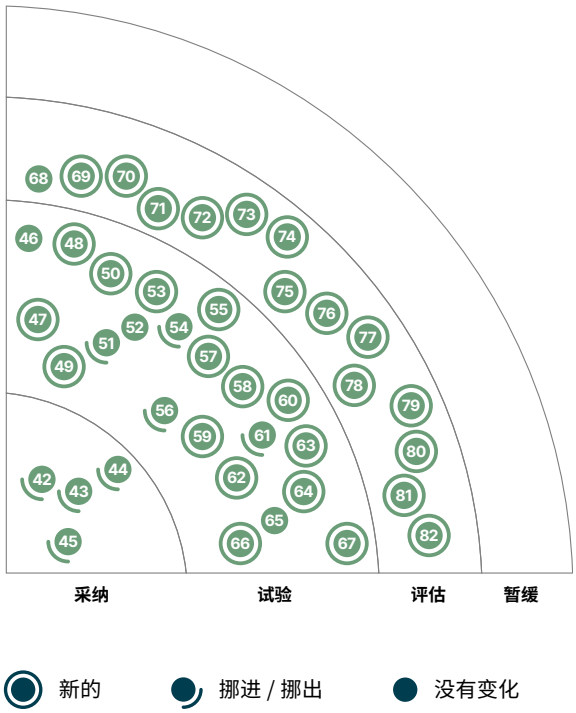
46. AWS Control Tower
47. Bloc
48. cdk-nag
49. Checkov
50. Chromatic
51. Cilium
52. 云服务的碳足迹
53. 容器结构测试
54. Devbox
55. DX DevEx 360
56. GitHub Copilot
57. Insomnia
58. IntelliJ HTTP 客户端插件
59. KEDA
60. Kubeconform
61. mob
62. MobSF
63. Mocks Server
64. Prisma 运行时防护
65. Terratest
66. Thanos
67. Yalc

评估

68. ChatGPT
69. Codeium
70. GitHub 合并队列
71. Google Bard
72. Google Cloud 工作站
73. Gradio
74. KWOK
75. Llama 2
76. Maestro
77. Open-source LLMs for coding
78. OpenCost
79. OpenRewrite
80. OrbStack
81. Pixie
82. Tabnine

暂缓

—



42. dbt

采纳

dbt 仍是我们在 ETL 工作流程中进行数据转换的首选工具。我们喜欢它的工程严谨性和它在 SQL 数据转换中实践模块化、可测试性、和可复用性的能力。 dbt 有开源和商业化 SaaS 产品两种版本和健康的生态，包括一个提供了许多用于单元测试、数据质量、数据可观测性等软件包的社区。这些包中尤为值得注意的是用于监测数据质量的 dbt-expectations 和用于构建数据转换的单元测试的 dbt-unit-testing。dbt 很好地集成了各种云数据仓库、数据湖和数据库，包括 Snowflake，BigQuery，Redshift，Databricks 和 Postgres。当需要处理结构化数据并且能使用 SQL 进行数据转换时，我们的团队们倾向于 dbt，因此我们将它移至采纳阶段。

43. Mermaid

采纳

Mermaid 通过使用类似 Markdown 的标记语言来生成图表。自从上次在技术雷达中介绍以来，Mermaid 添加了对更多图表和与源代码存储库、集成开发环境和知识管理工具集成的支持。 值得注意的是，它在 GitHub 和 GitLab 等流行源代码存储库中得到原生支持，从而可以在 Markdown 文档中嵌入并轻松更新 Mermaid 图表。我们的许多团队都倾向于使用 Mermaid 作为他们的图表即代码工具，因为它易于使用、集成广泛，且支持的图表类型不断增多。

44. Ruff

采纳

Ruff 是一个新的 Python linter。使用 linter 是毋庸置疑的，只需要考虑具体要使用哪一个。Ruff 能够脱颖而出有两个原因：开箱即用的体验，以及性能。其中内置了 500 多条规则，可以轻松取代 Flake8 和它的许多插件。我们的经验证实了 Ruff 团队对其性能的说法。实际上，它的速度至少比其它 linter 快出一个数量级，这是一个巨大的优势，有助于减少大型代码库的构建时间。基于上述原因，Ruff 已成为我们实施 Python linter 的默认选择。

45. Snyk

采纳

Snyk 提供静态应用程序安全测试（SAST）和软件组件分析（SCA）测试，以帮助您在软件开发生命周期中寻找、修复和监控安全问题。其广泛的功能旨在加快反馈循环，倾向于采用“左移”方法，而不是安全三明治反模式。作为今天可用的最佳安全平台之一，Snyk 之所以脱颖而出，是因为它能够识别更广泛的问题，而这主要得益于有专门的研究团队不断更新其漏洞数据库。但是 Synk 仍有改进的空间：仪表板目前没有提供一个简便的方法从一个具体的可操作的信息中过滤一些多余繁杂的信息；根据语言生态系统的不同，基于 SCA 的集成可能与基于流水线的集成相比产生误报，因为 Snyk 必须猜测已解决的依赖关系；自动解决方案的成功性不一致；在高度监管的环境中，需要进行重大的集成投资，以实现适当的门控或建立软件物料清单。尽管存在这些缺点，我们的许多企业客户依然采用了 Snyk，我们自己也在 IT 部门中使用了它。

46. AWS Control Tower

试验

在 AWS 中，多团队的账户管理是一项挑战，尤其是在设置和治理方面。AWS Control Tower 通过简化设置和自动化治理来应对这个挑战，并通过防护措施应对监管要求。AWS Control Tower 内置了一个账户工厂，帮助自动化账户的配置流程。您可以通过账户工厂来取消账户托管、更新和关闭创建与配置的账户。由于其缺乏自动化和定制化，亚马逊引入了 Terraform 的 AWS Control Tower 账户工厂（AFT）。AFT 允许配置定制化的 Webhook 或特定操作，以便与其他工具集成来启动账户创建流程。我们的团队通过整合一组开箱即用的账户配置项，一劳永逸地为 GitHub Actions 的角色做了基础设置和访问权限的配置。这可为开发者提供一个完全集成 VPC 安全基线、可用于 GitHub Actions 接收工作负载的账户。我们的团队报告说，使用 AWS Control Tower 和 AFT 统一管理多个团队的账户非常方便。

47. Bloc

试验

Bloc 是 Flutter 的一款响应式状态管理库。在 Flutter 可用的状态管理选项中，我们想突出 Bloc，因为我们团队在使用该库构建复杂移动应用程序时体验很好。当 UI 组件通过流和事件接收器与业务逻辑进行通信时，围绕 BLoC 模式 结构化的组织代码实现了业务逻辑与表示层的完全分离。Bloc 在 IntelliJ 和 VSCode IDE 中都提供了良好的插件支持。

48. cdk-nag

试验

cdk-nag 能够识别并报告 AWS CDK 应用程序或 CloudFormation 模板中的安全性和合规性问题。它附带了几个所谓的规则包：一个通用的 AWS 规则包，包括 AWS 认为的最佳实践检查，以及用于 HIPAA、NIST 和 PCI 合规性的规则包。您可以根据需要添加额外的规则。规则可以导致警告或错误，这两者都包含在工具生成的报告中。当存在错误时，cdk deploy 命令将无法进行部署。如果错误的原因无法及时修复，仍然可以在错误存在但已被抑制的情况下进行部署。显然，这应该只在特殊情况下执行。

49. Checkov

试验

Checkov 是一个专门用于基础设施即代码（IaC）的静态安全扫描器。它支持多种基础设施语言，包括 Kubernetes 清单、Helm 图表、CloudFormation 模板和 Terraform。它可在 CI/CD 管道中轻松部署，防止各种云基础设施配置中出现潜在的安全漏洞。它利用一套默认规则，识别常见的安全情景，并在其网站上提供详细的修改建议。Checkov 支持自定义规则，并使用 YAML 进行简单的准则定义，或使用 Python 制作更复杂的准则定义。我们的团队已成功使用 Checkov 在基础架构部署过程中增强安全性，并对其在部署前提供的潜在问题表示欣赏。

50. Chromatic

试验

Chromatic 是一款可视化回归测试工具，可帮助捕捉网络应用程序中的用户界面回归。它的工作原理是拍摄用户界面组件的快照，并在组件发生变化时将其与之前的快照进行比较。Chromatic 是一种托管服务，可与流行

的云代码托管服务集成。它建立在 Storybook 之上，进行组件级可视化回归测试。它可以在不同的视口宽度下渲染组件，以进行响应式测试，并与 CI 工作流集成，为每次提交生成用户界面变更集，从而方便审查。我们的团队发现 Chromatic 与该领域其他工具的视觉差异要大得多；可视化高亮显示变更的功能让它非常实用。

51. Cilium

试验

eBPF 以其应用透明、高性能和低开销而闻名，因此云原生社区一直在探索其在无边车网格服务（service mesh without sidecar）中的应用场景。Cilium 是一个为云原生环境如（Kubernetes 集群和其他容器编排平台）提供网络、安全性和可观察性的开源项目。Cilium 为路由或覆盖网络提供了一个简单的第三层网络，并且还支持 L7 协议。通过将安全性从寻址中解耦，Cilium 可以作为一种新的网络保护层发挥重要作用。我们已经看到一些云服务提供商采用了 Cilium，我们的一些项目中也使用了 Cilium。社区仍在讨论 eBPF 是否可以替代边车（sidecar），但似乎这已经达成共识，即某些网格功能不能或不应该在内核中执行。此外，使用 Cilium 还需要 eBPF 相关的经验。基于我们项目取得的良好成果，我们建议您亲自实践一下这项技术。

52. 云服务的碳足迹

试验

Cloud Carbon Footprint（CCF）是一款估算主要云服务提供商的云工作负载碳排放量的开源工具。它通过云平台的 API 查询资源使用数据，并使用多种（数据）来源跟踪碳排放情况。CCF 遵循公开发布的方法，将这些数据合并为排放量估算值，并提供随时间变化的数据可视化。云提供商已经开始在其平台上添加类似产品，但是一些企业仍在部署 CCF，因为它具有以下所有功能：它是开源的，可扩展的，能跨多云工作，并且有一个透明公开的计算方法。此外，它还包括范围 2 和范围 3 排放的估算，分别针对电力使用和硬件生产。在我们的实验中，不同工具的估算结果不尽相同，这并不奇怪，因为该领域的所有工具都会进行估算，并将估算值相乘。然而，确定一种工具、设定基准线并在此基础上进行改进是我们遇到的主要使用场景，类似 Kepler 这样的工具可能会在未来减少对估算的需求。CCF 还提供基于 GCP 和 AWS 的优化建议，这不仅有助于减少云服务的碳足迹，还可以成为更广泛的云成本优化战略的一部分。Thoughtworks 是 CCF 的重要贡献者。

53. 容器结构测试

试验

容器结构测试（CST）是由 Google 开发的一个工具，用于测试容器镜像的结构。CST 可以用于检查镜像文件系统中某个文件的存在或缺失，验证文件的内容，检查容器中发出的特定命令的输出或错误，并检查容器镜像的元数据（例如标签、入口点和命令），以确保符合 CIS Docker Benchmark 的规范。我们在使用 CST 方面有很好的经验，建议您可以试用一下。除了预防漏洞，检查容器是否暴露不必要的端口之外，我们还使用它来验证每个 Docker 容器是否满足在企业平台上部署和运行一个应用程序的所有必要要求。其中一个要求是镜像中安装了可观测性代理。需要注意的是，CST 并没有得到 Google 的官方支持，这可能会影响它的维护情况。

54. Devbox

试验

Devbox 是一款基于终端的工具，具有便捷易用的界面，用于创建可重用，项目独立的开发环境，Devbox 利用 Nix 软件包管理器，而无需使用虚拟机或容器。我们的团队使用它消除不同项目的开发环境中 CLI 工具和自定义

脚本的版本与配置不匹配的问题，以及标准化管理项目中不同语言包的提供。我们发现 Devbox 显著简化了项目入门流程，只要代码库配置了该工具，只需运行一个 CLI 命令（“devbox shell”）就能将开发环境配置到新机器上。Devbox 支持 shell 钩子、自定义脚本和生成便于集成 VSCode 的 devcontainer.json。

55. DX DevEx 360

试验

DX DevEx 360 是一款基于调查的工具，它通过聚焦开发人员在日常工作中面临的阻力点，如代码审查流程、代码质量、深度工作能力等，寻找提高开发人员生产力的关键指标。该调查由 Nicole Forsgren 和 Margaret-Anne Storey 制定，他们曾和其他专家一并主导了 DORA 和 SPACE 两个项目。

我们的平台工程团队已成功使用 DX DevEx 360 来了解开发人员的观点并识别存在的阻力点，从而为平台发展路线提供参考。与类似调查工具不同，使用 DX DevEx 360，我们获得了 90% 以上的回应率，开发人员通常会就问题和改进想法进行详细评论。该工具令人赞赏之处还有：它将结果透明化给公司的工程师，而不仅仅是经理，此外它支持按团队进行分析，从而实现每个团队环境的持续改进。

56. GitHub Copilot

试验

GitHub Copilot 被我们的许多团队用来帮助他们更快地编写代码，总体来说，我们的绝大多数开发人员认为这个工具非常有用，并且如果我们限制这个工具的使用，他们可能会感到失望。我们一直在通过生成式 AI 探索集和 Copilot 入门指南 整理和分享我们使用 Copilot 的经验。请注意，任何代码库都可以使用 Github Copilot，不局限于托管在 GitHub 上的代码库。

我们还很高兴看到自上次在技术雷达中亮相以来，来自 Copilot X 路线图 的 Copilot 聊天功能已经变得更加的普及。这是 Copilot 内联辅助功能的一个强大的补充。应用在 IDE 内的聊天界面，提高了常见信息检索的可发现性，并且与开放式编辑器上下文的集成使得对错误的研究或请求聊天协助执行与焦点代码相关的任务变得轻而易举。

57. Insomnia

试验

自从 Postman 在 2023 年 5 月宣布将逐渐淘汰具有离线功能的 Scratch Pad 模式以后，需要将 API 工作区数据从第三方服务器上隔离的团队不得不寻找替代方案。Insomnia 就是可选的替代方案之一：这是一款专为 API 测试、开发和调试而设计的开源桌面应用程序。虽然 Insomnia 支持在线同步，但它可以让你离线保存 API 工作区数据。我们的团队发现，从 Postman 到 Insomnia 进行人工 API 测试是无缝迁移的，因为它们功能相似，而且 Insomnia 允许导入 Postman 的集合。尽管我们的团队在 Insomnia 上获得了良好的体验，但我们仍在关注其他各种形式的开发替代方案——从 GUI 工具（如即插即用的 Insomnia），到 CLI 工具（如 HTTPie），再到 IDE 插件（如 IntelliJ HTTP 客户端插件）。

58. IntelliJ HTTP 客户端插件

试验

IntelliJ HTTP 客户端插件允许开发人员在代码编辑器中创建、编辑和执行 HTTP 请求，从而简化了构建和使用 API 的开发流程。它在我们的团队中越来越受欢迎，团队成员们喜欢它的用户友好性和便利性。它的显著特点包括支持私有文件（默认情况下将敏感密钥排除在 git 之外，从而保护这些密钥）、版本控制和使用变量的能力，这增强了开发者的体验。鉴于它能简化开发人员的工作流程并增强安全措施，我们建议您尝试使用这个工具。

59. KEDA

试验

KEDA 全称 Kubernetes Event-Driven Autoscaler，正如名字所展示的，它可以根据需要处理的事件数量来伸缩 Kubernetes 集群。根据我们的经验，相比采用 CPU 使用率等滞后指标，更加推荐使用队列深度等领先指标。KEDA 能支持不同的事件源，并提供了一个包含 50 多种自动缩放器的工具箱，可用于各种云平台、数据库、消息系统、遥测系统、CI/CD 系统等。我们的团队报告称，KEDA 的易集成性使他们能够继续在 Kubernetes 中运行微服务的全部功能，否则可能会考虑将一些事件处理代码转移到无服务器函数中。

60. Kubeconform

试验

Kubeconform 是一个用来验证 Kubernetes 清单和自定义资源（CRD）的简化工具。它能很容易地在 CI/CD 流水线或本地机器中部署，通过在部署前验证资源，以减少潜在的错误。鉴于 Kubeconform 在加强运行保证方面有良好记录，特别是在跨团队共享模板资源方面，我们建议试用 Kubeconform 以提高资源验证流程的安全性和效率。

61. mob

试验

mob 是一个用于远程结对编程或集体编程中无缝进行 git 交接的命令行工具。它将所有版本控制工具隐藏在一个命令行界面背后，这使参与集体编程会话变得更加简单。它还提供了关于如何远程参与的具体建议，例如，在 Zoom 中“窃取屏幕共享”，而不是结束屏幕共享，以确保参与者的视频布局不发生变化。我们的一些团队强烈推荐 mob，并且它已经成为我们在远程协作或集体编程中工具链的重要组成部分。

62. MobSF

试验

MobSF 是一个开源的、自动化的静态和动态安全测试工具，用于检测 iOS 和 Android 移动应用程序中的安全漏洞。它扫描应用程序源代码和二进制文件，并提供有关漏洞的详细报告。MobSF 以 Docker 镜像的形式分发，并提供易于使用的 REST API，可以通过 mobsfscan 集成到持续集成 / 持续发布流水线中。我们使用 MobSF 对 Android 应用程序进行安全方面测试的体验是积极的，我们建议您尝试使用它来满足您对移动应用程序安全测试需求。

63. Mocks Server

试验

Mocks Server 是一个基于 Node.js 的 API Mock 工具，它能够复制复杂的 API 响应、响应头和状态码，因此受到了我们团队的重视。它的动态响应生成支持模拟多种场景，允许对 API 交互进行严格测试。Mock 可以描述为 YAML 或 JSON，并通过 CLI、REST API 或 JavaScript 代码进行管理。Mocks Server 的功能包括请求匹配、代理和录制重现功能，这些功能有助于模拟真实的 API 交互。我们特别喜欢将它与 Docker 容器集成，使其可以轻易地在不同环境之间一致地部署，因此它可以作为整个生态系统的一种构件进行版本控制和维护。它简单直接的方法与我们在开发过程中强调的简单性和效率相一致。随着我们的解决方案和测试策略的演进，我们期待更广泛地使用 Mocks Server。

64. Prisma 运行时防护

试验

Prisma 运行时防护是 Prisma 云套件的一部分，为容器安全提供了一种新方法。它采用一种机制来建立容器预期行为模型，然后在运行期间发现异常时检测并阻止异常活动。Prisma 运行时防护监控容器进程、网络活动和文件系统，查找表明可能正在进行攻击的模式和变化，并根据配置的规则进行阻止。学习构成“正常”行为的模型是通过对 Docker 镜像的静态分析和预先配置的时间段内的动态行为分析构建的。我们的团队从使用中发现了可喜的成果。

65. Terratest

试验

Terratest 仍是我们感兴趣的基础设施测试工具。它是一个 Golang 库，用来简化基础设施代码的自动化测试编写。通过基础设施即代码的工具，例如 Terraform，你可以创建真实的基础设施组件（如服务器、防火墙或负载均衡器），在它们之上部署应用程序，并使用 Terratest 验证预期的行为。在测试结束后，Terratest 可以取消应用的部署并清理资源。我们的团队们认为这种测试基础设施组件的方式有助于提供对基础设施即代码的自信。我们看到我们的团队们对应用组件和它们之间的集成编写了各种基础设施安全测试，包括检查错误配置，验证访问权限（比如检查特定 IAM 角色和权限是否被正确配置），检查对敏感资源的未认证访问的网络安全测试等这使得安全测试左移并在开发过程中提供反馈成为可能。

66. Thanos

试验

尽管 Prometheus 一直是自维护可观察性工具链中的一个可靠选择，但当监测指标在基数和总量上增长，以及开始需要高可用性设置时，许多管理现代云原生分布式系统的团队都会碰到其单节点的限制。Thanos 通过添加一些适用于大规模、长期和高可用性监控的功能来扩展 Prometheus。例如，它引入了一些组件将从 Prometheus 实例中读取的数据存储到对象存储系统中，管理对象存储系统对数据的保留和压缩，并且横跨多个 Prometheus 实例做联合查询。我们的团队发现从 Prometheus 迁移到 Thanos 是无缝的，因为 Thanos 保持了与 Prometheus 查询 API 的兼容性。这意味着团队可以继续使用现有的仪表板、警报工具和其他与 Prometheus API 集成的工具。尽管我们的团队在使用 Thanos 上已经取得了成功，但我们也推荐关注另一种扩展 Prometheus 的方式 —— Cortex。

67. Yalc

试验

Yalc 是一个简易的本地 JavaScript 包管理库以及跨本地开发环境进行发布与包管理的工具。对于有一些限制的 npm link 指令来说，Yalc 是一个更可靠的替代品。在使用多个包的时候 Yalc 特别好用，尤其是当有些包使用 yarn，而有些包使用 npm 的时候。它同样能帮助在发布包到远端之前进行本地的测试。根据我们的经验，Yalc 在配置多个包和加速前端以及其他 JavaScript 应用开发的工作流方面非常有价值。

68. ChatGPT

评估

ChatGPT 继续引起关注。随着充满想象力的应用场景和创造性地提示词的涌现，它正在不断扩大其实用性。GPT4 这一驱动 ChatGPT 的大语言模型（LLMs），现在也有能力与知识管理库、沙箱编程环境或网络搜索等外部工具进行集成。最近推出的 ChatGPT 企业版提供了使用情况记录和通过单点登录的用户管理等“企业版”功能，可能有助于缓解一些关于知识产权的担忧。

尽管 ChatGPT“编写”代码的能力受到了诸多赞誉，但我们认为组织更应考虑在全软件生命周期范围内利用它来提高效率并减少错误。例如，ChatGPT 可以为需求分析、架构设计或对遗留系统进行逆向工程等任务提供额外的视角和建议。我们仍认为 ChatGPT 最好作为一个流程的输入—— 例如起草一个故事的初稿或给出某项编码任务的框架——而不是一个生成“成熟”结果的工具。话虽如此，ChatGPT 的能力还在持续增强，通过谨慎地给出提示词，它已经可以完成很多编程任务。而提示工程本身就是一门艺术。

69. Codeium

评估

在 AI 编程辅助工具中，Codeium 是一款较为有前景的产品。类似于 Tabnine，Codeium 也尝试去解决公司使用编程辅助工具最担心的问题：通过不使用无授权代码训练自己的模型，他们消除了部分开源代码许可证带来的担忧。同时他们允许您自托管这个工具,所以无需向第三方发送代码片段。 Codeium 在广泛支持各种 IDE 和计算笔记本上表现突出，并且虽然它的出现晚于 GitHub Copilot 和 Tabnine，但是我们对这一产品的第一印象非常正面。

70. GitHub 合并队列

评估

我们一直是短暂开发分支的倡导者，这些分支经常合并到主代码分支中，主代码分支始终准备好进行部署。这种主干开发的实践与持续集成密切相关，并且在条件允许的情况下，可以实现最快的反馈循环和最高效的开发流程。然而，并不是每个人都喜欢这种方法，我们经常根据客户的实践来调整我们的风格。有时，这包括长期存在的特性分支和拉取请求必须被手动审查和批准，然后才能将它们合并到主分支中。在这些情况下，我们使用新的 GitHub 合并队列功能。它允许我们自动排队接收的拉取请求，并将它们合并到特殊分支中，按接收顺序进行排序。然后，我们可以选择自动执行我们自己的“合并检查”，以防止不兼容的提交。这本质上是模拟主干开发（即使 PR 尚未合并到主代码分支中），允许开发人员在上下文中测试其功能，而无需等待批准拉取请求。使用 GitHub 合并队列，即使你不能直接提交到主干，也可以获得主干开发的好处。

71. Google Bard

评估

Google Bard 是由 Google AI 开发的生成式 AI 聊天机器人。 与 ChatGPT 非常相似，它能够通过生成类似人类的文本，会话式地对各种提示和问题提供回复。 Bard 由 Google 的 Pathways 语言模型（PaLM 2）提供支持，PaLM 2 是在海量文本和代码数据集上训练出的大语言模型（LLM）。Bard 能够生成文本、翻译语言、生成各种创意内容，以及详尽回答提出的问题。

Bard 还可以指导软件开发。我们的开发人员发现，Bard 有时可以针对不同场景提供一些编码建议、最佳实践或基础设施配置。我们还尝试使用 Bard 翻译技术雷达，可以得到不错的初稿。尽管该工具仍在开发中，与 ChatGPT 相比它可能会慢一些，但我们仍然鼓励开发人员探索使用这个工具，并评估其潜在优点。

72. Google Cloud 工作站

评估

Google Cloud 工作站 是 GCP 提供的云端开发环境（Cloud Development Environment，CDE）。它提供了完全托管的容器化开发环境，可以通过 SSH、HTTPS、VSCode、Jetbrains IDEs 等多种方式进行访问，使开发人员可以享受和连接本地环境一致的体验。Google Cloud 工作站允许管理员将容器化开发环境纳入私有网络，并可以选择使其对外公开或仅在内部访问。这种网络配置的灵活性，再加上支持使用自定义或预定义的镜像构建环境，使得 Google Cloud 工作站，在我们看来，值得那些在其 GCP 边界内寻找安全的 CDE 解决方案的组织将其纳入评估。如果您正在考虑使用 Google Cloud 工作站，我们建议在广泛推广之前先测试网络配置，因为高延迟可能会对开发者在这些容器中的使用体验造成一定的影响。

73. Gradio

评估

Gradio 是一个开源的 Python 库，它能够帮助我们快速简单地为机器学习模型创建基于 Web 的交互式界面。基于机器学习模型的图形化界面，使得非技术受众能够更好地理解输入、约束和输出。Gradio 支持多种输入和输出类型，从文本到图像再到语音，并且它已经成为了快速原型设计和模型评估的首选工具。 Gradio 可以让您轻松地将您的演示托管到 Hugging Face，或者在本地运行它，然后允许他人通过带有“XXXXX.gradio.app”的 URL 来访问您的远程演示。例如，著名的 DALL-E 迷你实验就是利用 Gradio 实现，并且托管在 Hugging Face Spaces。我们的团队在实验和原型设计中使用这个库的体验非常愉快，这也是我们将其纳入评估的原因。

74. KWOK

评估

KWOK (Kubernetes WithOut Kubelet) 是一个模拟虚拟节点和 pod 生命周期的工具，旨在测试 Kubernetes 集群的控制面板。在没有显著大型集群的情况下，很难对自定义 Kubernetes 控制器和操作器进行压力测试。然而，通过 KWOK，你可以在笔记本电脑上轻松设置一个拥有数千个节点的集群，而无需消耗大量 CPU 或内存资源。这种模拟支持节点类型和 pod 的不同配置，以测试各种场景和边缘情况。如果你需要一个真正的 Kubernetes 集群来测试操作器和自定义资源定义（CRDs），我们推荐 kind 或 k3s; 但如果你只需要模拟大量的虚拟节点，那么我们建议你评估 KWOK。

75. Llama 2

评估

Llama 2 是一个来自 Meta 的强大的语言模型，可免费用于研究和商业用途。它既提供原始的预训练模型，也提供了经过微调的用于对话的 Llama-2-chat 和用于代码补全的 Code Llama。Llama 2 提供了多种尺寸的模型——7B、13B 和 70B，因此如果您想控制自己的数据，Llama 2 是自托管式大型语言模型的一个好选择。

Meta 声称 Llama 2 是“开源”的，但这一说法受到了一些批评。Meta 的许可证和适用策略对用户的商业用途以及对模型和软件的用途做了限制。Llama 2 的训练数据并不开放，这可能会阻碍理解和修改模型。尽管如此，至少在“半开放”的形式下，一个强大、能干的模型仍然是值得欢迎的。

76. Maestro

评估

Maestro 是一款新的跨平台移动端 UI 测试自动化工具，它内置了由网络因素导致的应用加载时长变化的容错能力。通过声明式的 YAML 语法，它使编写和维护移动应用自动化测试变得很简单。Maestro 支持 iOS 和 Android 原生应用、React Native 和 Flutter 应用，能够自动化复杂的移动端 UI 交互（如点击、滚动和滑动）的各种功能。Maestro 以单个二进制文件进行发布、方便使用，以解释器模式运行，并且通过连续模式等特性来简化新的测试的编写。尽管 Maestro 对 iOS 设备的特定功能还欠缺支持，但该工具正在迅速演进。

77. Open-source LLMs for coding

评估

GitHub Copilot 是软件开发时有价值的辅助编程工具。而在工具背后，大语言模型（LLMs）通过赋能内联代码助手、代码微调和 IDE 中的对话支持等方式，无缝提升开发人员的体验。 大多数这些模型都是专有的，只能通过订阅服务使用。好消息是，您可以使用几种开源的 LLMs 进行编码。如果您需要构建自己的编码辅助服务（比如受到高度监管的行业），可以考虑 StarCoder 和 WizardCoder。StarCoder 使用由 BigCode 维护的大型数据集 进行训练，而 Wizardcoder 是 Evol-Instruct 调整后的 StarCoder 模型。

我们在实验中使用了 StarCoder，发现它对于生成诸如代码、YAML、SQL 和 JSON 等 结构化软件工程元素十分有用。根据我们的实验，我们发现这两个模型都可以使用提示词中的 小样本示例 进行上下文学习。尽管如此，对于特定的下游任务（例如为 Postgres 等特定数据库生成 SQL），模型仍需要微调。最近，Meta 推出了 Code Llama，一款专用于编程的 Llama 2。使用这些开源模型时务必要小心谨慎。在选择任何这些编码 LLMs 供您的组织使用之前，请考虑它们的许可，包括代码的许可和用于训练模型的数据集的许可，仔细评估这些方面后再做决定。

78. OpenCost

评估

OpenCost 是一个开源项目，用于监控基础设施成本，并以 Kubernetes 对象（Pod、容器、集群等）的粒度展示成本信息，涵盖各种集群内资源（CPU、GPU、RAM、存储、网络）的成本数据。它与多个云服务提供商的 API 进行集成，以获取计费数据，并可配置用于本地 Kubernetes 集群的定价策略。OpenCost 是最初由 Kubecost 构建并仍被其使用的成本分配引擎，但也可以单独使用。OpenCost 的成本分配数据可以导出为 CSV 文件或导

入到 Prometheus 进行进一步分析和可视化。我们的团队正在密切关注像 OpenCost 和 Kubecost 这样的工具的发展，这些工具可以为采用 Kubernetes 的产品和平台团队提供成本可见性。然而在目前的阶段，我们发现 OpenCost 在某些工作负载（如数据管道中经常使用的短期 Spot 实例）上尚不完全适用。

79. OpenRewrite

评估

我们已经看到了一些代码智能工具的使用案例：例如把一个广泛使用的库迁移到新的 API 版本，了解一个库中刚发现的漏洞对整个企业的影响，以及对从同一模板创建的多个服务应用更新时。在这一领域，Sourcegraph 仍然是一个很受欢迎的工具。OpenRewrite 是另一个我们想提及的工具。我们的团队已经在 Java 中使用它解决特定的问题，比如更新用入门套件创建的服务。目前它仍在持续拓宽覆盖的语言和使用案例。我们喜欢它附带的变革方案，这些方案描述了需要进行的更改，例如用于跨版本迁移常用框架。重构引擎、捆绑方案和构建工具插件都是开源软件，这使得团队在需要时可以更容易地使用 OpenRewrite。代码智能工具都是基于将源代码解析为抽象语法树（AST），这些工具将如何受到大语言模型领域快速发展的影响，我们拭目以待。

80. OrbStack

评估

OrbStack 是在 macOS 上运行 Docker 容器的一种方式；我们的开发人员发现，与 Docker Desktop 和 Colima 相比，它更轻量、更快速并且更容易部署和使用。这个工具仍在开发阶段，所以目前功能较少，但其简洁和速度已经显示出了它的巨大潜力。您也可以使用 OrbStack 在 macOS 上创建和管理 Linux 虚拟机。

81. Pixie

评估

Pixie 是一个用于 Kubernetes 原生应用程序的可观察性工具。它通过利用 eBPF 从多个 数据源 自动地采集遥测数据，以一种有趣的方式实现了可观察性。收集到的遥测数据被本地存储到每个节点上，并通过其控制平面 API 进行集中处理。总的来说，我们认为 Pixie 在 Kubernetes 生态系统中用于可观察性是值得评估的。

82. Tabnine

评估

Tabnine 是目前炙手可热的编程助手领域的有力竞争者之一。它提供了内嵌的代码补全建议，以及直接在 IDE（Integrated development environment，集成开发环境）中进行对话的能力。与 GitHub Copilot 类似，Tabnine 的出现远远早于现在这个各家纷纷大肆炒作的时期，也因此成为了该领域中最成熟的产品之一。与 Copilot 不同的是，Tabnine 使用的模型仅在获得授权的代码上进行训练，并提供了一个可以自行托管的版本，供担心其代码片段会被发送给其他第三方服务的组织使用。Tabnine 既提供有受限制的免费版本，也提供付费版本，后者会有更全面的建议，还提供了一种使用本地模型的模式（尽管功能较弱），供您在没有互联网连接的情况下使用。

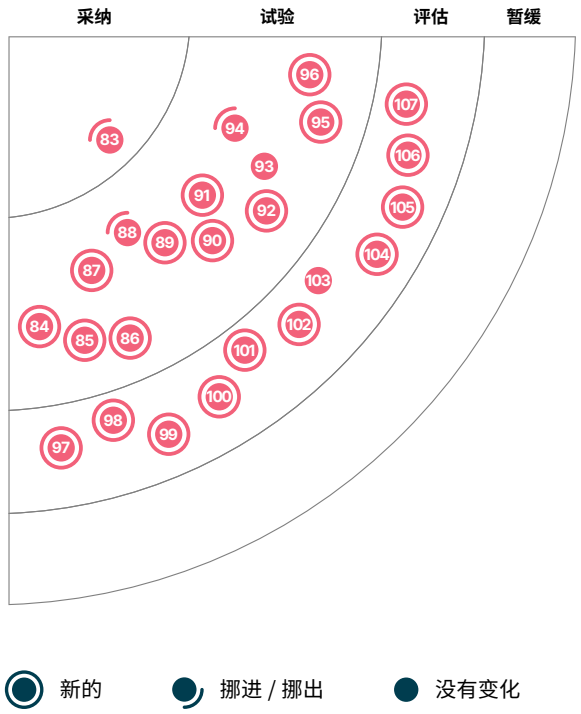
语言和框架

采纳
83. Playwright

试验
84. .NET Minimal API
85. Ajv
86. Armeria
87. AWS SAM
88. Dart
89. fast-check
90. Kotlin with Spring
91. Mockery
92. Netflix DGS
93. OpenTelemetry
94. Polars
95. Pushpin
96. Snowpark

评估
97. 基准配置文件
98. GGML
99. GPTCache
100. 语法性别 API
101. htmx
102. Kotlin Kover
103. LangChain
104. LlamaIndex
105. promptfoo
106. Semantic Kernel
107. Spring Modulith

暂缓
—



83. Playwright

采纳

使用 Playwright，您可以编写在 Chrome、Firefox 和 WebKit 中运行的端到端测试。 通过使用 Chrome 开发者工具（DevTools）协议（CDP），Playwright 可以提供新功能并消除 WebDriver 中出现的许多问题。 基于 Chromium 的浏览器直接实现了 CDP。不过，为了支持 Firefox 和 Webkit，Playwright 团队不得不向这些浏览器提交补丁，这有时可能会限制框架。

Playwright 的功能包括：内置自动等待，这使得测试更可靠、更易于理解；浏览器上下文，可让您测试跨标签页的持久会话是否正常工作；以及模拟通知、地理位置和黑暗模式设置的能力。 Playwright 为测试套件带来的稳定性给我们的团队留下了深刻印象，并且喜欢它可以并行运行测试以更快地获得反馈。 Playwright 的其他特色包括更好地支持懒加载和追踪。 尽管 Playwright 有一些局限性（例如，组件支持目前处于实验阶段），但我们的团队认为它是首选的测试框架，甚至在某些情况下会从 Cypress 和 Puppeteer 上迁移过来。

84. .NET Minimal API

试验

ASP.NET Core MVC 已经被证明是一种用于构建托管 APIs 的 Web 应用程序的强大而灵活的方法。然而，它的灵活性也带来了一定的复杂性，包括一些不明显的样板代码和约定。ASP.NET 提供的路由功能允许在单个应用程序中托管多个服务，但在当今的无服务器函数和可独立部署的微服务的世界中，这种灵活性可能会显得有些过剩。.NET Minimal APIs 在 .NET 生态系统中提供了一种简单的方法来实现 single-API 的 Web 应用程序。Minimal API 框架可以用仅仅几行代码就实现一个 API 端点。Minimal API 加入了新一代的 API 框架，包括 Micronaut、Quarkus 和 Helidon，这些框架针对轻量级部署和快速启动时间进行了优化。我们对 Minimal APIs 和 .NET 7 Native AOT 的结合很感兴趣，可以用于在无服务器函数中实现简单、轻量级的微服务。

85. Ajv

试验

Ajv 是一个流行的 JavaScript 库，用于根据使用 JSON Schema 定义的结构验证数据对象。对于验证复杂的数据类型，Ajv 既快速又灵活。它支持多种多样的 schema 特性，包括自定义关键字和格式。许多开源的 JavaScript 应用程序和库都使用它。 我们的团队使用 Ajv 在 CI 工作流中实现消费者驱动的契约测试，另外再配合其他工具一起通过 JSON Schema 生成模拟数据，功能非常强大。在 TypeScript 的世界中，Zod 是一种流行的替代方案，它可以用于定义 schema 和验证数据的声明式 API。

86. Armeria

试验

Armeria 是一个用于构建微服务的开源框架。我们的团队使用它来构建异步 API，我们非常喜欢用它的服务装饰器来解决跨切面关注点，例如分布式跟踪或断路器。该框架支持 gRPC 和 REST 流量的端口复用以及其他巧妙的设计选择。借助 Armeria，我们可以在现有 REST 代码库之上逐步添加新的 gRPC 功能，或者反过来，在现有的 gRPC 代码库之上逐步增加新的 REST 功能。总的来说，我们发现 Armeria 是一个灵活的微服务框架，具有多个开箱即用的集成功能。

87. AWS SAM

试验

AWS Serverless Application Model（SAM）是一款用于在 AWS 云基础设施上构建无服务器应用的开源框架。此前入选技术雷达条目的 Serverless Framework 作为一种在各个云服务商上部署无服务器服务的流行框架，主要用于基于 AWS Lambda 的服务。近来，AWS SAM 因其长足的发展而日益受到欢迎。我们的团队发现 AWS SAM 非常易于配置，并且我们还尝试将其用于测试和调试基于 AWS Lambda 的服务，包括开发时在本地执行 Lambda 函数。

88. Dart

试验

Dart 是由 Google 开发的编程语言，支持构建跨平台的应用程序，包括 Web 浏览器、WebAssembly、桌面和移动应用。它的采纳是由 Flutter 的主导地位推动的。在跨平台原生移动应用框架领域，Flutter 是一个流行的、使用 Dart 作为其核心语言的跨平台 UI 工具包。根据社区的反馈，Dart 从最初的版本开始不断演进，除了具备健壮的类型系统，还在 3.0 版本中添加了内置的 sound null safety。此外，Dart 的生态系统正在快速发展，拥有活跃的社区、丰富的可用库和工具，这使其对开发者非常具有吸引力。

89. fast-check

试验

fast-check 是一个为 JavaScript 和 TypeScript 设计的基于属性的测试工具，它能够自动生成测试数据，从而无需创建不同的测试即可仔细检查各种输入，这使得它更容易发现边缘场景。 我们的团队在后端测试中使用 fast-check 取得了不错的结果，归功于它良好的文档、易用性以及与现有测试框架的无缝集成，单元测试的效率得到了明显提高。

90. Kotlin with Spring

试验

五年前，我们将 Kotlin 移至了采纳阶段，如今我们的许多团队报告称，Kotlin 不仅成为了他们在 JVM 语言上的默认选择，而且已经几乎完全取代了 Java，成为他们编写软件的主要语言。与此同时，microservice envy 似乎正在减弱，我们注意到人们开始使用诸如 Spring Modulith 等框架，探索具有更大可部署单元的架构。我们已经知道有许多出色的 Kotlin 原生框架，并且在之前也提到过一些；然而在大部分情况下，Spring 框架的成熟度和功能丰富性是一个巨大的优势，并且我们已成功地结合使用 Kotlin 与 Spring，而并未遇到很大的问题。

91. Mockery

试验

Mockery 是一个成熟的 Golang 库，它能够生成接口的 mock 实现，并模拟外部依赖的行为。通过类型安全的方法生成期望的调用，并通过灵活的方式 mock 返回值，它使得测试能够专注于业务逻辑，而无需担忧外部依赖的正确性。Mockery 使用了 Go 生成器，且简化了测试套件中的 mock 的生成与管理。

92. Netflix DGS

试验

我们大多时候将 GraphQL 用于服务端资源聚合，并且使用多种技术实现服务端。对于使用 Spring Boot 开发的服务，我们的团队使用了 Netflix DGS，体验很好。Netflix DGS 基于 graphql-java 开发，并提供了许多 Spring Boot 编程模型的特性与抽象。因此使得开发 GraphQL 端点，以及与 Spring Security 等 Spring 特性集成就变得很容易。虽然 DGS 是用 Kotlin 编写的，但在 Java 上它也工作的很好。

93. OpenTelemetry

试验

我们使用 OpenTelemetry 作为解决方案已经有一段时间了，并且在之前的雷达中推荐试用。它能够在多个服务和应用之间无缝地捕获、检测和管理遥测数据，从而改善我们的观察堆栈。OpenTelemetry 的灵活性和与多样化环境的兼容性使其成为我们工具包中有价值的补充。目前，我们对最近发布的 OpenTelemetry Protocol（OTLP）特别感兴趣，该规范包括了 gRPC 和 HTTP 两种协议。这一协议标准化了遥测数据的格式和传输方式，促进了互操作性并简化了与其他监控和分析工具的集成。随着我们继续探索该协议的集成潜力，我们正在评估它对我们的监控和可观察性策略以及整个监控领域的长期影响。

94. Polars

试验

Polars 是 Rust 实现的一个内存运行的 DataFrame 库。 与其他 DataFrame 库（如 Pandas）不同，Polars 是多线程、支持惰性求值、并且并行操作安全的。 Polars 使用 Apache Arrow 格式作为内存模型，以高效实现分析操作，并实现与其他工具的互用性。 如果您熟悉 Pandas，就可以快速上手 Polars 的 Python 绑定。 基于 Rust 实现和 Python 绑定的 Polars 是一个高性能内存 DataFrame 库，可满足您的分析需求。 我们的团队在 Polars 方面继续拥有良好的体验，因此我们将其将其移至 “试验”。

95. Pushpin

试验

Pushpin 是一种反向代理工具，充当处理长连接（如 WebSockets 和服务器发送事件）的后端服务器与客户端之间的中介。它提供了一种终止来自客户端的长连接的方法，意味着系统的其他部分可以从这项复杂工作中解放出来。它能有效管理大量持久连接，并自动将其分配给多个后端服务器，从而优化性能和可靠性。我们在使用 Pushpin 来处理移动设备的 WebSockets 实时通信时获得了良好的体验，并且我们已经通过横向扩展成功使其服务于数百万台设备。

96. Snowpark

试验

Snowpark 是一个用于 Snowflake 大规模查询和处理数据的库。我们的团队使用它来编写可管理代码，用以与 Snowflake 中存储的数据进行交互 ——这类似于为 Snowflake 编写 Spark 代码。总的来说，它是一个引擎，能够将代码转换为 Snowflake 能够理解的 SQL。您在构建应用程序时无需将 Snowflake 中待处理的数据移动到您代码运行的地方。一个缺陷：单元测试的支持仍可优化；我们的团队通过编写其它类型的测试来弥补这个缺陷。

97. 基准配置文件

评估

不要与安卓基准配置文件相混淆，基准配置文件是指导提前编译的安卓运行时配置文件。基准配置文件会在发布前在开发机上创建，并随应用程序一起发布。因此，相较于依赖云配置文件（一种较早的相关技术），基准配置文件会更快可用。通过在应用或库中分发基准配置文件，安卓运行时可以优化重要的代码路径，从而在下载或更新应用程序时改善新老用户的使用体验。根据其文档所提到的，创建基准配置文件相对简单并可以显著提高性能（最多 30%）。

98. GGML

评估

GGML 是一个机器学习的 C 语言库，它支持 CPU 推理。它定义了一种分布式大语言模型（LLMs）的二进制格式。为此，GGML 采用了量化技术，这种技术可以使 LLM 在用户的硬件上运行有效的 CPU 推理。GGML 支持多种量化策略（例如 4 位、5 位、以及 8 位量化），每种策略动都在效果和性能之间提供了不同的取舍。一种快捷地对使用这些量化模型的应用进行测试、运行和构建的方法是使用一个叫做 C Transformers 的 Python 绑定。它是一个 GGML 之上的 Python 封装，通过高级的 API 来消除推理的样板代码。我们已经在尝试使用这些库构建原型和实验。如果你正在考虑为你的组织搭建自托管式大语言模型，请慎重选择这些社区支持的库。

99. GPTCache

评估

GPTCache 是一个用于大型语言模型（LLM）的语义缓存库。我们认为需要在 LLM 前增设缓存层主要出于两种原因——通过减少外部 API 调用来提升整体性能，以及通过缓存近似响应来减少运营成本。不同于使用精确匹配的传统缓存方式，基于 LLM 的缓存解决方案需要对输入进行相似或相关匹配。 GPTCache 通过使用嵌入算法将输入转化为嵌入，再通过向量数据库对这些嵌入进行相似性搜索。这种设计有一个缺点，可能会导致缓存命中时遇到假阳性结果，或缓存未命中时遇到假阴性结果，因此我们建议你在构建基于 LLM 应用时，仔细评估 GPTCache。

100. 语法性别 API

评估

在许多语言中，性别的表现都比英语更为明显，且词语会根据性别发生变化。例如，称呼用户时，可能需要对词语进行变形，但通常的做法是默认使用男性形式。有证据表明，这会对人的表现和态度产生负面影响 —— 当然，这也是不礼貌的。使用性别中立语言的变通办法往往显得笨拙。因此，正确地称呼用户是首选。借助 Android 14 中引入的语法性别 API，安卓开发者们现在可以更容易地做到这一点。

101. htmx

评估

htmx 是一款轻量简洁的 HTML UI 库，近期突然迅速走红。在技术雷达讨论期间，我们发现其前身 intercooler.js 早在十年前就已存在。与其他越来越复杂的预编译 JavaScript/TypeScript 框架不同，htmx 鼓励直接使用 HTML 属性来实现诸如 AJAX、CSS transitions、WebSockets 和服务器发送事件等操作。从技术角度来看，htmx 并不复杂，但它的流行让人想起早期网页中超文本的简洁风格。该项目的网站上还提供了一些关于超媒体和网页开发的深入（且有趣）的文章，这显示出 htmx 团队对其目标和理念进行过认真思考。

102. Kotlin Kover

评估

Kotlin Kover 是一个专为 Kotlin 设计的代码覆盖率工具集。它支持 Kotlin JVM 、Multiplatform 和 Android 工程。代码覆盖率的重要性在于能够凸显未经测试的代码片段，从而增强软件的可靠性。随着 Kover 的发展，它因能够生成为 Kotlin 量身定做的、全面的 HTML 和 XML 报告而引人注目。对于深度使用 Kotlin 的团队，我们建议您评估 Kover，以助力您提高代码质量。

103. LangChain

评估

LangChain 是一个利用大语言模型（LLM）构建应用程序的框架。要构建实用的 LLM 应用，需要将其与用户或领域的特定数据结合起来，这些数据不属于训练数据的一部分。LangChain 利用提示管理、链式、代理 和文档加载器等功能填补了这一空白。如提示模板和文档加载器等组件的好处是可以加快产品上市速度。尽管 LangChain 是实施检索增强生成（Retrieval-Augmented Generation）应用程序和 ReAct 提示工程 模式的热门选择，但它也因难以使用和过于复杂被批评。当您为 LLM 应用程序选择技术栈时，可能需要在这个快速发展的领域继续寻找类似的框架（如 Semantic Kernel）。

104. LlamaIndex

评估

LlamaIndex 是一个旨在促进私有或领域特定数据与大语言模型（LLMs）集成的数据框架。它提供了从各种数据源获取数据的工具，包括 API、数据库和 PDF，然后将这些数据结构化为 LLMs 能够轻松消费的格式。通过各种类型的“引擎”，LlamaIndex 使得对这些结构化数据进行自然语言交互变得可能，从而使其可用于从基于查询的检索到对话界面等各种应用。与 LangChain 类似，LlamaIndex 的目标是加速跟大语言模型应用的开发，但它更多地采用了数据框架的方法。

105. promptfoo

评估

promptfoo 是一款测试驱动的 prompt engineering。 在应用程序中集成 LLM 时，调整提示词为生成最佳回答并保证输出的一致性，往往会耗费大量时间。你可以将 promptfoo 作为 CLI 和库使用，根据预定义的测试用例

对提示词进行系统测试。测试用例和结果断言则可通过简单的 YAML 配置文件完成设置。 这个配置文件包含需要测试的提示词、模型提供者、断言以及将会在提示词中被替换的变量值。promptfoo 支持多种断言，包括相等性、JSON 结构、相似性、自定义函数检查，甚至支持使用 LLM 对模型输出结果分级。如果你想对提示词和模型质量进行自动化反馈，请务必体验 promptfoo。

106. Semantic Kernel

评估

Semantic Kernel 是微软 Copilot 产品套件中的一个核心组件的开源版本。它是一个 Python 库，与 LangChain 类似，它可以帮助你在大语言模型（LLMs）之上构建应用程序。Semantic Kernel 的核心概念是计划器，它可以帮助你构建由 LLM 驱动的代理，这个代理可以为用户创建一个计划，然后在各种插件的帮助下逐步执行这个计划。

107. Spring Modulith

评估

尽管我们是微服务（microservices）的早期倡导者，并看到该模式在无数系统上取得了成功，但我们也看到微服务被误用和滥用，这通常是 microservice-envy（microservice envy）导致的。与其从头开始构建一个由单独部署的进程组成的新系统，我们通常建议从一个精心设计的单体应用开始，并且仅当应用程序达到一定规模时，才将其分解为可单独部署的单元，此时微服务的好处才能超越分布式系统所固有的额外复杂性。最近，我们看到人们对这种方法重新产生了兴趣，以及对什么构成了精心分解的整体有了更详细的定义。Spring Modulith 是一个框架，它以一种使代码在适当时候更容易拆分成微服务的方式来组织代码。它提供了一种模块化代码的方法，使领域和限界上下文的逻辑概念与文件和包（package）结构的物理概念保持一致。这种对齐方式使得在必要时重构单体架构以及单独测试领域变得更加容易。Spring Modulith 提供了一种进程内事件机制，有助于进一步解耦单个应用程序中的模块。 最重要的是，它与 ArchUnit 和 jmolecules 集成，可以自动验证其领域驱动的设计规则。

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 18 个国家 / 地区的 51 个办公室拥有超过 11,500 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.