技术雷达

有态度的前沿技术解析

/thoughtworks

第 27 期

关于技术雷达 3
雷达一览 4
贡献者 5
本期主题 6
The Radar 8
技术 11
平台 19
工具 26
语言和框架 33

关于技术雷达

Thoughtworker 酷爱技术。我们为每个人建造技术，研究技术，测试技术，开源技术，书写技术，并不断致力于改进技术。我们的使命是支持卓越软件并掀起 IT 革命。我们创建并分享 Thoughtworks 技术雷达就是为了支持这一使命。由 Thoughtworks 中一群资深技术领导组成的 Thoughtworks 技术顾问委员会创建了该雷达。他们定期开会讨论 Thoughtworks 的全球技术战略以及对行业有重大影响的技术趋势。

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。

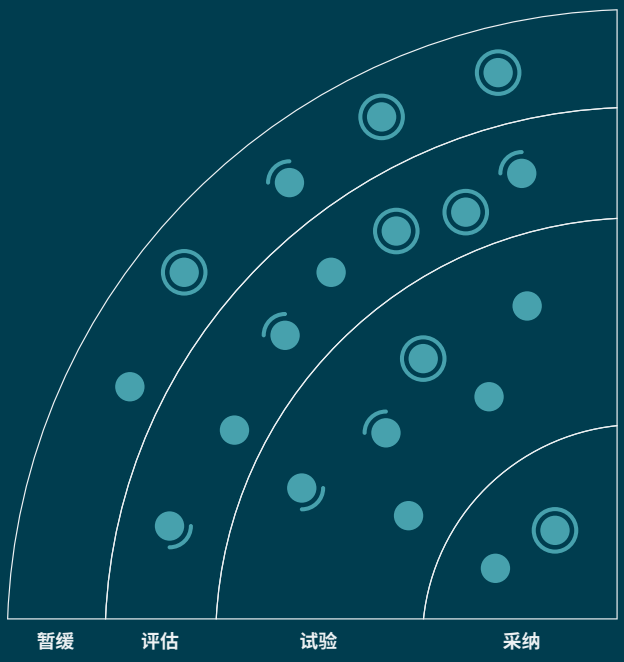
我们建议您探究这些技术以了解更多细节。技术雷达的本质是图形性质，把各种技术项目归类为技术、工具、平台和语言和框架。如果技术可以被归类到多个象限，我们选择看起来最合适的一个。我们还进一步将这些技术分为四个环以反映我们目前对其的态度。

想要了解更多技术雷达相关信息，请点击：
http://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 17 名高级技术专家组成。TAB 每年召开两次面对面会议，每两周召开一次电话会议。技术顾问委员会由 Thoughtworks 首席技术官 Rebecca Parsons 组建。

技术雷达由 Thoughtworks 技术顾问委员会（TAB）创建。本期技术雷达的创建基于 Thoughtworks 技术顾问委员会（TAB）于 2022 年 9 月在西班牙巴塞罗那的线下会议。

Rebecca Parsons (CTO)

Martin Fowler (Chief Scientist)

Bharani Subramaniam

Birgitta Böckeler

Brandon Byars

Camilla Falconi Crispim

Erik Dörnenburg

Fausto de la Torre

Hao Xu

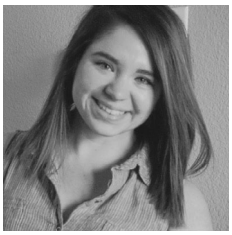
Ian Cartwright

James Lewis

Marisa Hoenig

Mike Mason

Neal Ford



Perla Villarreal

Scott Shaw

Shangqi Liu

中国区技术雷达汉化组：

李辉 | 徐培 | 杨光 | 杨洋 | 边晓琳 | 程显通 | 邓奕 | 樊卓文 | 符雨菡 | 高晓明 | 高语越 | 郭晨 | 何蜜 | 何宇哲 | 胡钰涵 | 黎鹏 | 李陈泽 | 李光正 | 李嘉骏 | 李思远 | 李岩 | 梁晶 | 刘鑫 | 娄麒麟 | 卢冠昀 | 马宇航 | 孟然 | 秦睿 | 史靖雅 | 孙郁俨 | 田鹏 | 田彪 | 王梦蛟 | 王鹏熹 | 王启瑞 | 王乔阳 | 王薇 | 熊欣 | 许家文 | 杨君君 | 杨旭东 | 杨阳 | 杨召 | 余亚彬 | 张广洁 | 张基 | 张丽 | 张真 | 郑茗蔓 | 朱雪晴

本期主题

机器学习的主流化

机器学习（Machine Learning）曾是一片只有一小部分拥有相关工具和资源的幸运儿才能发挥创造的领域。幸运的是，随着各种规模设备计算能力的增长，开源工具的出现，更严格的对隐私和个性化信息保护意识以及法规要求，造就了一个蓬勃发展的生态系统，我们看到机器学习也逐渐成为主流。联邦学习等技术能够使 ML 模型为敏感信息提供隐私保障。TinyML 可以让模型在资源受限的设备上执行，将推理转移到边缘，这既可以释放资源，又可以提高敏感数据的隐私性。Feature Stores 为应用程序开发的模型—视图—控制器设计模式提供了类似的好处，使得数据整理、模型训练和推理之间的分离更为清晰。诸如 Stable Diffusion 之类的公开可用模型既突出了机器学习的惊人能力，也突出了对源数据和道德的关注。机器学习组件也比以往任何时候都更容易连接在一起，从而使得通过自定义业务模型和功能强大的通用模型来创造性的构建机器学习体验和解决方案更为容易。我们对这一领域的新功能表示赞赏，并热切期待未来的进步。

“平台即产品”的力量

在我们的技术雷达会议期间，“平台”仍然是最常提到的词语之一，因为这个概念在业界非常普遍。它以许多不同的表现形式出现，包括聚焦业务或领域的平台，还有基础设施或者开发者体验平台。从根本上说，组织在所有平台上遇到问题或感到失望的根因是没有正确地将它们作为产品对待。例如，许多旨在供开发者使用的平台缺乏在其它类型产品中所期望的用户调研与上下文分析。平台所有者需要验证他们对开发者需求的假设，并响应实际的使用模式。像任何好的产品一样，平台需要持续的支持。它必须发展并适应开发者不断变动的需求。此外，像项目经理和业务分析师等角色所需的经常与传统应用程序的范围不同。“平台即产品”的隐喻仅当作为一种实践而不是一个时髦短语而被完全采纳时才会生效。

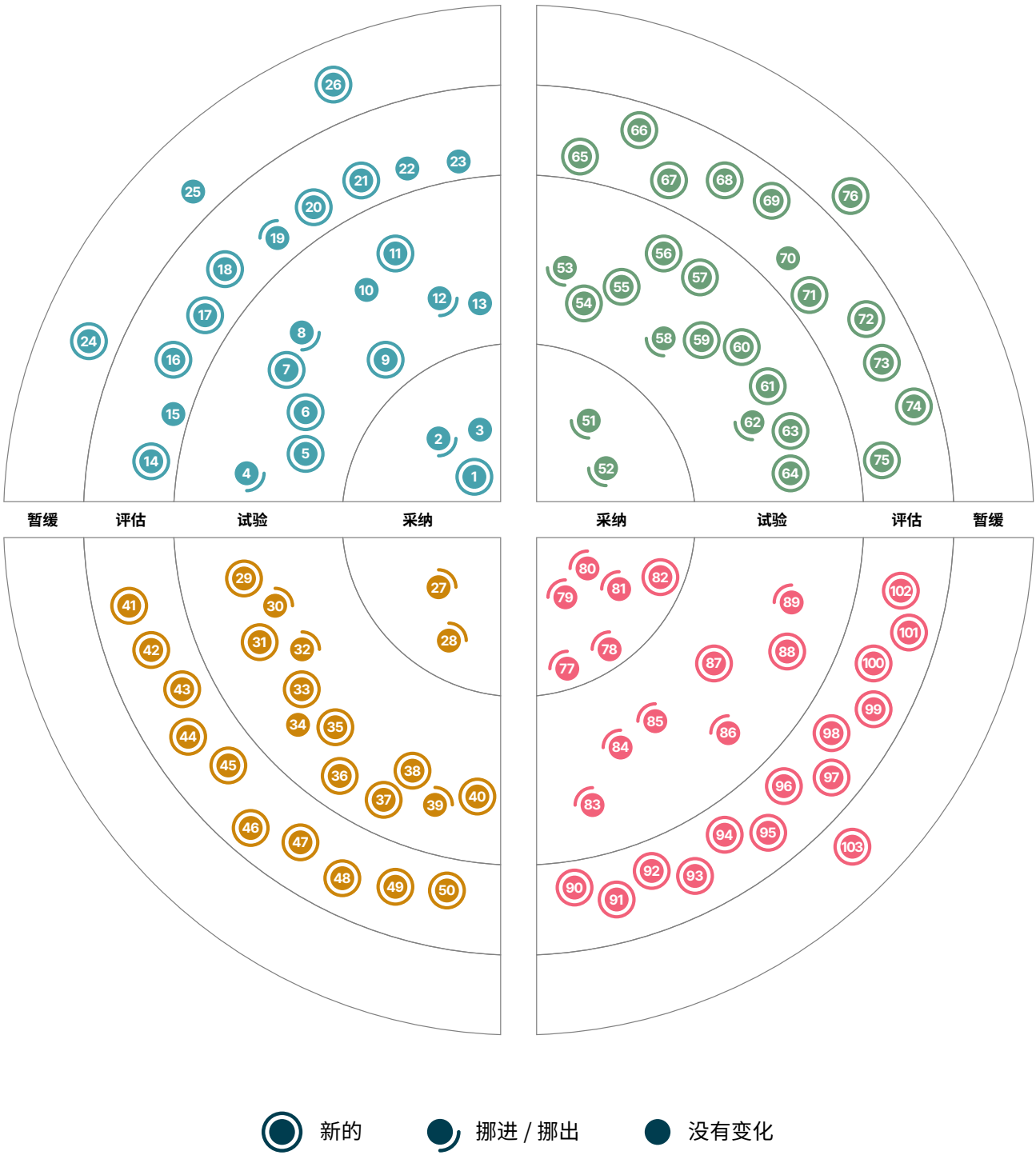
将数据所有权移至边缘节点

我们都知道，任何形式的中心化都会导致限制、瓶颈、和不必要的暴露。所以我们一直在努力寻找打破这些耦合点的新方法，这些方法也在我们技术雷达的更新中被重点提及。而基于无冲突复制数据类型（CRDTs）的研究，使得数据应用可以不使用中心化的数据库，这种本地优先的软件 / 应用技术鼓励开发者们去考虑围绕点对点的数据构建应用而不是使用中心化的数据库。正如机器学习的主流化主题中所展示的一样，将数据所有权移至边缘节点能让开发者更好地利用不同设备的功能，例如面容识别等的许多功能可以在边缘节点进行，保持所用的数据永久保存在本地。

移动端也应该模块化

软件工程师们已经了解到主要围绕领域概念和业务功能构建应用架构的价值。虽然根据领域逻辑划分 UI 代码的技术问题仍然很重要，但它已经不是首要问题了。随着移动应用的成熟，它们通常会变得越来越大，有时会发展为所谓的超级应用程序。这些应用包含许多服务，完全可以视为一个个平台。虽然有些应用没有那么大，但多年来积累的很多功能通常也可以分解为各种模块，许多公司发现移动应用也能从同样的模块化方法中受益。模块化的应用程序可以由多个自主团队编写，这带来了许多有据可查的好处。然而会增加额外复杂度的是，满足应用商店的上架要求，并且支持原生 iOS，Android 以及基于 web 的版本，还得进行微调以适应每种版本的特点。虽然对于这些移动开发固有的独特难题，我们看到了更好的框架支持，但总的来说，尽管有好处，许多组织很难将模块化方法引入到移动开发中。

The Radar



The Radar

技术

采纳

1. 生产路径图
2. 团队的认知负载
3. 威胁建模

试验

4. BERT
5. 组件视觉回归测试
6. 设计令牌
7. 用于测试收发邮件的虚拟邮箱服务
8. 联邦学习
9. 增量开发人员平台
10. 移动微前端
11. CI/CD 流水线的可观测性
12. SLSA
13. 软件物料清单

评估

14. 碳效能作为软件架构的特性
15. CUPID
16. GitHub 推送保护
17. 本地优先应用程序
18. 指标库
19. 服务器端驱动 UI
20. SLI 和 SLO 定义为代码
21. 模型测试的合成数据
22. TinyML
23. 可验证凭证

暂缓

24. 没有“使用原生的远程工作方法”的卫星式工人
25. 默认选择 SPA
26. 肤浅的云原生

平台

采纳

27. Backstage
28. Delta Lake

试验

29. AWS 数据库迁移服务
30. Colima
31. Databricks Photon
32. DataHub
33. DataOps.live
34. eBPF
35. Feast
36. Monte Carlo
37. Retool
38. Seldon Core
39. Teleport
40. VictoriaMetrics

评估

41. Bun
42. Databricks Unity Catalog
43. Dragonfly
44. Edge Impulse
45. GCP Vertex AI
46. Gradient
47. IAM Roles Anywhere
48. Keptn
49. OpenMetadata
50. OrioleDB

暂缓

—

The Radar

工具

采纳

51. Great Expectations
52. k6

试验

53. Apache Superset
54. AWS Backup 文件库锁定
55. AWS Control Tower
56. Clumio Protect
57. Cruft
58. Excalidraw
59. Hadolint
60. Kaniko
61. Kusto 查询语言
62. Spectral
63. Styra Declarative Authorization Service
64. 监视项目构建的 xbar

评估

65. Clasp
66. Databricks Overwatch
67. dbtvault
68. git-together
69. Harness 云服务成本管理
70. Infracost
71. Karpenter
72. Mizu
73. Soda Core
74. Teller
75. Xcode Cloud

暂缓

76. 在线格式化或代码解析服务

语言和框架

采纳

77. io-ts
78. Kotest
79. NestJS
80. React Query
81. Swift Package Manager
82. Yjs

试验

83. Azure Bicep
84. Camunda
85. Gradle Kotlin DSL
86. Jetpack Media3
87. Ladle
88. Moshi
89. Svelte

评估

90. Aleph.js
91. Astro
92. BentoML
93. Carbon Aware SDK
94. Cloudscape
95. Connect
96. 跨设备 SDK
97. Cypress 组件测试
98. JobRunr
99. Million
100. Soketi
101. Stable Diffusion
102. 合成数据保险库

暂缓

103. Carbon

技术

采纳

1. 生产路径图
2. 团队的认知负载
3. 威胁建模

试验

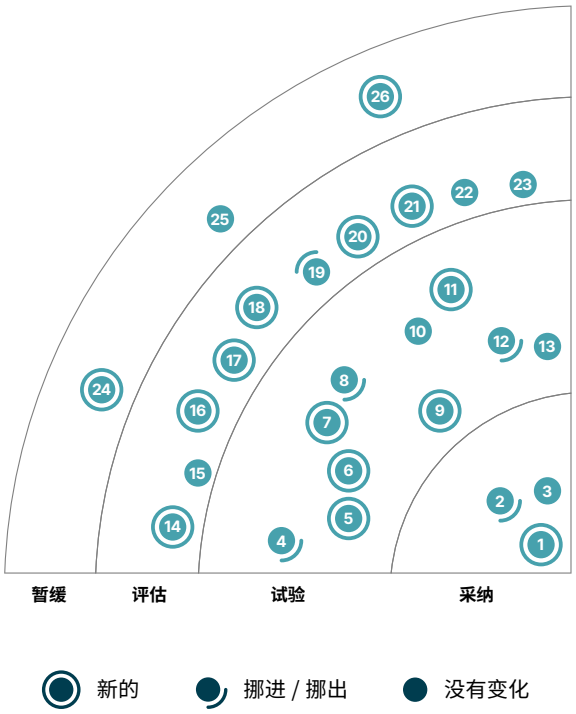
4. BERT
5. 组件视觉回归测试
6. 设计令牌
7. 用于测试收发邮件的虚拟邮箱服务
8. 联邦学习
9. 增量开发人员平台
10. 移动微前端
11. CI/CD 流水线的可观测性
12. SLSA
13. 软件物料清单

评估

14. 碳效能作为软件架构的特性
15. CUPID
16. GitHub 推送保护
17. 本地优先应用程序
18. 指标库
19. 服务器端驱动 UI
20. SLI 和 SLO 定义为代码
21. 模型测试的合成数据
22. TinyML
23. 可验证凭证

暂缓

24. 没有“使用原生的远程工作方法”的卫星式工人
25. 默认选择 SPA
26. 肤浅的云原生



1. 生产路径图

采纳

尽管生产路径图自从持续交付被编著时在 Thoughtworks 就是一种近乎普遍的实践，但是我们经常会遇到一些不熟悉生产路径图实践的组织。这一活动经常由一群跨功能团队的人在工作坊中完成（包括参与设计、开发、发布与运营软件的所有人），他们围在同一张白板前面（或者是一个虚拟的等价物）。首先，按照顺序把流程的步骤罗列出来，从开发阶段一直到发布上产的所有路径。然后，主持一个会议来获取更多的信息和痛点。我们最常见的技术是基于价值流图，尽管有很多具有同等价值的流程图变种。这项活动经常会让参与者眼界大开，因为他们识别出了延迟，风险和不一致的地方，并且可以用可视化的陈述来持续改进构建和部署的流程。我们认为这是一项很基本的技术，所以当我们发现在之前的技术雷达中我们没有提到它的时候，我们感到非常惊讶。

2. 团队的认知负载

采纳

在为组织重新设计业务敏捷性和交付速度时，团队交互是一个关键概念。这些交互将反映在正在构建的软件中（参见康威定律），并表明团队可以如何有效地自主为客户提供价值。我们的建议是关注团队的设计和互动方式。因为我们相信组织设计和团队交互会随着时间而发展，所以我们认为衡量和跟踪团队的认知负载尤为重要，这可以让你理解团队构建、测试和维护其服务的难易程度。我们一直在使用一个模板来评估团队的认知负载，该模板基于高效能团队模式（Team Topologies）作者的想法。

在与客户沟通和重新设计组织时，应用本书的概念所产生的积极影响给我们留下了深刻的印象。作者推荐了一种简单但强大的组织设计方法，仅确定四种类型的团队和三种交互模式；这有助于减少组织内的歧义，并为团队、利益相关者和领导层提供一个通用词汇来描述和设计团队的工作。为了实施组织设计变更，我们设计了理想的未来团队拓扑结构，应用任何技术 / 人员限制（即员工不足），然后形成最终的结构。这使我们能够更好地为客户提供建议，并通过比较现有 / 未来的团队结构来预测我们是否确实在改善认知负载。

3. 威胁建模

采纳

我们继续推荐团队实施威胁建模——一系列有助于在开发过程中发现潜在威胁并对其进行分类的技术，但是我们想要强调的是，这件事不是只在项目开始时做一次就能一劳永逸的，团队需要避免 security sandwich 现象。这是因为，在任何软件的整个生命周期中，由于外部事件以及需求和架构的调整，可能会出现新的威胁，而现有的威胁将继续发展。这就意味着，威胁建模需要定期重复，重复的频率视情况而定，需要综合考虑诸多因素，例如执行的成本和对业务的潜在风险等。如果结合其他技术使用，例如建立跨功能的安全需求来发现项目所采用的技术有什么公共风险，以及使用自动化安全扫描，这时威胁建模将变得非常有用处。

4. BERT

试验

自我们上次在技术雷达中谈到 BERT（来自变换器的双向编码器表征量）之后，我们团队已经成功地在一些自然语言处理（NLP）项目中使用了它。在其中一个项目里，当我们把默认的 BERT 分词器换成一个经过领域训

练的词块（word-piece）分词器，再去处理那些含有品牌名称或者维度之类名词的查询任务时，我们看到了显著的改进。虽然 NLP 有一些新的转换模型，但 BERT 凭借优秀的文档以及活跃的社区支持，更加通俗易懂，而且我们也一直发现它在企业级 NLP 背景下非常有效

5. 组件视觉回归测试

试验

视觉回归测试是很有用的强大工具，值得被您收进工具箱。但是在整个页面上使用视觉回归测试的成本非常高。我们看到随着 React 和 Vue 等基于组件的框架逐渐兴起，组件视觉回归测试也越来越流行。这项技术能确保在应用程序中不会引入非必要的视觉元素，从而很好的维持了投入和产出之间的平衡。在我们的实践里，组件视觉回归测试很少误测，而且有助于改善架构风格，通过和 Vite 以及 webpack 的模块热替换（HMR）功能共同使用，还可以视之为测试驱动开发应用于前端开发领域的范式转变。

6. 设计令牌

试验

当面临如何跨多种尺寸设备和平台一致地使用设计系统的挑战时，Salesforce 的团队提出了设计令牌这个概念。令牌将诸如颜色和字体等值存储在一个中心位置。这使得将选项与决策分开成为可能，并且显著改善了团队之间的协作。设计令牌并不是新事物，但随着 Tailwind CSS 和 Style Dictionary 等工具的引入，我们看到设计令牌被更频繁地使用。

7. 用于测试收发邮件的虚拟邮箱服务

试验

使用真实电子邮箱的测试账户，或使用 SMTP（简单邮件传输协议）服务器进行测试仍然是较为常见的软件测试方法。然而，使用真实的服务进行测试会带来测试邮件将被发送到真实的人的风险，也常常会使自动集成测试变得复杂。我们完全可以使用虚拟 SMTP 服务器来测试邮件发送，它记录了发送电子邮件的请求，但并没有实际发送。在这个领域存在多个开源工具，比如 fake-smtp-server，它提供了 Web UI 来展示邮件，便于可视化测试；以及 mountebank，它提供了 REST API 来获取已发送的邮件，便于集成测试。我们建议探索这种技术，以减少风险，提高测试效率。

8. 联邦学习

试验

我们看到许多客户项目正在使用联邦学习（ML）。传统上机器学习模型训练时需将数据放在集中运行模型训练算法的地址。在隐私角度上，这存在问题，特别是当训练数据包括敏感或者可用于身份识别信息时；用户也许不愿意分享数据，当地的数据保护法律也可能不允许我们将数据转移至一个中心化的位置。联邦学习是一个去中心化的技术，它使模型可以在大量不同来源的数据集上训练，并让数据保持在远端，例如用户的设备上。尽管网络带宽和设备的算力限制目前仍是这项技术重大的挑战，但是我们喜欢联邦学习的思路，让用户可以完全控制自己的个人信息。

9. 增量开发人员平台

试验

自从 2017 年以来，我们几乎每一期技术雷达都写过关于开发人员平台以及如何构建它们的内容。与此同时，团队拓扑学这本书还很好地描述了一个理想的平台，即用“自服务的 API、工具、服务和知识”来支持开发人员。然而，我们经常看到团队对于实现这种平台的愿景操之过急。事实上，构建一个增量开发平台才是关键所在。

团队拓扑学建议在任何特定阶段，都努力去实现一个所谓的“最小可行平台”，这个平台的第一版甚至可以是一系列的 wiki 文档。下一个增量可以通过提供模版或允许成员创建请求来提高服务水平。进一步的增量可以引入自服务 API，但前提是有价值。简而言之，尽管我们小心地反对全量工单驱动的平台运营模式，但从零到自服务是另一个极端。调整你自己的步调，将产品管理思维应用于内部平台并逐步增量地建立它。

10. 移动微前端

试验

自从 2016 年在技术雷达中介绍微前端以来，我们已经看到 Web UI 广泛地采用它们。然而，最近我们看到一些项目把这种架构风格也拓展到了移动微前端。当应用程序变得足够庞大与复杂时，就有必要将开发工作分配给多个团队。这提出了围绕团队自治、仓库结构与集成框架的许多挑战。过去，我们提到过 Atlas 与 BeeHive，但是这些框架未能获取关注而且也不再处于活跃开发状态。最近的方法包括用于将多个团队的工作集成到单个应用程序中的 Tuist 或 Swift 包管理器。但根据我们的经验，团队最终经常会实现自己的集成框架。虽然我们毫无疑问看到了在扩充移动开发团队规模时对模块化的需求，但是对微前端的需求却不太确定。这是因为尽管微前端意味着团队与页面或组件之间的直接对应关系，但是这种结构却有可能最终导致业务领域上下文的职责模糊，由此增加团队认知负载。我们的建议是遵循良好、整洁的应用程序设计，当规模扩大为多个团队时采纳模块化，仅当模块与业务领域高度对齐时才采用微前端架构。

11. CI/CD 流水线的可观测性

试验

可观测性实践已经将对话从监测通俗易懂的问题转换为帮助解决分布式系统中的未知问题。通过应用 CI/CD 流水线的可观测性，我们已经看到在传统生产环境之外采纳这种立场以帮助优化测试与部署瓶颈的成功。复杂的流水线会在运行太慢或遭受非确定性影响时给开发者带来摩擦，进而减少关键反馈循环并且阻碍开发者效率。此外，它们作为关键部署基础设施的角色在快速部署周期中产生了压力点，就像一些组织在应对最近的 log4shell 漏洞时所发生的那样。追踪的概念能够很好地转移到流水线上：子跨度抓取构建每个阶段的信息，而不是抓取服务调用级联。在分布式架构中用于分析调用流的瀑布图同样可以有效地帮助我们识别流水线瓶颈，甚至是那些扇入扇出的复杂流水线。这使得针对性大幅提高的优化工作成为可能。尽管此项技术适用于任何追踪工具，但是 Honeycomb 支持一个名为 buildevents 的工具，其有助于抓取流水线追踪信息。一种替代方案，即抓取已经由 CI/CD 平台暴露的信息，由 buildviz（由一位 Thoughtworker 构建并维护）采纳，允许在不更改步骤配置文件自身的情况下进行相似的调查。

12. SLSA

试验

随着软件复杂性的不断增加，软件依赖项的威胁路径变得越来越难以守护。软件工件供应链层级，又称 SLSA（读作“salsa”），是一个由联盟组织策划的，为组织机构提供防范供应链攻击的指南集。该框架衍生于一个 Google 多年来一直使用的内部指南。值得称赞的是，SLSA 并没有承诺提供“银弹”，即仅使用工具确保供应链安全的方法，而是提供了一个基于成熟度模型的具体威胁和实践的清单。这个威胁模型是很容易理解的，其中包含了真实世界发生的攻击实例，并且要求文档中也提供了指南，帮助组织基于日渐增强的稳健性水平为其行动措施排定优先级，以改善他们供应链的安全态势。自我们第一次在技术雷达中提到 SLSA 以来，它已经通过示例，围绕软件认证增添了很多细节，以跟踪如构建出处等问题。我们的团队发现 SLSA 在具体执行指导和更高层次对供应链威胁的认知之间取得了良好的平衡。

13. 软件物料清单

试验

在保障系统安全性的压力不变且总体安全威胁不减的情况下，一个机器可读的软件物料清单（SBOM）可以帮助团队掌握他们所依赖的库中的安全问题。随着美国总统令的发布，业界对 SBOM 的概念和如何创建 SBOM 都有了清晰的认识，例如国家标准与技术研究院（NIST）也针对如何执行总统令提出了更详细的建议。现在我们已经具备了在项目中使用 SBOM 的生产经验，项目范围从小型企业到大型跨国公司，甚至是政府部门，而且我们确信它能为我们提供益处。更多的机构和政府部门应当考虑索取正在使用的软件的 SBOM。随着 Firebase Android BOM 等可以自动罗列应用软件 BOM 中依赖库的新工具的不断出现，这项技术也将持续强化。

14. 碳效能作为软件架构的特性

评估

可持续发展是一个需要企业持续关注的话题。在软件开发领域，可持续发展的重要性日益显著，并且我们也可以看到出现了很多不同的达成途径。例如，在构建软件碳足迹方面，我们建议对碳效能作为软件架构的特性进行评估。一个重视碳效能的架构会将碳效率作为架构设计和基础设施选型的考虑因素，以最大限度地减少能源消耗，进而实现减少碳排放。该领域的测量工具和开发建议也日趋成熟，使得开发团队可以将碳效能与性能、可扩展性、财务成本和安全性等其他因素一起进行考量。就像软件架构中的所有因素一样，碳效能应该被视为一种权衡。我们建议将碳效能视为构成架构的一整套质量属性中的一个附加特性。因为这类特性应由组织目标驱动并划分优先级，而不应该留给一小部分专家孤立地思考。

15. CUPID

评估

你应该如何编写好的代码？如何判断自己是否写了好的代码？作为软件开发者，我们总是在寻找一些自然易记的规则、原则和模式，以便在讨论如何编写简单的、易修改的代码时，我们有统一的语言和价值观。

Daniel Terhorst-North 最近尝试为好代码创建了一个类似于检查表的东西。他认为与其拘泥于像 SOLID 这样一套规则，不如使用一组特性作为目标。他设计出了名为 CUPID 的特性组，来描述为了写出“令人愉悦”的代

码，我们需要做出哪些努力：在该特性指导下的代码应该是可组合的，遵循 Unix 哲学的，可预测的，风格自然的以及基于领域的。

16. GitHub 推送保护

评估

意外泄露机密似乎是一个老生常谈的事故，也出现了像 Talisman 这样的工具来帮助解决这个问题。在此之前，拥有高级安全许可证的 GitHub Enterprise Cloud 用户可以对其帐户启用安全扫描，意外提交和推送的任何机密（API 密钥、访问令牌、凭据等）都会触发警报。GitHub 推送保护更深入了一步，并将其提前到了开发工作流程中，如果更改被推送的时候检测到有机密，则直接拒绝这次推送。这需要为组织进行配置，当然仅适用于许可证持有者，但欢迎提供额外的保护以防止泄露机密。

17. 本地优先应用程序

评估

在集中式应用程序中，服务器上的数据是可信单一数据源，对数据的任何修改都必须经由服务器完成，本地数据只是服务器数据的副本。这似乎是实现软件的多人协作自然且必然的选择。本地优先应用程序，或本地优先软件，是一组即能够实现多人协作，且可以使数据所有权本地化的软件设计原则。它优先使用本地存储和本地网络，而不是远程数据中心或云服务器。无冲突复制数据类型（CRDT）和点对点（P2P）网络等技术有可能成为实现本地优先软件的基础技术。

18. 指标库

评估

指标库，有时也被称作无头商业智能（BI），是一种将指标的定义与其在报表和可视化中的使用相解耦的中间层。在传统意义上讲，指标定义在商业智能（BI）工具的上下文中。但是这样的做法会导致重复与不一致，因为不同的项目组可能将指标用于不同的上下文中。通过指标库解耦指标的定义，便可以清晰且一致地在多个报表、可视化甚至嵌入式分析中复用指标。这并不是一项新技术；例如，Airbnb 在几年前就引入了 Minerva。不过，随着越来越多的工具开箱即用地支持指标库，指标库在数据和分析生态的影响力越来越大。

19. 服务器端驱动 UI

评估

服务器端驱动 UI 仍然是移动开发圈的一个热议话题，因为这项技术允许移动端开发者利用更快的变更周期，而不违反应用商店关于重新验证移动应用的任何政策。服务器端驱动 UI 将渲染分离到移动应用程序的一个通用容器中，而每个视图的结构和数据由服务器提供。这意味着对于过去那些需要经历一次应用商店发布之旅的修改，现在只需要简单改变服务器发送的响应数据即可实现。虽然一些非常大的移动应用团队已经利用这种技术取得了巨大的成功，但它也需要大量的投入来建立和维护一个复杂的私有框架。这样的投入需要一个令人信服的商业案例。在此之前，最好谨慎行事。我们的确陷入过某种过度配置的可怕困境，并没有真的获得预期的收益。但在 Airbnb 和 Lyft 等巨头的背书下，我们很可能会看到一些有用的框架出现，有助于降低这种复杂度。这一领域值得关注。

20. SLI 和 SLO 定义为代码

评估

自从谷歌首次将服务质量指标（SLIs）和服务质量目标（SLO）作为其网站可靠性工程（SRE）实践的一部分推广以来，Datadog、Honeycomb 和 Dynatrace 等观察性工具开始将 SLO 监控纳入其工具链。OpenSLO 是一个基于 Kubernetes 使用的 YAML 格式声明式、中立的规范语言新兴的，且允许将 SLI 和 SLO 定义为代码的标准。虽然该标准仍然很新，但我们看到了一些令人鼓舞的势头，比如 Sumo Logic 公司贡献了 slogen 工具来生成监控和仪表盘。我们对在代码中对 SLI 和 SLO 定义进行版本化的承诺，和将可观察性工具作为所部署服务的 CI/CD 流水线一部分这一更新感到兴奋。

21. 模型测试的合成数据

评估

在我们本期技术雷达的讨论中，出现了几个用于生成合成数据的工具和应用。我们发现，随着工具的成熟，模型测试的合成数据成了一项强大而且有广泛应用的技巧。在验证机器学习模型判别能力的过程里，合成数据虽然尚不能取代真实数据，但也有相当广泛的使用场景。例如，合成数据可以用于预防小概率事件下模型彻底失效，或者在不暴露个人隐私信息的前提下对数据流水线进行测试。在探索缺乏真实数据的边缘场景以及确认模型偏差时，合成数据也很有用处。有一些有助于生成数据的工具，例如 Faker 和 Synth 可以生成服从预期统计特性的数据，Synthetic Data Vault 等工具可以依照输入数据集特性来生成数据。

22. TinyML

评估

我们仍旧为 TinyML 这项技术和它在构建可以运行在低功耗和移动设备上的机器学习（ML）模型的能力而感到兴奋。时至今日，运行机器学习（ML）模型仍然需要高昂的计算成本，并且在某些情况下还需要使用专用硬件。虽然模型的创建仍然大致属于上述情况，但可以通过一种方式创建模型，使它们能够在小型、低成本和低功耗设备上运行。如果你一直在考虑使用 ML 但苦于计算能力或网络环境的限制而放弃，那么这种技术值得你去评估。

23. 可验证凭证

评估

当我们两年前第一次把它纳入雷达时，可验证凭证是一个有趣的标准，有着一些潜在的应用前景，但它并没有在爱好者群体之外广为人知。当涉及到将负责实施该标准的证书授予机构时，如州级政府等，情况更是如此。自疫情以来的两年后，随着对加密安全、尊重隐私和可由机器验证的电子凭证的需求的增长，政府开始意识到可验证凭证的潜力。我们现在开始看到可验证证书出现在我们与公共部门客户的合作中。W3C 标准以凭证持有者为中心，这与我们使用物理凭证时的经验相似：用户可以将其可验证凭证放在自己的数字钱包中，并在任何时候向任何人展示，而无需得到凭证发行者的许可。这种去中心化的方法也使用户能够更好地管理和有选择地披露自己的信息，这大大改善了数据隐私的保护。例如，在零知识证明技术的支持下，你可以在不透露你的生日情况下构建一个可验证凭证来证明你是一个成年人。值得注意的是，尽管许多基于可验证凭证的去中心化身份解决方案依赖于区块链技术，但区块链并不是所有可验证凭证实施的先决条件。

24. 没有“使用原生的远程工作方法”的卫星式工人

暂缓

术语“远程团队配置”并不只是描述一种配置；它包含了多种模式和口味。许多团队最近都在改变模式。他们正从新冠疫情下被迫采取的“每个人总是远程”的工作模式中走出来，转而采用（通常是轮流）卫星式工人的模式，即部分团队在同一地点办公，部分团队远程工作。我们看到他们中的许多人没有正确考虑这对工作方式意味着什么。没有“使用原生的远程工作方法”的卫星式工人回到了优先考虑同地办公的工作方式。在有卫星式工人的配置中，重要的是仍然默认使用“原生的远程工作方法”。例如，如果团队中在同一地点工作的人一起参加会议，他们仍然应该在各自的笔记本电脑上参与数字协作或会议聊天。团队需要意识到排斥他们的卫星工人和创造孤岛和排斥感的风险。如果你知道总是有至少一个卫星团队成员，那么默认的工作方式应该假定是远程的。

25. 默认选择 SPA

暂缓

团队搭建网站时会默认选择单页面应用（SPA）的普遍现象让我们担心人们甚至没意识到 SPA 原本只是一种架构风格时，就立即进行了项目的框架选型。SPA 会招致传统基于服务器的网站所不具备的复杂性：譬如搜索引擎优化，浏览历史管理，网站分析，首页加载时间等。我们需要适当地分析和考虑，来确定这种复杂性是出于业务需求还是用户体验，以此做出权衡。我们通常看不到团队去进行这种权衡分析，即使是在业务需求不能证明这种使用是合理的情况下，也盲目地接受了默认使用 SPA 的复杂性。事实上，我们已经开始注意到许多新的开发人员甚至都不知道有替代的方法，因为他们整个职业生涯都是在类似 React 这样的框架中度过的。我们相信，许多网站都会受益于服务端逻辑的简洁性，并且我们从例如 Hotwire 这种有助于减少用户体验差异的技术中受到了鼓励。

26. 肤浅的云原生

暂缓

“云原生”一词最初用于描述最大限度利用公有云托管的架构。例如由许多小型、无状态和协作流程组成的分布式架构，以及具有高度自动化的构建、测试和部署能力的系统。然而，我们注意到越来越多肤浅的云原生设计，简单地大量使用云供应商提供的专有服务，而没有注意到应用程序其实是大单体、脆弱且笨重的。要注意的是，无服务函数本身并不能使应用程序更具弹性或更易于维护。云原生实际上是架构设计，而非对于实现方式的选择。

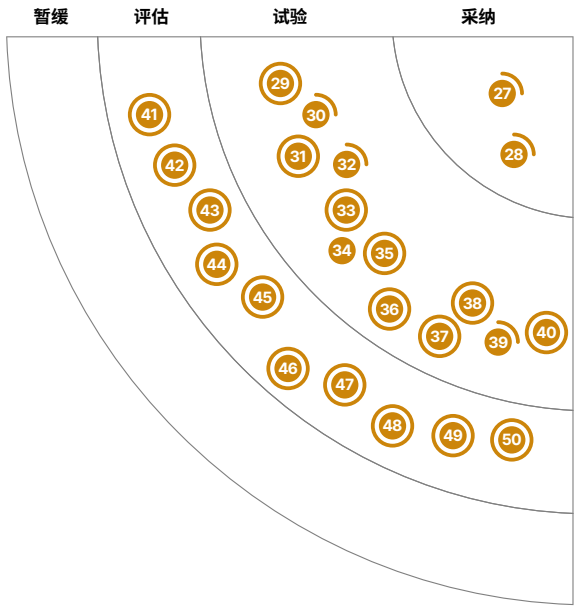
平台

采纳
27. Backstage
28. Delta Lake

试验
29. AWS 数据库迁移服务
30. Colima
31. Databricks Photon
32. DataHub
33. DataOps.live
34. eBPF
35. Feast
36. Monte Carlo
37. Retool
38. Seldon Core
39. Teleport
40. VictoriaMetrics

评估
41. Bun
42. Databricks Unity Catalog
43. Dragonfly
44. Edge Impulse
45. GCP Vertex AI
46. Gradient
47. IAM Roles Anywhere
48. Keptn
49. OpenMetadata
50. OrioleDB

暂缓
—



新的　挪进 / 挪出　没有变化

27. Backstage

采纳

在一个日渐数字化的世界里，大型组织怎样提高开发者的效率往往是资深领导们的核心关注点。总的来说我们已经足够意识到了开发者门户的价值，特别是 Backstage 的价值，我们非常愿意在采纳环中推荐它。Backstage 是由 Spotify 开发的一款能提升整个组织内的软件资产发现的开源开发者门户平台。它使用了存放在代码库中的 Markdown TechDocs 来追踪每个服务，这很好地平衡了中心化发现和分布式资产所有权的需求。Backstage 支持软件模版，这可以加速新项目的开发。它还支持插件架构，能够增强组织基础设施生态系统的可扩展性和适应性。Backstage 服务目录使用 YAML 文件来追踪组织生态系统中所有软件的所有权和元数据；它甚至可以让你追踪第三方的 SaaS 软件，通常来说这么做需要追踪的权限。

28. Delta Lake

采纳

Delta Lake 是由 Databricks 实现的开源存储层，旨在将 ACID 事务处理引入到大数据处理中。在使用了 Databricks 的 data lake 或 data mesh 的项目中，我们的团队更喜欢使用 Delta Lake 存储，而不是直接使用 AWS S3 或 ADLS 等文件存储类型。Delta Lake 此前一直是 Databricks 的闭源项目，最近成为了开源项目，并且可以在 Databricks 之外的平台使用。但是，我们只建议使用 Parquet 文件格式的 Databricks 项目将 Delta Lake 作为默认选择。Delta Lake 促进了需要文件级事务机制的并发数据读 / 写用例的发展。我们发现 Delta Lake 与 Apache Spark batch 和 micro-batch 的无缝集成 API 非常有用，尤其是其中诸如时间旅行（在特定时间点访问数据或还原提交）以及模式演变支持写入等功能，尽管这些功能有一些限制。

29. AWS 数据库迁移服务

试验

我们的许多团队已成功使用 AWS 数据库迁移服务（DMS）将数据迁移到 AWS 或从 AWS 迁移数据。在我们的一项数字化转型项目中，当我们将数据从 Microsoft SQL 服务迁移到 AWS 关系数据库服务（RDS）PostgreSQL 实例时，我们以几乎零宕机时间切换到了新系统。这次转型涉及许多部分，需要跨多专业团队进行规划和协调，然而对于数据迁移来说，我们对 DMS 非常满意。它自动管理所有必需资源的部署、管理和监控。经过多年的发展，DMS 已经很成熟，可以支持多种来源和目标数据库，我们也会继续支持它。

30. Colima

试验

Colima 正在成为 Docker Desktop 的一个流行的开源替代方案。它通过 Lima VM 的方式提供 Docker 容器运行时，在 macOS 上配置 Docker CLI 并处理端口转发和卷挂载。Colima 使用 containerd 作为容器运行时，这也是大多数托管 Kubernetes 服务采用的容器运行时，这一方案提升了开发与生产环境的一致性。通过 Colima，你可以轻松地使用和测试 containerd 的最新特性，例如容器镜像的懒加载。我们在项目中使用 Colima 已经取得了不错的效果。在使用 Kubernetes 时，我们也使用 nerdctl，这是一个与 Docker 兼容的 containerd CLI。由

于 Kubernetes 已经不再将 Docker 作为容器运行时，而且大多数托管服务（EKS、GKE 等）都在追随它的脚步，因此更多的人将会使用 containerd 这类原生工具，使得像 nerdctl 这样的工具更加重要。在我们看来，Colima 有着强大的潜力，并会成为 Docker Desktop 的首选替代方案。

31. Databricks Photon

试验

自 Databricks 9.1 LTS（长期支持版）开始，新运行时 Databricks Photon 开始可用，它是一个使用 C++ 重新从底层构建的替代品。现在我们的许多团队在生产中使用了 Photon，并对它的性能提升和因此带来的成本节约感到满意。实际使用中的提升和成本变化受到包括数据集大小和事务类型在内的多种因素影响，所以我们推荐在决定使用 Photon 前进行试验以获得数据进行比较，而非直接用于真实的工作负载。

32. DataHub

试验

自从我们第一次在技术雷达中提及数据的可发现性以来，LinkedIn 已经将 WhereHows 进化为 DataHub，一个通过可扩展的元数据系统实现数据可发现性的下一代平台。与爬取和拉取元数据不同，DataHub 采用了基于推送的模式。数据生态系统中各个组件，都可以通过 API 或者流（stream）向中心化的平台上发布元数据。这种基于推送的数据集成，将数据发现所有权从中心实体转移到各个团队，使他们对自己的元数据负责。因此，我们已成功将 DataHub 用于组织层面的元数据存储库和多种自维护的数据产品入口。当使用它时，请确保它足够轻量并避免让它滑坡成对共享资源的中心化控制系统。

33. DataOps.live

试验

DataOps.live 是一个可在 Snowflake 中实现环境自动化的数据平台。受 DevOps 实践的启发，DataOps.live 通过采用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，让你可以像对待任何其他 web 平台一样对待数据平台。你可以立即回滚更改而不会影响数据，或者从完全故障中恢复，并在几分钟或几小时而不是几天内重建新的 Snowflake 租户。我们的团队在使用 DataOps.live 的过程中体验良好，因为它让我们能够在 Snowflake 之上构建数据产品时快速迭代。

34. eBPF

试验

Linux 内核在多年前就内置了扩展的伯克利数据包过滤器（eBPF），它是一个可以将过滤器附加到特定套接字能力的虚拟机。但是 eBPF 的能力远远超出了包过滤的范围，它允许在内核中的不同点位触发自定义脚本，并且开销非常小。通过在操作系统内核中运行沙箱程序，开发人员可以通过 eBPF 程序给操作系统运行时添加额外的功能。在一些项目需要进行系统调用层的故障排除和剖析时，我们的团队发现像 bcc 和 bpftrace 这样的工具会简化工作。eBPF 也可用于可观测性和网络基础设施，例如 Cilium 项目可以在 Kubernetes 中以无 sidecar 开销

的方式实现流量负载平衡和可观察性，Hubble 项目在其基础上加强了安全和流量可观察性。Falco 项目使用 eBPF 进行安全监控，Katran 项目使用 eBPF 建立更有效的 L4 负载平衡。eBPF 社区正在迅速发展，并且我们看到了与可观察性领域越来越多的协同作用。

35. Feast

试验

Feast 是一个用于机器学习的开源 Feature Store。它具有几个有用的特性，包括生成时间点（point-in-time）正确的特征集以避免在训练时将容易出错的未来特征值泄露给模型，以及支持流数据源与批数据源。然而，它目前仅支持具有时间戳的结构化数据，因此，如果你在模型中处理非结构化数据，那么 Feast 可能并不适用。我们已成功地大规模采用 Feast 作为模型训练时的离线存储与预测时的在线存储。

36. Monte Carlo

试验

Monte Carlo 是一个数据可观测性平台。它使用机器学习模型，可以推断和学习数据，识别问题并在出现问题时通知用户。它使我们的团队能够跨 ETL 流水线、数据湖、数据仓库和商业智能（BI）报告维护数据质量。凭借监控仪表板即代码、中央数据目录和字段级沿袭等功能，我们的团队发现 Monte Carlo 是整体数据治理的宝贵工具。

37. Retool

试验

在之前的技术雷达中，我们曾建议评估限界低代码平台，将其作为一种将低代码解决方案应用于极少数领域的特定用例的方法。我们已经看到这个方向越来越受欢迎，特别是低代码平台 Retool，我们的团队用它来为内部用户构建解决方案，主要用于数据的查询和可视化。它使得内部用户可以更快地制作非关键业务的只读解决方案。据报告显示，Retool 的主要优势在于它的用户界面组件，以及与公共数据源快速便捷集成的能力。

38. Seldon Core

试验

Seldon Core 是一个在 Kubernetes 集群上打包、部署、监控和管理机器学习模型的开源平台。它针对一些机器学习框架提供开箱即用的支持，你可以很容易的通过各种预打包的推理服务器、定制的推理服务器、以及语言包装器将你的模型容器化。同时，借助 Jaeger 的分布式追踪能力，并通过 Alibi 实现模型可解释性，Seldon Core 解决了阻碍机器学习部署阶段某些最后一公里的难题，因此我们的数据团队非常喜欢它。

39. Teleport

试验

Teleport 是访问零信任网络基础设施的工具。传统的设置需要复杂的策略或跳板机来限制对关键资源的访问，然而，Teleport 通过统一的访问平面和取代了跳板机、VPN 或共享凭证的细粒度授权控制来简化了这些设置。

Teleport 被实现为单个二进制文件，并且开箱即用地支持多种协议（包括 SSH、RDP、Kubernetes API、MySQL、MongoDB 和 PostgreSQL 连接协议），使用户可以轻松设置和管理跨 Linux、Windows 或 Kubernetes 环境的安全访问。自从我们第一次在技术雷达中提到它以来，已经有一些团队使用了 Teleport，整体的积极体验促使我们强调这一平台。

40. VictoriaMetrics

试验

现代可观察性依赖于收集和汇总一组详尽的细粒度指标，以充分理解、预测和分析系统行为。但面对由大量冗余，协作进程和主机组成的云原生系统时，由于基数（唯一时间序列数）会随着每个额外的服务、容器、节点、集群等呈指数增长，现代可观察性的应用显得相对笨重。对于这些高基数的数据，我们发现 VictoriaMetrics 表现出众。VictoriaMetrics 尤其适用于运行由 Kubernetes 托管的微服务架构，它的 operator 使团队可以轻松地以自助服务的方式实现自我监控。我们还喜欢它的组件化架构以及即使在中央服务器不可用时也能继续收集指标的能力。虽然我们的团队对 VictoriaMetrics 很满意，但云原生的可观察性是一个快速发展的领域，我们建议也同时关注其他高性能的、Prometheus 兼容的时间序列数据库，例如 Cortex 或 Thanos。

41. Bun

评估

Bun 是一个新的 JavaScript 运行时，与 Node.js 或 Deno 相似。然而不同于 Node.js 或 Deno 的是，Bun 使用 WebKit 的 JavaScriptCore 构建，而不是 Chrome 的 V8 引擎。Bun 被设计为 Node.js 的直接替代品，它是一个单独的二进制文件（使用 Zig 编写），能够充当 JavaScript 和 TypeScript 的打包器、转译器（transpiler）与包管理器。Bun 目前正在 beta 测试，所以我们可以预计有一些漏洞或是与一些 Node.js 库的兼容性问题。然而，Bun 是从头开始构建的并且带有一些优化，包括快速启动和改进的服务端渲染，我们认为它值得评估。

42. Databricks Unity Catalog

评估

Databricks Unity Catalog 是一种可以用于 lakehouse 中文件、表或者机器学习模型等资产的数据治理方案。尽管你可以在企业数据治理领域中找到很多平台，但如果你已经在使用其他 Databricks 解决方案，那你更应该了解一下 Unity Catalog。我们想强调的是，虽然这些治理平台通常会采用一个集中式的解决方案，以更好地维持不同工作空间和工作负载的一致性，但治理责任应该通过使各个团队分别治理自己的资产而统一起来。

43. Dragonfly

评估

Dragonfly 是一个可兼容 Redis 和 Memcached API 的新型内存数据库。它利用 Linux 新有的 io_uring API 实现 I/O，并在多线程、无共享架构的基础上实现新型的算法和数据结构。因为这些明智的实现方案选择，Dragonfly 在性能方面取得了令人印象深刻的结果。尽管 Redis 仍然是我们对内存数据库解决方案的默认选项，但我们认为 Dragonfly 是一个值得评估的选项。

44. Edge Impulse

评估

在往期的技术雷达中，我们写过 TinyML。TinyML 是在带有板载传感器的小型设备上运行经过训练的模型，在不经过云端的情况下做出决策或提取特征的实践。Edge Impulse 是一个端到端托管平台，用于开发运行在如微控制器等小型边缘设备上的优化模型。该平台指导开发人员完成整个流程，包括收集并给训练数据打标签的任务。你可以轻松地上手，先在手机上进行数据收集和运行分类器，而在更强大的云托管环境中进行模型训练和优化。由此产生的识别算法还可以优化、编译并上传到广泛的微控制器架构中。虽然 Edge Impulse 是一家商业公司，但该平台对开发人员是免费的，即使对那些刚接触机器学习的人来说，整个过程也十分有趣并具有吸引力。创建工作应用程序的低门槛意味着我们将看到更多内置智能决策的边缘设备。

45. GCP Vertex AI

评估

GCP Vertex AI 是一个统一的人工智能平台，它让团队可以构建、部署、并扩缩机器学习（ML）模型。Vertex AI 中包含了可以被直接、微调后、或者结合 AutoML 使用的预训练模型，也包含了如特征存储和流水线等机器学习模型的基础设施。我们喜爱 Vertex AI 的集成能力，这有助于让它更像一个连贯的人工智能平台。

46. Gradient

评估

Gradient 是一个用于构建、部署和运行机器学习应用的平台，它非常类似于谷歌的 Colab。用户可以从模板创建 Notebook，以快速搭建 PyTorch 或 TensorFlow 或像 Stable Diffusion 这样的应用。根据我们的经验，Gradient 非常适合用于构建 GPU 密集型的模型，而且我们喜爱它的 Web 环境能够持久化保存的特点。

47. IAM Roles Anywhere

评估

IAM Roles Anywhere 是 AWS 的一项新服务，可让你在 IAM 中为运行在 AWS 之外的服务器、容器和应用程序等工作负载获取临时的安全凭证。我们发现它对工作负载分散在 AWS 和非 AWS 资源之间的混合云中特别有用。借助 IAM Roles Anywhere，现在你可以使用 X.509 数字证书创建短期凭证以此来访问 AWS 资源，而不用创建长期凭证。我们相信这种方法简化了整个混合云的访问模式，建议你可以了解看看。

48. Keptn

评估

Keptn 是用于交付和运维的控制平台，它依赖于 CloudEvents 进行插桩。就像我们在 CI/CD 流水线观测技术中提到的，Keptn 将其编排可视化为 traces。交付流水线旨在将 SRE 意图与底层实现分离，依靠其他可观测性、流水线和部署工具来响应适当的事件。我们对 Keptn 将服务级别目标（service-level objective，SLO）验证作为架构适应度函数添加到 CI/CD 流水线的想法感到特别兴奋：Keptn 允许你将服务水平指标（service-level indicators，SLI）定义为键值对，它的值表示对观测性基础设施的查询方法。之后，Keptn 将根据定义的

SLO 评估结果作为交付质量关口。例如，Keptn 对自动化操作采用相同处理，允许使用声明式定义来指定扩展 ReplicaSet 的意图，以降低平均响应时间。作为由 Dynatrace 创建的产品，Keptn 还可以与 Prometheus 和 Datadog 进行集成。

49. OpenMetadata

评估

毫无疑问，数据的可发现性已变成各个公司重要的关注点，因为它使得不同人群之间的数据共享和使用变成可能，在之前的版本中技术雷达已经包括了 DataHub，Collibra 等平台，但是我们仍在评估这个领域的其他选项，并且最近我们对 OpenMetadata 产生了兴趣。OpenMetadata 是一个致力于使用开放标准进行元数据管理的平台。我们的团队喜欢这个开源平台，因为它简单的架构、聚焦于自动化的部署、和在数据可发现性的重点关注能提升开发体验。

50. OrioleDB

评估

OrioleDB 是一个为 PostgreSQL 打造的全新存储引擎。我们团队在大量使用 PostgreSQL 的过程中发现，尽管其现有多个选项用来适应现代硬件，但因其存储引擎最初是为机械硬盘设计，达成优化目标的过程可能困难并且繁琐。OrioleDB 为了解决这些问题，实现了显式支持固态硬盘和非易失性随机存取存储器（NVRAM）的云原生存储引擎。想要试用该引擎，用户首先需要给当前的表访问方法安装增强补丁，然后以 PostgreSQL 扩展的形式安装 OrioleDB。我们相信 OrioleDB 有很大的潜力去解决某些 PostgreSQL 中的长期遗留问题，因此，我们鼓励你仔细地对该引擎进行评估。

工具

采纳

51. Great Expectations
52. k6

试验

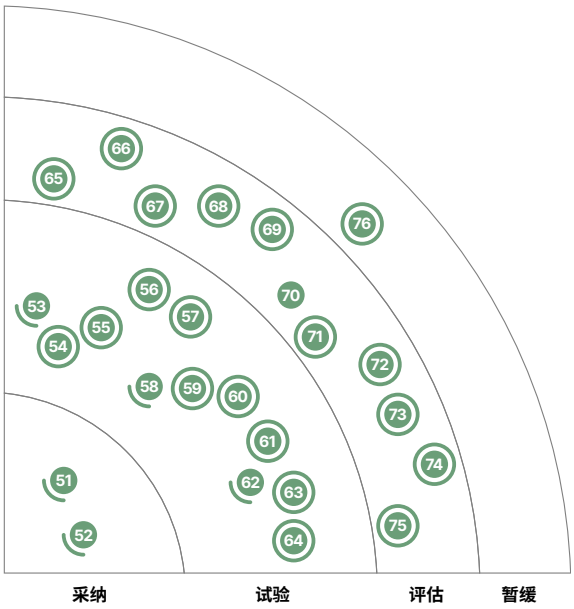
53. Apache Superset
54. AWS Backup 文件库锁定
55. AWS Control Tower
56. Clumio Protect
57. Cruft
58. Excalidraw
59. Hadolint
60. Kaniko
61. Kusto 查询语言
62. Spectral
63. Styra Declarative Authorization Service
64. 监视项目构建的 xbar

评估

65. Clasp
66. Databricks Overwatch
67. dbtvault
68. git-together
69. Harness 云服务成本管理
70. Infracost
71. Karpenter
72. Mizu
73. Soda Core
74. Teller
75. Xcode Cloud

暂缓

76. 在线格式化或代码解析服务



新的　挪进 / 挪出　没有变化

51. Great Expectations

采纳

Great Expectations 已经成为了我们的团队在数据质量领域的默认选择，这也是为什么我们建议采纳它，不仅仅是因为没有更好的替代方案，更多的是因为我们的团队在几个客户项目中都报告了它非常好的表现。Great Expectations 作为框架，允许用户搭建用于标记数据流水线中的异常或质量问题的内置控件。正如单元测试在构建流水线中运行一样，Great Expectations 在数据流水线的执行过程中进行断言。我们喜欢它的简单性和易用性，无需数据工程技能，保存在 JSON 里面的规则都可以被我们的数据领域专家修改。

52. k6

采纳

自从我们第一次在技术雷达中提及 k6 以来，它正逐渐成为性能测试的首选工具。我们非常喜欢它。因为它很容易为测试编写 JavaScript 代码，此外 k6 也有一个低代码测试生成器，使它更易于使用。文档显示 k6 可以很容易地在多种 CI/CD 流水线中添加性能测试。我们的团队发现，k6 很容易与可视化工具集成，如 Grafana 和 New Relic，这有助于调试基础设施和应用程序。当需要测试系统在重负荷下的行为时，k6 对开发者友好的特性及其生态系统，让它成为了不二之选。

53. Apache Superset

试验

Apache Superset 是一个很棒的 BI（Business Intelligence，商业智能）工具，用于与大型数据湖和数据仓库一起进行数据探索和可视化。它支持多种数据源，包括 AWS Redshift，BigQuery，Azure MS SQL，Snowflake 和 ClickHouse。此外，并非只有数据工程师才可以使用它，所有的工程师在日常工作中探索数据都能够从中受益。对于高性能需求的用例，我们发现 Superset 易于横向扩展，因为它被部署在一个 Kubernetes 集群上。自从我们上次在技术雷达中提及它后，Superset 已经从 Apache 孵化器中毕业，并且我们在多个项目中见到了它的巨大成功。

54. AWS Backup 文件库锁定

试验

保证备份在失效前无法被恶意或意外地删除、修改，在实现稳健、安全且可靠的灾难恢复时是十分必要的。在之前使用 AWS Backup 时，这些策略和保证只能通过手动实现。最近，AWS 添加了文件库锁定这一功能，以确保备份不可变且不可修改。AWS Backup 文件库锁定强制应用保留和删除策略，甚至防止包括拥有管理员权限的用户修改或删除备份文件。该功能的加入非常有价值，填补了之前在这一需求上的空白。

55. AWS Control Tower

试验

多团队的账户管理在 AWS 中是一个挑战，特别是设置和管控方面；AWS Control Tower 试图解决这一挑战。据我们团队的报告，使用它在单一且集中的位置为组织中的多个团队进行账户管理和访问控制，取得了良好的效果。

56. Clumio Protect

试验

我们成功地用 Clumio Protect 备份了 AWS 数据，特别是针对 S3。作为一个商业 SaaS 解决方案，Clumio Protect 还可以备份一系列其他 AWS 服务，并在无法通过互联网访问的地方离线存储数据。我们负责处理大规模数据保护和恢复的团队发现 Clumio Protect 很容易设置和维护；当 S3 存储桶特别大的情况下，其性能远远超过原生的 AWS 备份服务。

57. Cruft

试验

自从我们第一次定义微服务以来，我们就一直在谈论定制化服务模板。如果组织着手创建一个可以独立但一致地开发、构建、部署和操作的微服务集合，那么为团队提供一个符合标准的坚实起点是有意义的。然而，这种方法一个持久的问题是，随着时间的推移，模板会随着技术和业务需求的变化而发展，基于旧版本模板的项目就会过时了。改造模版从而改进已建立的项目成为一个重大的痛点。Cruft 试图通过提供工具来解决这个问题，以识别和修补本地项目与主模板库的当前版本之间的差异。它将 Cookiecutter 模板引擎与 git 哈希值相结合，以识别和应用模板的变化。可以把它看作是项目模板的一个包管理器。保持模板的更新是一个众所周知并长期存在的难题，因此对我们来说，Cruft 提供的解决方案听起来好得令人难以置信。然而，根据我们团队先前的反馈，Cruft 确实是有效的，并使服务构建者和维护者的工作更轻松。我们急切地想看看它的长期表现如何，但现在这个可能有用的工具也值得尝试。

58. Excalidraw

试验

我们不断地从我们的团队中听到关于 Excalidraw 的热心报告，但我们先前关于安全的警告仍然存在。Excalidraw 是一个简单却强大的在线绘图工具。有些时候团队只是需要一张快速的图片而不是正式的图表；对于远程团队而言，Excalidraw 提供了一种快速创建和分享图表的方式。我们的团队还喜欢它可以产出低保真样式的图表，这让人想起他们在同地协作时绘制的白板图表。至于安全性，在撰写本文时，任何拥有链接的人都可以查看您的图表；但请注意，付费版 Excalidraw 提供了进一步的身份验证功能并且拥有部署本地化服务器的选项。

59. Hadolint

试验

我们乐于传播有关代码静态检查工具的讯息，这些工具实际上可以帮助您发现问题而不仅仅是处理团队中缩写风格的争议。Hadolint 是一个这样的工具，它有助于发现 Dockerfile 中的常见问题。我们发现它可以快速、准确地发现问题且具有良好的使用文档。它会在第一时间解释如何解决问题以及为什么它是一个问题，从而促使 Dockerfile 作者趋向好的实践。顺便一提，Hadolint 是基于我们推荐使用于检查 shell 脚本的 ShellCheck 工具构建的。

60. Kaniko

试验

当前大多数的 CI/CD 流水线工具和平台都是以容器作为运行时构建的。而我们的许多团队正在使用 Kaniko 从这些基于容器的流水线中构建容器镜像。这是一种趋势的一部分，即不再将 Docker 容器运行时作为业界标准。有了 Kaniko，您可以在不使用 Docker 守护进程的情况下构建镜像。这有助于避免 Docker“特权”模式所带来的安全问题，而“特权”模式对任何“在 Docker 中运行 Docker”的活动都是必要的。此外，您不必首先确保流水线可以访问 Docker 守护进程。通常这都需要额外的配置，而现在都不再是理所当然。

61. Kusto 查询语言

试验

当数据相关工作变得越来越常见，我们不断见到尝试对 SQL 语言进行改进的工具；Kusto 查询语言（KQL）是其中的一种。KQL 是由 Azure 创建的语言，它给关系型查询语言带来了模块、封装、组合、复用、扩展、和动态化的能力。我们的团队格外喜欢它的交互性：你可以将一个查询语句导入渲染操作符并立刻看到生成的图表。你也可以组合这些图表到仪表盘上，同时从分钟级的运行日志上获得洞见。尽管 KQL 目前只能在 Azure Data Explorer 中使用，我们预计对 SQL 进行改进以实现更好的数据操作性的步伐不会停止。

62. Spectral

试验

Spectral 是一个强调 OpenAPI 和 AsyncAPI 的 JSON/YAML 代码静态检查工具（linter）。在设计和实现 API 或进行事件驱动的协作时，它所提供的全面且开箱即用的规则可以帮助开发者省去很多麻烦。这些规则可以用于检查 API 参数规范或规范中存在的许可声明等。其 CLI 能够让本地开发和 CI/CD 流水线中更容易地引入 Spectral，而 JavaScript API 则支持更高级的使用场景。它的 GitHub 页面链接了一些公开的真实公司（比如 Adidas）正在使用的规则集，这使得团队在采用他们自己的检查规则时有了一个良好的开始。

63. Styra Declarative Authorization Service

试验

Styra Declarative Authorization Service（DAS 声明式授权服务）是一个用来规模化管理 Open Policy Agent（OPA）的治理和自动化工具。该工具由 OPA 的开发者建立，它允许我们在不同“系统”中部署策略，包括 Kubernetes 集群、基础设施代码仓库、命名空间以及其他。最重要的是，它能对 OPA agent 做出的决定进行实时分析，以及具有调试和调查假设性策略变更场景的回放能力。它还带有审计日志，可以帮助安全团队进行历史报告。

64. 监视项目构建的 xbar

试验

在远程团队中，我们非常缺乏一个专用的项目构建监视器；不幸的是，新兴的持续集成（CI）工具缺乏对旧 CCTray 格式的支持。其结果是，失败的构建并不能及时地被团队发现。为了解决这个问题，我们的许多团队已经开始使用监视项目构建的 xbar。xbar 可以执行脚本来轮询构建状态，并将其显示在电脑的菜单栏上。也可

以编写复杂脚本来跟踪其他团队指标，例如检查凭证到期，或生产版本落后于用户验收测试（UAT）版本的差距等。当然，xbar 的功能远不止这些，但它解决了远程工作中直接和紧急的问题。Rumps 等工具同样也可以解决这些问题。

65. Clasp

评估

不幸的是，电子表格（spreadsheet）在这个世界依然大行其道，而且在可见的未来也仍会如此。电子表格是人们构建符合他们特定需求的自定义小工具的终极武器。然而，当你试图使用一些需要“真正的”代码逻辑来增强这些小工具的时候，电子表格的低代码属性就会变得非常有局限性。如果你在一个类似 Thoughtworks 这样使用 Google 的 G-Suite 的公司，Clasp 使你至少可以在 Apps Script 代码里应用一些 Continuous Delivery 的实践。它可以让你在 Apps Script 项目之外编写代码，这样就有了测试、版本控制和构建流水线，甚至使用 TypeScript 的可能性。Clasp 已经存在一段时间了，你不应期待它提供一个常规的舒适编程环境，但它确实能极大地提升使用 Apps Script 的体验。

66. Databricks Overwatch

评估

Databricks Overwatch 是 Databricks 实验室的项目，它让团队能分析 Databricks 负载运行时的各种包括成本、治理、和性能在内的多种运营指标，并且支持运行假设性的试验。它本质上是一系列在 Databricks 中填充数据表格的数据流水线，这使得它可以被例如计算笔记本（Notebooks）这类工具分析。Overwatch 是一个非常强大的工具，但仍处于试验阶段，并且它需要一定的时间成本进行配置。在使用中我们发现，Overwatch 需要 Databricks 解决方案架构师帮助建立并填充一个用于成本计算的价格参考表，但是我们预计使用它会变得日益轻松。Overwatch 提供了相对于云提供商的成本分析工具进行更深入分析的可能性。例如，我们能分析任务失败的成本（认识到快速失败比在接近最后一步时才失败的任务更省钱），并且将成本划分为各种组别（工作区，集群，任务，计算笔记本，团队）。我们也喜欢它提高的操作可见性，这使得我们可以轻松审计集群设置的访问控制并分析运营指标，例如找到长时间工作的计算笔记本或者是最大的读写卷。Overwatch 可以分析历史数据，但它的实时数据也允许设置警报，这能帮助你对 Databricks 负载添加合适的控制。

67. dbtvault

评估

Data Vault 2.0 是一种数据建模方法和设计模式，相对于其他流行的建模方法，它的目的是进一步提高数据仓库的灵活性。Data Vault 2.0 可以应用于任何数据存储中，例如 Snowflake 或 Databricks。当实现 Data Vault 仓库时，我们发现 dbt 的 dbtvault 包是一个有用的工具。dbtvault 提供了一组 jinja 模版，用于生成和执行填充 Data Vault 仓库所需的 ETL 脚本。尽管 dbtvault 存在一些小缺陷，如它缺乏对强制隐含惟一性和增量加载的支持。但总的来说，它填充了市场空白并且只需最小配置即可开始使用。

68. git-together

评估

我们对于 git-together 的出现感到十分激动，因为我们一直在消除由结对编程带来的不便。git-together 使用 Rust 编写，简化了结对编程时代码提交的属性配置。通过将 **`git-together`** 设置为 **`git`** 的别名，**`git-together`** 允许您在 **`git config`** 中添加扩展配置以捕获提交者的信息，并以每个提交者的首字母为别名。更改结对对象或者切换到单人编程或暴徒式编程（Mob Programming）时，需要您运行 **`git with`** 命令，并以 pair 对象名称的首字母收尾（例如：**`git with bb cc`**），这将便于您此后恢复到常规的 git 工作流。每次提交时，git-together 都会在 git 存储的结对对象信息中轮换作者身份，并且它会自动将任何其他作者添加到提交消息的底部。相关的配置可以和代码仓库一起 check in，从而使 git-together 可以在克隆代码仓库后自动生效。

69. Harness 云服务成本管理

评估

Harness 云服务成本管理是一款为三个主要的云服务提供商及其托管的 Kubernetes 集群提供服务的商业工具，用来帮助其可视化及管理云服务成本。该产品通过查看空闲资源以及未分配给任何工作负载的资源来计算成本效率分数，并使用历史变化趋势来帮助其优化资源分配。仪表盘突出显示成本峰值，并允许用户注册未按照预期发生的意外现象，然后为他们围绕异常检测的强化学习算法提供素材。云成本管理可以提供调整内存和 CPU 使用限制的建议，并提供优化成本或性能的选项。“Perspectives”允许您根据按照组织结构定义的过滤器（可能对应于业务部门、团队或产品）对成本进行分组，并自动分发报告以可视化云服务支出。我们相信 Harness 云服务成本管理提供了一个有说服力的功能集，来帮助组织使他们的 FinOps 实践变得更加成熟。

70. Infracost

评估

我们不断地看到有企业在没有恰当理解如何持续监控云上运营成本的时候向云服务迁移。我们之前提过将运行成本实现为架构适应度函数，而 Infracost 是一个旨在在 Terraform pull request 中可视化成本权衡的工具。它是一个开源软件，在 macOS、Linux、Windows 和 Docker 均可访问，开箱即用支持 AWS、GCP 和微软 Azure 的定价。它还提供了一个公共 API，可以查询到当前的成本数据。我们仍然对它的潜力感到兴奋，特别是它还将支持在 IDE 中提供更好的成本可见性。

71. Karpenter

评估

Kubernetes 的基本功能之一是它能够在需要额外生产力时自动启动新的 Pod，并在负载减少时关闭它们。这种水平自动缩放是一个有用的功能，但它只有在需要托管 Pod 的节点已经存在时才能工作。虽然 Cluster Autoscaler 可以做一些由 pod 故障触发的基本集群扩展，但它的灵活性有限；不过我们发现了另一个开源的 Kubernetes Operator 自动扩缩器 Karpenter，它内建了更多智能：可以分析当前的工作负载和 Pod 调度约束，以自动选择合适的实例类型，然后根据需要启动或停止它。Karpenter 是一个具有如 Crossplane 等工具精神的 operator，可以在集群外提供云资源。Karpenter 是云供应商通过其托管的 Kubernetes 集群原生提供的自动缩放服务的一个极具吸引力的工具。例如，AWS 现在在其 EKS Cluster Autoscaler 服务中支持将 Karpenter 作为首选的替代方案。

72. Mizu

评估

Mizu 是一个 Kubernetes 的 API 流量查看器。与其他工具不同的是，Mizu 不需要增加监测工具或改动代码，而是在 Kubernetes 集群的节点层面上注入一个容器作为 DaemonSet，做一些类似 tcpdump 的操作。我们发现 Mizu 可以实时监测多种协议（REST、gRPC、Kafka、AMQP 和 Redis）下的所有 API 通信，因此它可以作为一个调试工具来使用。

73. Soda Core

评估

Soda Core 是一个开源数据质量与可观测性工具。我们在之前的技术雷达上讨论过 Great Expectations，而 Soda Core 作为另一种解决方案，其关键区别在于数据校验可以通过一种称为 SodaCL（之前称为 Soda SQL）的 DSL 来表示，而非 Python 函数。数据校验一旦被写好就可以作为数据流水线的一部分被执行，也可以通过编程方式调度运行。随着我们越来越多地受数据驱动，数据质量的维护变得至关重要，因此我们建议你评估 Soda Core。

74. Teller

评估

Teller 是一款面向开发人员的开源通用密钥管理器，可确保应用程序启动时，环境变量可以被正确地配置。然而，它本身并非一个存放私密信息的保险库（vault），而是一个可连接多种源的 CLI 工具，从云密钥提供者到诸如 HashiCorp Vault 的第三方解决方案再到本地环境文件。Teller 还具有如下的额外功能：扫描代码中保存在 vault 中的密钥、编辑日志中的密钥、检测各密钥提供者之间的变化并进行同步。鉴于访问密钥访问行为的敏感性，无论如何强调对开源依赖供应链安全保障的必要性都不为过，但我们依然欣赏 Teller 作为一款 CLI 工具在本地开发环境、CI/CD 流水线和部署自动化中的使用便捷性。

75. Xcode Cloud

评估

Xcode Cloud 是一个集成在 Xcode 中的 CI/CD 工具，用于开发、测试和部署苹果应用程序。它为熟悉 Xcode、App Store Connect 和 Test Flight 的苹果开发人员提供了一体化的体验。根据我们团队的经验，它的优势在于简化流水线配置和创建配置文件和证书。这个工具面世不久，我们大多数的移动端开发团队仍然在使用更为成熟的 Bitrise。尽管如此，它仍然值得我们去评估和追踪它的发展。

76. 在线格式化或代码解析服务

暂缓

我们之前提到了测试环境中的生产数据，现在想强调另一个需要小心处理甚至完全停止的常见实践：在线格式化代码解析服务。有许多有用的网站提供格式化或解析格式（如 JSON 和 YAML）的服务，还有许多评估代码教程或在线生成代码衡量指标的网站。使用这些产品时需要非常小心。将 JavaScript、JSON 或类似代码片段粘贴到一个未知的网站很容易造成安全和隐私问题，并可能在不知不觉中将个人数据导出到不同的信息管辖区。这些站点不应该与生产数据一起使用，并且在任何其他情况下都应该谨慎对待。

语言和框架

采纳

77. io-ts
78. Kotest
79. NestJS
80. React Query
81. Swift Package Manager
82. Yjs

试验

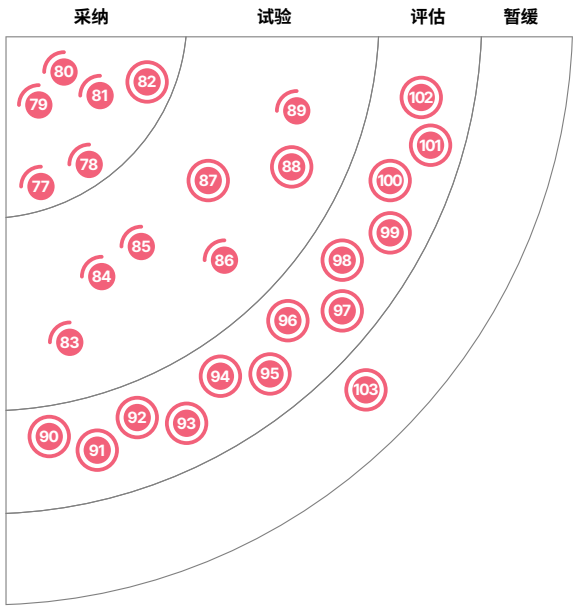
83. Azure Bicep
84. Camunda
85. Gradle Kotlin DSL
86. Jetpack Media3
87. Ladle
88. Moshi
89. Svelte

评估

90. Aleph.js
91. Astro
92. BentoML
93. Carbon Aware SDK
94. Cloudscape
95. Connect
96. 跨设备 SDK
97. Cypress 组件测试
98. JobRunr
99. Million
100. Soketi
101. Stable Diffusion
102. 合成数据保险库

暂缓

103. Carbon



新的　挪进 / 挪出　没有变化

77. io-ts

采纳

我们使用 TypeScript 开发的团队发现 io-ts 非常有价值，特别是在与最终导致创建具有特定类型的对象的 API 交互时。在使用 TypeScript 时，将数据输入到类型系统的范围内（比如来自上述 API）可能会导致运行时错误，而这些错误可能很难发现和调试。io-ts 通过提供编码和解码函数，在编译时类型检查和运行时消耗外部数据之间架起桥梁。鉴于我们团队的经验和其方法的优雅性，我们认为 io-ts 值得采纳。

78. Kotest

采纳

Kotest（原名 KotlinTest）是 Kotlin 生态中的一个独立测试工具，它在我们的团队各式各样的 Kotlin 实现（原生、JVM 或 JavaScript）中越来越受到关注。Kotest 的主要优点是它提供了丰富的测试风格来搭建测试套件，其中还有一套全面的匹配器，可以帮助你使用优雅的内部领域专用语言（DSL）编写表达式测试用例。Kotest 除了支持基于属性的测试之外，我们团队也看好它可靠的 IntelliJ 插件和支持社区。我们的许多开发者将它列为首选并推荐那些仍在 Kotlin 中使用 JUnit 的开发者考虑切换到 Kotest。

79. NestJS

采纳

过去，我们警告过 Node 泛滥的问题，即使现在，在选择 Node 时，我们仍然持谨慎态度。然而，据我们团队报告，在需要使用 Node.js 构建后端应用程序的某些场景下，NestJS 是一个很好的选择。它使开发人员能够创建可测试、可扩展、松耦合且易于维护的企业级应用程序。NestJS 是一个 TypeScript 优先的框架，它使 Node.js 应用程序的开发更安全，更不容易出错。NestJS 采用了面向对象的编程思想，遵循 SOLID 原则，同时受到 Angular 启发，采用了开箱即用的架构。

80. React Query

采纳

React Query 通常被描述为 React 缺失的数据获取库。获取，缓存，同步和更新服务器状态是许多 React 应用程序常见的需求，尽管这些需求易于理解，但众所周知，正确地实现这些需求非常困难。React Query 提供了一种基于 hooks 的更直接的方式。它与现有的基于 promise 机制的异步数据获取库协同工作，如 axios、Fetch 和 GraphQL。作为应用程序开发人员，你只需要传递一个解析数据的函数，其余的事情可以留给框架完成。该工具开箱即用，但也可以按需进行配置。它的开发者工具也能帮助刚接触此框架的开发人员理解其工作原理，遗憾的是，其开发者工具尚不支持 React Native。对于 React Native，你可以使用第三方开发者工具插件 Flipper。基于我们的经验，React Query 的第三版为我们的客户提供了生产环境所需的稳定性。

81. Swift Package Manager

采纳

当 Swift 在 2014 年推出时，并没有发布相应的包管理器。后来，Swift Package Manager 作为苹果官方开源项目创建，该项目在之后的时间中不断发展和成熟。现如今，我们的团队越来越依赖 SwiftPM，因为大多数的

依赖包都可以通过它进行管理，并且通过 SwiftPM，依赖包的创建者和使用者的操作流程都得到了极大的简化。在之前的技术雷达中，我们建议大家可以尝试使用该项目进行包管理，但如今，我们认为应该在启动新项目时将其作为首选。对于那些使用 CocoaPods 或 Carthage 等工具的现有项目，进行一个快速试验，来衡量迁移的难易程度，并检查所有依赖项是否都可用是值得的。

82. Yjs

采纳

无冲突复制数据类型（CRDT）算法被证明能够在对等节点中自动地分发与合并修改而不产生冲突。但是在实践中，即使对于足够小的数据，这些算法通常也需要大量内存来追踪由不同对等节点做出的所有修改，从而变得不切实际。Yjs 是一个精心优化的 CRDT 实现，能够在面对大型数据集和数百万个修改时将内存开销保持在合理的水平。它也能绑定到流行的文本编辑器上，这一点显著地降低了构建协作工具的成本。

83. Azure Bicep

试验

Azure Bicep 是一种使用声明式语法的领域特定语言（DSL），主要面向那些喜欢使用比 JSON 更自然的语言来编写基础设施代码的人。它支持可重用参数化模板来实现模块化资源定义。它有 Visual Studio Code extension 插件为其提供实时类型安全、智能感知和语法检查的功能，并且它的编译器允许双向转换 Azure Resource Manager（ARM）模板。Bicep 面向资源的 DSL 以及与 Azure 生态系统的原生集成使其成为 Azure 基础设施开发人员的不二之选。

84. Camunda

试验

自从我们上次提到 Camunda 以来，我们已经看到了我们的许多团队和客户在使用该平台，使其在适合引入工作流引擎的领域里，成为我们的首选工作流引擎之一。Camunda 提供的工作流和决策引擎可以作为库集成到用户的 Java 代码中。这使得测试、版本化和重构工作流变得更容易，缓解了其他低代码工作流引擎的一些缺点。我们甚至已经看到 Camunda 在具有高性能要求的环境中被使用。一些团队还很喜欢它可以很容易与 Spring Boot 做集成及它漂亮的用户界面。

85. Gradle Kotlin DSL

试验

之前，我们介绍过 Android Gradle 插件 Kotlin DSL，或 Gradle Kotlin DSL，它为使用 Gradle 构建脚本的 Android 工程增加了对 Kotlin 脚本的支持，以替代 Groovy。用 Kotlin 替换 Groovy 的目的是在 IDE 中为重构与更简便地编辑提供更好的支持，以及最终产出更易于阅读和维护的代码。对已经正在使用 Kotlin 的团队而言，这也意味着使用一门熟悉的语言处理构建。一般来说，我们现在建议在 Gradle 工程中试用 Kotlin DSL 作为 Groovy 的替代语言，尤其是当您有庞大或复杂的 Gradle 构建脚本时。许多 IDE 现在都支持迁移现有工程。仍然存在一些警告，我们建议检查文档以获取包括前置条件在内的最新细节。我们有一个团队把至少有七年历史的、450 行的构建脚本在几天之内成功地迁移了。

86. Jetpack Media3

试验

现如今安卓拥有多个媒体 API: Jetpack Media（也被称为 MediaCompat），Jetpack Media2 和 ExoPlayer。然而，这些库都是分别开发的，它们的目的不同但是功能重叠。这就导致安卓开发者在编码的时候不仅需要斟酌类库的选型，当使用的特性来自于多个库的时候，还需要编写适配器或者兼容代码。Jetpack Media3 是从现有 API 中选取通用的功能: 包括 UI、播放和媒体会话处理，然后将它们合并和改进成一个新的 API。ExoPlayer 的播放器界面也进行了更新、增强和简化，被用作 Media3 的通用播放器界面。在早期访问阶段之后，Media3 目前仍处于早期开发版本。虽然它的第一个正式版本即将发布，但我们已经在应用程序中使用 Media3 得到了积极的体验。

87. Ladle

试验

随着 Storybook 变得受欢迎，它变得越来越像一个庞然大物。如果你真正关心的是隔离和测试你的 React UI 组件，那么 Ladle 是一个替代品。Ladle 支持大多数的 Storybook API（MDX 文件暂时不支持），且可以作为 Storybook 的替代品。它是轻量级的，与 Vite 有更好的集成。并且它还提供了简单整洁的能轻易地与其他测试框架集成的 APIs。

88. Moshi

试验

听说很多使用 Kotlin 语言的团队在寻找可以替代 GSON 这样基于 Java 的 JSON 处理工具。刚出现不久的 Moshi 已经成为了许多此类团队的首选方案。从 GSON 迁移到 Moshi 非常简单，且后者原生支持 Kotlin 语言中的非空类型和默认参数特性，使得处理 JSON 的工作变得更加高效便捷。如果你正在 Kotlin 中使用基于 Java 的 JSON 处理工具，我们建议你尝试一下 Moshi。

89. Svelte

试验

在 Web 组件框架中，Svelte 通过将反应性从浏览器中转移到编译器中而脱颖而出。Svelte 不是通过使用虚拟 DOM 和浏览器优化技巧来优化 DOM 更新，而是将你的代码编译成无框架的 JavaScript 代码，像做外科手术一样更新 DOM。除了运行时的性能优势之外，这也让 Svelte 在不牺牲开发者功能的情况下优化浏览器必须下载的代码量；此外，事实证明，由于在浏览器中执行的代码较少，它对移动网络应用的性能和电池需求更加友好。除了性能优势之外，我们团队还欣赏它友好的学习曲线和来自于编写更少代码的维护优势。Svelte 本身只是组件框架，但 SvelteKit 增加了可以构建完整 Web 应用程序的功能。

90. Aleph.js

评估

目前并不缺少通过 JavaScript/TypeScript 去创建网络应用的架构。我们在技术雷达中已经介绍了很多种，但真正让 Aleph.js 从中脱颖而出的是它最初就是建立在 Deno 上的。考虑到 Deno 是由 Node 原本的开发者所创建的新的服务器端运行时，这意味着 Aleph.js 处在一个更先进的平台上，能够解决 Node 相关的很多缺陷和问

题。虽然 Aleph.js 很新，至截稿时仍然在准备 1.0 版本的发布，但它已经能够为我们提供稳定的开发体验，包括组件的热插拔等。鉴于 Deno 早已完成 1.0 的发布，对于那些能够承担风险的项目来说，Aleph.js 无疑是一个现代化的选择。

91. Astro

评估

令人难以置信的是，即使到了 2022 年，开发者社区仍在持续推出有趣的，用于构建 web 应用程序的新框架，Astro 就是最新推出的开源，多页面响应的应用程序框架，它可以在服务器上渲染页面并尽可能减少通过网络发送的 JavaScript 数量。Astro 看起来非常适合那些从多种渠道获取资源的内容型网站。我们喜欢 Astro 的一点是，尽管 Astro 鼓励只发送 HTML，但它仍然支持在适当的时候选择用您选用的前端 JavaScript 框架编写的活动组件。它通过 island architecture 做到这一点。岛屿是单个页面中的交互区域，仅在需要时才下载必要的 JavaScript。Astro 是相对较新的技术并且看起来支持日益增加的开发者及代码生态系统。它的发展值得关注。

92. BentoML

评估

BentoML 是一个 Python 优先的框架，用于在生产环境中规模化部署机器学习模型。它提供的模型与环境无关；所有模型属性、源代码和依赖包都封装在称为 Bento 的自包含格式中。这就像“模型即服务”一样。您可以将 BentoML 视为机器学习模型的 Docker：它使用预编程 API 生成可立即部署的虚拟机镜像并包含相应的功能使测试这些映像变得十分简单。BentoML 可以通过简化项目初始阶段的任务加速项目的开发，这是我们将其归纳在评估环中的原因。

93. Carbon Aware SDK

评估

当我们着眼于减少一款应用程序的碳足迹（运行软件间接导致的二氧化碳排放）时，注意力通常被导向让软件更加高效上。思路很明确：更高效的软件只需要更少的电力和服务器，从而减少发电与制造服务器所带来的碳排放。另一个策略是使应用程序具有碳意识。这是因为同样的工作负载并不总是具有相同的碳足迹。例如：在较冷气候的数据中心运行时，用于空调的电力需求会减少；或者，在能够使用更多的可再生能源（更多的阳光，更强的风力）时，碳基来源的电力需求会减少。借助 Carbon Aware SDK，软件工程师们可以查询数据源来发现对于给定的工作负载而言碳密集度更低的选项，然后将它移动到不同的位置或是在不同的时间运行它。这对那些对于时间和延迟都不敏感的大型工作负载来说是有意义的，例如训练机器学习模型。虽然这个 SDK 和可获取的数据源还不是很全面，但是我们相信是时候开始探索如何能让我们的系统具有碳意识了。

94. Cloudscape

评估

Cloudscape 是一款开源的设计系统，它不仅有一套丰富的组件集，还有 35 种交互和内容展示模式。与此同时，它使用设计令牌来进行主题化，并提供元素包装器给所有组件，这极大地简化了单元测试。这几点让它从众多设计系统中脱颖而出。

95. Connect

评估

Connect 是一系列用于构建与浏览器和 gRPC 兼容的 HTTP API 的库。与 gRPC 类似，您编写协议缓冲区的架构并实现其应用程序逻辑，然后 Connect 生成代码来处理编组、路由、压缩和内容类型协商。但是，Connect 尝试以多种方式去改进 gRPC。包括在没有翻译代理的情况下对 gRPC-Web 的原生支持；与第三方路由器或中间件的互操作性，因为 connect-go 构建在 net/http 之上（与 grpc-go 不同）；以及完全生成的类型安全的，具有手工代码的人体工程学的客户端。我们大多更喜欢 REST，而不喜欢使用 RPC 方法去构建 API。也就是说，Connect 似乎解决了我们对 RPC 的一些担忧，我们鼓励您对其进行尝试。

96. 跨设备 SDK

评估

随着智能设备持续融入我们的生活，我们开始看到跨越多个设备的新用例出现。典型的例子是我们在手机上开始阅读一则文本但是更喜欢在平板电脑上读完它。其它例子包括在笔记本电脑上绘制骑行路线，然后把数据传输到自行车电脑上以便于导航，或是使用移动手机作为网络摄像头。这些使用场景需要非常特定类型的功能，例如发现附近设备、安全通信以及多设备会话。Apple 不久前已经开始将此类功能引入到它自己的 SDK 中了，现在 Google 也发布了其跨设备 SDK 的首个预览版本。尽管该预览版本有一些限制例如，仅支持手机与平板，并且一次仅支持两个设备，但是这项技术还是令人兴奋，在其推出后我们可以随着时间的推移而采用它。

97. Cypress 组件测试

评估

Cypress 组件测试提供了一个组件测试工作台，以快速构建和测试 UI 组件。你可以用编写端到端（E2E）UI 测试的相同 API 来编写组件视觉回归测试。尽管仍处于测试阶段，但组件测试将成为 Cypress 第十版中最重要的功能。

98. JobRunr

评估

JobRunr 是一个可以替代 Quartz 调度器的 Java 后台任务执行库。它内置的用于监控和调度后台任务的仪表盘操作简便，深受我们团队喜爱。JobRunr 是开源且免费商用的，但任务迁移和恢复等功能需要付费使用。

99. Million

评估

Million 是一个新的虚拟 DOM JS 库。与 Svelte 类似，它使用编译器 Vite，来创建具有卓越渲染性能的小型 JS 库。Million 库作为单个 NPM 包提供，其中包含了多个模块，包括 router，jsx-runtime 和一个用于保证 React 兼容性的模块以创建单页应用程序。尽管 React 在十年前就普及了虚拟 DOM，但在这个领域看到新的创新还是很吸引人的。

100. Soketi

评估

Soketi 是一个开源的 WebSockets 服务器。如果你的应用程序兼容 Pusher 协议，你可以直接接入 Soketi，因为它完全遵循了 Pusher 协议版本 7。我们发现由 Cloudflare Workers 提供的 beta 支持非常有趣，因为它打开了在网络边缘使用 WebSockets 的大门。

101. Stable Diffusion

评估

OpenAI 的 DALL・E 可以从文本提示创建图像，这吸引了所有人的注意。现在，Stable Diffusion 也可以提供相同的功能。更重要的是，它是开源的。任何可以使用性能强劲的显卡的人都可以对模型进行试验，而任何拥有够足够计算资源的人都可以自己重新创建模型。结果很震撼，但也带来了重大的问题。例如，该模型使用互联网爬虫获得的图像—文本对进行训练，因此它带有社会偏见，这意味着它可能会产生非法、令人不安，或至少不受欢迎的内容。虽然 Stable Diffusion 现在有基于 AI 的安全分类器，但由于它是开源的，使用者可以禁用这一分类器。还有，艺术家们注意到，在特定的提示词下，模型可以模仿他们的艺术风格。因此引发了对于能够模仿艺术家的人工智能的伦理和法律影响的疑问。

102. 合成数据保险库

评估

合成数据保险库是一个数据生成工具库，它可以学习数据集的分布，以生成与源数据具有相同格式和统计属性的合成数据。在往期的技术雷达中，我们讨论过下游应用测试环境中的生产数据。然而，生产环境中数据分布的细微差别很难手工复制，这会导致合成数据中的一些缺陷和无法预知的情况。我们相信合成数据保险库和类似的工具可以通过合成类生产环境的单表，复合多表和多变量时间序列来填补这个差距。虽然合成数据保险库并非新鲜事物，我们仍然非常看好这项技术并推荐它。

103. Carbon

暂缓

我们看到了一些对 Carbon 编程语言产生的兴趣。这一点也不令人惊讶：它有 Google 的背书，而且它被展现为 C++ 的天生继承者。在我们看来，C++ 不会以足够快的速度被取代，正如在过去几十年的时间里软件工程师们所表现的那样，写出安全且没有错误的 C++ 代码是一件极其困难且耗时的事情。虽然 Carbon 是一个有意思的概念，它专注于从 C++ 移植，但是在没有一个可工作的编译器的情况下，很明显它离可以使用还有很长的路要走，而且如果你想从 C++ 移植，也有其他现代的编程语言可以作为不错的选择。现在谈 Carbon 是否会成为 C++ 的天生继承者还太早了，不过，以今天的视角来看，我们推荐项目组去关注一下 Rust 和 Go 而不是等着 Carbon 的到来而推迟移植项目。

想要了解技术雷达最新的新闻和洞见?

请选择你喜欢的渠道来关注我们

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。我们在 18 个国家 / 地区的 50 个办事处拥有 12000 名员工。我们拥有专业卓越的跨职能团队，汇集了大量战略专家、开发人员、数据工程师和设计师，25 年多来，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 首创“分布式敏捷”概念，我们深知如何集全球团队之力大规模交付卓越的软件。如今，我们致力于帮助客户开启流畅数字化之路，提升公司应变能力，引航未来征程。

/thoughtworks