技术雷达

有态度的前沿技术解析

thoughtworks

第25期

关于技术雷达

Thoughtworker 酷爱技术。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。

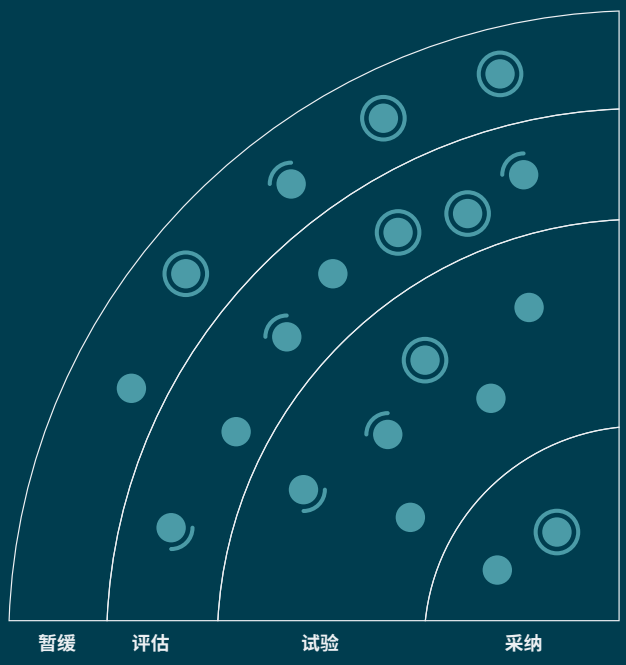
我们建议您探究这些技术以了解更多细节。技术雷达的本质是图形性质，把各种技术项目归类为技术、工具、平台和语言和框架。如果雷达技术可以被归类到多个象限，我们选择看起来最合适的一个。我们还进一步将这些技术分为四个环以反映我们目前对其的态度。

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 18 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。

技术雷达由 Thoughtworks 技术顾问委员会 (TAB) 创建。受疫情影响，本次技术雷达仍然通过线上讨论完成。

中国区技术雷达汉化组：

王妮 | 伍斌 | 姚琪琳 | 张凯峰 | 包欢 | 蔡虎 | 邓奕 | 樊卓文 | 符文 | 黄进军 | 寇永 | 李陈泽 | 李辉 | 李燕子 | 梁晶 | 梁若琳 | 林晨 | 娄麒麟 | 吕曦冉 | 马宇航 | 孟然 | 闵锐 | 彭天俊 | 索云清 | 汪耀 | 王启瑞 | 王乔阳 | 吴兵华 | 吴瑞峰 | 向博 | 向量 | 熊欣 | 杨光 | 杨君君 | 杨麟 | 杨旭东 | 张刚 | 张丽

Rebecca Parsons (CTO)
Martin Fowler (Chief Scientist)
Bharani Subramaniam
Birgitta Böckeler
Brandon Byars
Camilla Falconi Crispim
Cassie Shum
Erik Doernenburg
Fausto de la Torre
Hao Xu
Ian Cartwright
James Lewis
Lakshminarasimhan Sudarshan
Mike Mason
Neal Ford
Perla Villarreal
Scott Shaw
Shangqi Liu
Zhamak Dehghani

本期主题

适配到 Kafka, 还是改编自 Kafka

本期技术雷达的许多话题（其中一些并没有进入本期最终版本），都是有关我们团队正在使用工具，来适配到 Kafka，或改编自 Kafka。其中一些工具允许使用更传统的 kafka 接口（例如 ksqlDB、ConfluentKafka REST Proxy 和 Nakadi），而另一些则旨在提供额外的服务，如 GUI 前端界面和编排插件。我们猜测，存在大量这类工具的部分重要原因，来自 Kafka 某些组件深层的复杂性。组织越发需要及时改编 Kafka，来适应现有的架构和流程。一些团队最终将 Kafka 作为下一代企业级服务总线，而这正是“追求方便引发的滑坡谬论”主题的一个例证。但另一些团队则使用 kafka 来全局性地访问所发生的业务事件。后者认识到，有时使用集中式的基础设施，能更容易地适应业务的复杂性。所以他们会通过仔细的设计和治理，来避免复杂性的蔓延。无论怎样，这些都表明 Kafka 会成为异步发布 / 订阅大批量消息事实上的标准。

便利背后的陷阱

下面这个软件开发的反模式，在技术雷达诞生之初就已经存在了——开发团队一般为了图方便，而在其技术生态系统的复杂环节中，引入一些不合理的行为，从而欠下长期存在的技术债务，并会引发更糟的问题。这种例子不胜枚举，包括将数据库作为集成点，使用 Kafka 作为全局编排器，以及将业务逻辑代码混杂到基础设施代码中。现代软件开发为开发人员提供了许多隐藏上述不当行为的场所，这样缺乏经验或考虑不周的团队，经常因为没有仔细考虑不当耦合的长期后果，而陷入困境。不适当的团队结构和对康威定律的偏离，也让问题雪上加霜。随着软件系统变得越来越复杂，开发团队必须勤于创建与维护经过深思熟虑的架构和设计，而不是为了图方便而做出草率的决定。通常，考虑特定方法的可测试性，会使团队降低做出有问题决策的机会。如果不加合理设计，软件会逐渐趋于复杂。精心设计，加上更为重要的持续治理，可以确保进度压力或其他众多破坏力，不会导致团队为图一时之便而做出不正确的决策。

康威定律仍然有效

许多架构师会引用康威定律——来自 1960 年代的观察：团队的沟通结构会影响到软件的设计，以证明团队组织的变化是合理的。我们对本期技术雷达中提名的几个条目的发现是，组织的团队结构在处理得当时，仍然是一个关键的推动因素，而在处理得不好时，则会变成严重的障碍。我们讨论的例子包括需要围绕平台团队进行产品思考，而不是将他们视为办公室文员；涉及效能的团队拓扑以及得到越来越多认可的团队认知负载；以及围绕程序员生产力而开发的新框架 SPACE。组织在工具上花费了大量资金，但许多组织通过关注构建软件的人，以及能让他们在特定组织内变得更高效的原因，发现了更好的生产力提升。

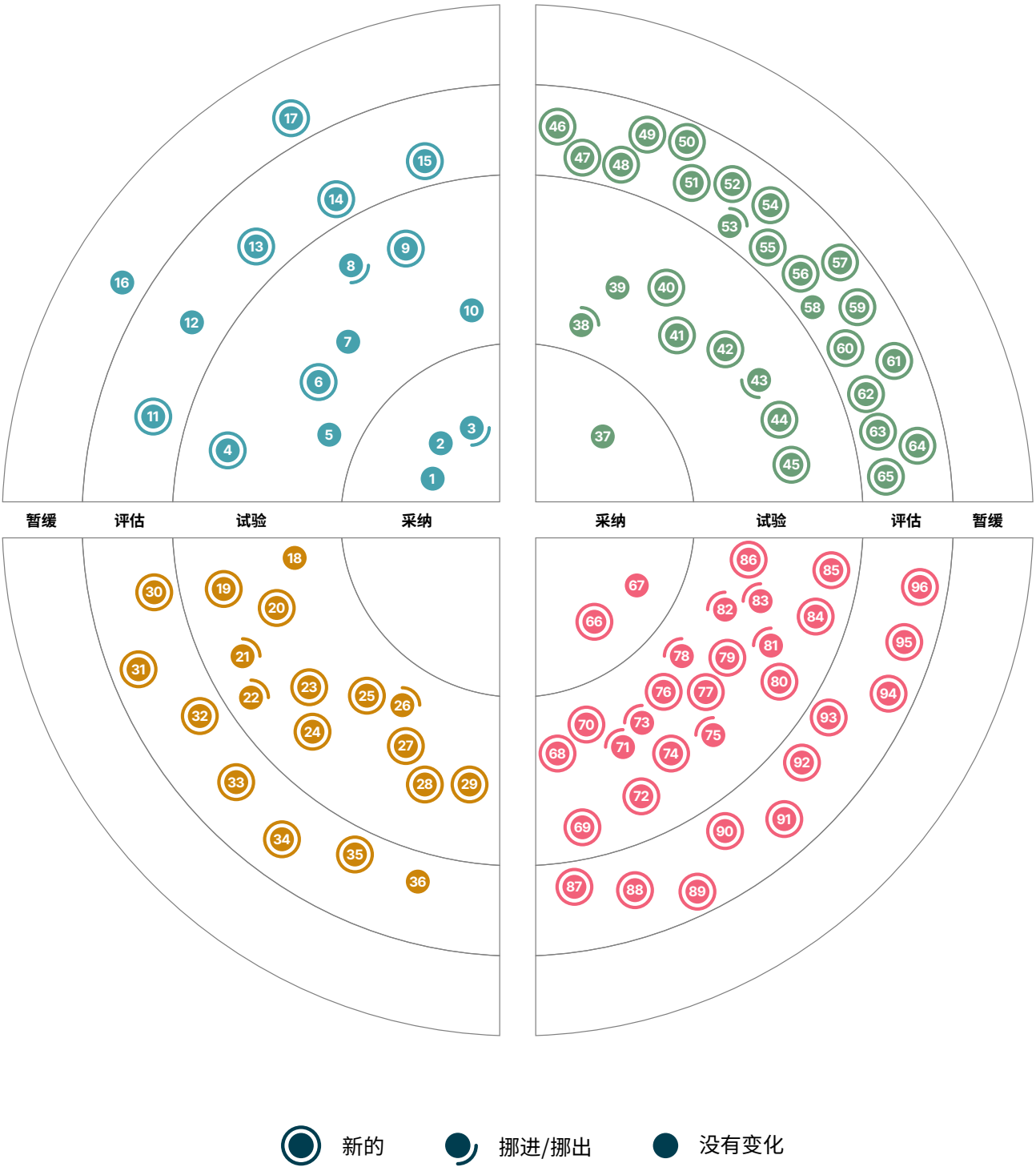
避免过度使用巧妙的技术

许多软件界人士都欣赏针对复杂问题的巧妙解法，但其实这些巧妙的解决方案往往都源于其自身引入的偶然复杂度。这种现象有很多例子。一种不幸但常见的实践是将用于编排或协调的代码藏到不合适的地方。例如，过度应用 Airflow、Prefect 等灵巧的工作流管理工具，以编排的方式管理复杂的数据流水线。也有很多 Nx 这样的工具用于解决单体式代码库导致的问题。团队通常不会意识到，这只是在两倍、三倍地增加偶然复杂度，而不会退后一步审视全局，想想当前的解决方案会不会比问题本身更糟糕。与其使用更多技术来解决问题，团队更应该分析根因，解决潜藏的本质复杂度并不断改进。例如数据网格就解决了关于组织及技术的基本假设，这些假设通常会导致过度复杂的数据流水线及工具。

雷达中平台相关条目变少

本期雷达中和平台相关的条目明显减少了，我们将其归因于一些行业标准的整合：大多数公司已经选择好了云供应商，他们几乎都基于 Kubernetes 和 Kafka 实现了容器编排和高性能消息系统的标准化。这是否意味着平台不再重要？或者说，我们是否正在经历一个盛衰周期？举个类似的例子，在数据库的发展史中，我们已经见证了一些飞速创新然后发展停滞的周期（如同 Stephen J. Gould 提出的“间断平衡”理论）。随着企业大规模地完成云迁移，行业或许已经进入了相对冷静期，等待着下一波颠覆式创新浪潮的到来。

The Radar



The Radar

技术

采纳

1. 交付核心四指标
2. 平台工程产品团队
3. 零信任架构

试验

4. CBOR/JSON 双重协议
5. 数据网格
6. 遗留系统的活文档
7. 移动微前端
8. 远程集体编程
9. 统一远程团队墙
10. 团队的认知负载

评估

11. AR 空间定位点
12. Hotwire
13. 非集群资源的 Operator 模式
14. 远程自发技术讨论
15. 软件物料清单

暂缓

16. Pull Request 等同于代码同行评审
17. 测试环境中的生产数据

平台

采纳

—

试验

18. Backstage
19. ClickHouse
20. Confluent Kafka REST Proxy
21. GitHub Actions
22. K3s
23. Mambu
24. MirrorMaker 2.0
25. 用于 Kubernetes 的 OPA Gatekeeper
26. Pulumi
27. Sealed Secrets
28. Vercel
29. Weights & Biases

评估

30. Azure 认知搜索
31. Babashka
32. ExternalDNS
33. Konga
34. Milvus 2.0
35. Thought Machine Vault
36. XTDB

暂缓

—

The Radar

工具

采纳
37. fastlane

试验
38. Airflow
39. Batect
40. Berglas
41. Contrast Security
42. Dive
43. Lens
44. Nx
45. Wav2Vec 2.0

评估
46. cert-manager
47. Cloud Carbon Footprint
48. Code With Me
49. Comby
50. Conftest
51. Cosign
52. Crossplane
53. gopass
54. Micoo
55. mob
56. 现代 Unix 命令
57. Mozilla Sops
58. Operator Framework
59. Pactflow
60. Prefect
61. Proxyman
62. Regula
63. Sourcegraph
64. Telepresence
65. Vite

暂缓
—

语言和框架

采纳
66. Jetpack Compose
67. React Hooks

试验
68. Arium
69. Chakra UI
70. DoWhy
71. Gatsby.js
72. Jetpack Hilt
73. Kotlin Multiplatform Mobile
74. lifelines
75. Mock Service Worker
76. NgRx
77. pydantic
78. Quarkus
79. React Native Reanimated 2.0
80. React Query
81. Tailwind CSS
82. TensorFlow Lite
83. Three.js
84. ViewInspector
85. Vowpal Wabbit
86. Zap

评估
87. Headless UI
88. InsightFace
89. Kats
90. ksqlDB
91. Polars
92. PyTorch Geometric
93. 乾坤
94. React Three Fiber
95. Tauri
96. Transloco

暂缓
—

技术

采纳

1. 交付核心四指标
2. 平台工程产品团队
3. 零信任架构

试验

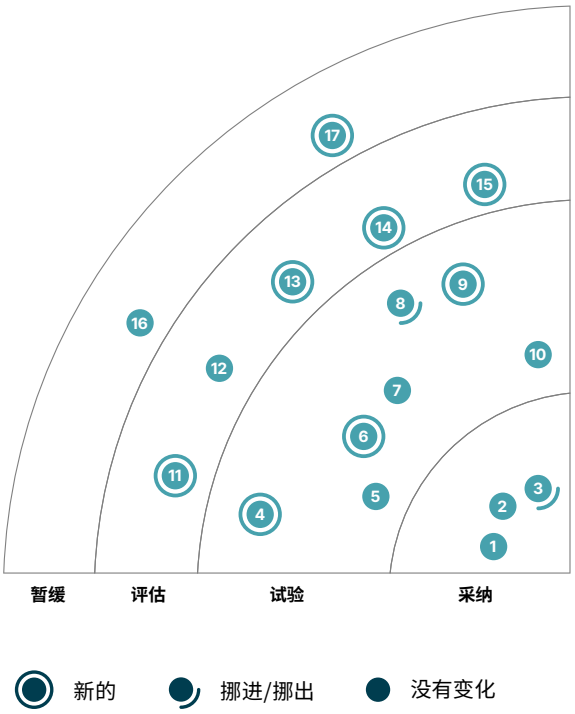
4. CBOR/JSON 双重协议
5. 数据网格
6. 遗留系统的活文档
7. 移动微前端
8. 远程集体编程
9. 统一远程团队墙
10. 团队的认知负载

评估

11. AR 空间定位点
12. Hotwire
13. 非集群资源的 Operator 模式
14. 远程自发技术讨论
15. 软件物料清单

暂缓

16. Pull Request 等同于代码同行评审
17. 测试环境中的生产数据



1. 交付核心四指标

采纳

为了度量软件交付的效能，越来越多的组织开始采用由 DORA 研究项目定义的交付核心四指标，即：更改前置时间、部署频率、平均修复时间（MTTR）和变更失败率。这项研究及其统计分析展示了高效能交付团队和这些指标的高度相关性，它们为衡量一个团队、甚至整个交付组织的表现提供了极佳的领先指标。

虽然我们仍是这些指标的支持者，但自从我们最早开始监控它们以来，也得到了一些教训。我们也持续看到被误导的度量方式，这些方式使用的工具仅基于持续交付（CD）流水线。在衡量稳定性指标（MTTR 和更改失败百分比）时，仅依赖 CD 流水线数据提供的信息并不足以确定部署失败对真实用户的影响。只有包含真实事故（如用户服务降级）的数据时，稳定性指标才有意义。

与其它所有指标一样，我们建议始终牢记度量这些指标的终极目的，并使用它们反复思考和学习。例如，在花费数周时间构建复杂仪表板工具之前，可以考虑定期在团队回顾会议当中进行 DORA 快速检查。这样的做法能够使团队有机会思考哪些能力应被提升以改进这些指标，这比过于详细的开箱即用工具更有效。

2. 平台工程产品团队

采纳

我们继续将平台工程产品团队视为默认团队配置，关键在于是他们只是一个专注于内部平台客户的产品团队。因此，在使用与任何其他（以外部为重点的）产品团队相同的工程学科和工作方式的同时，明确定义客户和产品至关重要；平台团队在这方面并不特殊。我们强烈警告不要只将现有的内部团队重命名为“平台团队”，而同时保持工作方式和组织结构不变。在考虑如何最好地组织平台团队时，我们仍然非常喜欢使用团队拓扑中的概念。我们认为平台工程产品团队是一种标准方法，也是高性能 IT 的重要推动者。

3. 零信任架构

采纳

我们总是听闻企业发现由于过度依赖“安全”的网络边界，他们的安全性受到严重损害。一旦这个外部边界被攻破，内部系统就会变得欠缺保护，攻击者能够快速而轻松地部署自动数据提取工具和勒索软件，而这些攻击往往在很长一段时间内都不会被察觉。这使我们认为推荐零信任架构（ZTA）作为默认方案是一个明智之选。

ZTA 是安全架构和策略的一个范式转变。它基于这样的假设：网络边界不再代表安全边界，同时不应仅仅根据用户或服务的物理或网络位置来予以盲目的信任。可用于实现 ZTA 各方面的资源、工具和平台的数量不断增加：包括基于最小特权与尽可能细化的原则，还有以持续监控与自动减轻威胁的方式执行安全策略即代码；通过运用服务网格去执行应用到服务与服务到服务的安全控制；通过实现二进制证文去验证二进制执行文件的来源；除了传统的加密方式之外，该架构还添加了安全隔离区去加强数据安全的三大支柱：数据在传输、存储和内存中的安全。关于该主题的介绍，请参考 NIST ZTA 的出版物和谷歌关于 BeyondProd 的白皮书。

4. CBOR/JSON 双重协议

试验

尽管 CBOR 协议并不是什么新鲜事物，然而我们发现越来越多的数据交换场景都在使用它，尤其是在多种类型的应用程序需要互相通信的时候，比如服务与服务之间、浏览器与服务之间等等。Borer 是 CBOR 编码器 / 解码器的 Scala 实现，它允许互相通信的两端通过协商自由选择二进制或者传统 JSON 作为消息格式，这一点我们认为非常有用。同样的消息，既能够以普通文本在浏览器里显示，又能够表示为精简高效的二进制，这是非常实用的特性。在未来，随着物联网、边缘计算以及其他受限环境计算的逐渐兴起，我们认为 CBOR/JSON 混合协议会更加流行。

5. 数据网格

试验

我们越来越多地看到数据驱动的组织想要实现的目标，与当前数据架构和组织结构所允许的目标是不匹配的。组织希望将数据驱动决策、机器学习和分析嵌入到其产品和服务以及内部运营的许多方面中；从本质上讲，他们希望通过数据驱动的智能来增强其运营环境的各个方面。然而，在我们可以嵌入分析数据、访问并在业务领域和运营中管理这些数据之前，还有很长的路要走。现在，管理分析数据的各个方面，都被外部化到运营业务领域之外的数据团队和数据管理单体：数据湖和数据仓库。数据网格用于消除分析数据和业务运营的二分法，是一种去中心化的社会技术方法。其目标是将分析数据的共享和使用嵌入运营业务的各个领域，并缩小运营和分析平台之间的差距。它建立在四个原则之上——域数据所有权、数据即产品、自助数据平台和计算联合治理。

我们的团队一直在实施数据网格架构；他们创建了新的架构抽象，例如数据产品量子，可以将代码、数据和策略作为分析数据共享的自治单元进行封装，嵌入到运营域中；他们还构建了自助数据平台功能，以声明方式管理数据产品量子的生命周期，如数据网格。尽管我们的技术取得了进步，但仍然在数据网格拓扑中使用现有技术时遇到了阻力，更不用说在某些组织中，将共享和使用数据作为业务领域的首要职责时所遭遇的抵抗。

6. 遗留系统的活文档

试验

活文档来自行为驱动开发（BDD）社区，通常被视作有可执行规范且维护良好的代码库的“专利”。如今我们发现这种技术也可以应用于遗留系统。团队在进行系统现代化改造时，时常受限于缺乏业务知识。由于人员流动以及现有文档已经过时，代码成了唯一可靠的依据。因此当我们接管遗留系统时，如何重新建立文档与代码间的关联，以及如何在团队中传播业务知识变得尤为重要。在实践中，我们会首先尝试对代码进行简单的清理和安全的重构，以此加深我们对业务的理解。在此过程中，我们需要向代码添加注释，以便随后自动生成活文档。这与在全新项目中使用 BDD 非常不同，但对于遗留系统来说这是个良好的开端。根据生成的文档，我们可以进一步将一些规范转换为可执行的高阶自动化测试。反复执行此操作后，最终可以获得一份与代码密切相关并且部分可执行的遗留系统的活文档。

7. 移动微前端

试验

自 2016 年雷达介绍了微前端以来，我们已经看到其在 Web UI 中得到了广泛应用。然而，最近我们发现不少项目也将这种架构风格扩展到了移动应用程序中，亦可称其为移动微前端。当应用程序变得足够庞大且复杂时，有必要将其开发分布在多个团队中。如此一来，在保证团队自治的同时，还要将他们的工作集成到单个应用中是很有挑战的。有的团队在编写自己的框架来支持这种开发风格，过去我们提到过 Atlas 和 Beehive 可以作为可行框架以简化多团队应用开发的集成问题。最近我们看到有的团队亦使用 React Native 达成了此目标。各个 React Native 微前端在各自的代码仓库中保存，以支持独立的编译、测试和部署。负责整体应用的团队则将各个团队构建的微前端统一集成到发布版本中。

8. 远程集体编程

试验

我们持续看到许多团队在工作中远程协作。对于这些团队来说，远程集体编程是一种值得尝试的技术。远程集体编程允许团队成员快速“蜂拥”在一个问题或者一小段代码周围，不用受多个人挤在一张办公桌前这种物理限制。团队可以对一个问题或者一段代码，使用其选择的视频会议工具进行快速地协作，而无需连接到一个大型显示器，预定会议室或者找一个白板。

9. 统一远程团队墙

试验

随着远程分布式工作的团队越来越多，我们察觉到人们遗漏了一样东西——物理团队墙。它能统一显示各种故事卡、任务、状态和进度，在团队中充当信息辐射器和信息枢纽。通常来说，团队墙仅仅作为一个集成点，实际的数据存储在各种不同的系统中。而随着团队越来越远程化，我们不得不回归在各个信息源系统中查看所需信息的模式，很难再通过团队墙一目了然地了解项目。统一远程团队墙是一项重新引入虚拟团队墙的简单技术。与它为团队带来的益处相比，我们认为值得花费一定开销来保持团队墙的更新。对于一些团队来说，更新物理墙是日常“仪式”的一部分，远程墙也可以做到这一点。

10. 团队的认知负载

试验

系统的架构反映了组织架构和沟通机制。我们应当有意识地关注团队如何互动，这并不是什么大新闻，正如康威逆定律（Inverse Conway Maneuver）所描述的那样。团队的交互是影响团队向客户交付价值的速度和容易程度的重要因素。我们很高兴找到一种方法来度量这些交互。我们使用高效能团队模式（Team Topologies）的作者提供的评估方法，这种评估方法可以让你理解团队构建、测试和维护其服务的难易程度。通过衡量团队的认知负载，我们能够为客户提供更好的建议，帮助他们改变团队架构，改进团队的互动方式。

11. AR 空间定位点

评估

许多增强现实 (AR) 应用需要获得用户设备的位置和方向，为此，它们一般会使用基于 GPS 的方案，但空间定位点这一新技术也值得考虑。这一技术利用设备摄像头记录视觉图像，通过图像的特征点以及它们在 3D 空间中的相对位置来识别其在真实世界中的位置，然后在 AR 空间中创建相应的定位点。虽然空间定位点无法取代所有基于 GPS 和标记的定位点，但它们确实比大多数基于 GPS 的方案精度更高，也比基于标记的定位点更能适应不同的视角。目前，我们只使用过 Google 的 Android 云定位点服务，它确实很有效。此外，Google 还一反常态地推出了 iOS 云定位点服务，而微软的 Azure 空间定位点服务甚至支持更多的平台。

12. Hotwire

评估

继成功发布服务器端 Email 应用 HEY 之后，Basecamp 对外公布，他们于今年夏天将旗舰产品 Basecamp 3 迁移到了 Hotwire。当很多组织越来越多地把单页应用 (SPAs) 作为新 Web 开发的首选，我们非常兴奋地看到 Hotwire 一直在逆流而上。和单页应用 (SPAs) 不同，Hotwire 应用程序的绝大部分业务逻辑和界面导航都在服务器端运行，只有很少量的 JavaScript 在浏览器上运行。Hotwire 把 HTML 页面模块化为组件 (称为 Turbo Frames)，这些组件支持延迟加载，有相互独立的上下文，能够基于用户行为修改上下文从而更新 HTML 页面。单页应用 (SPAs) 提供无可否认的用户响应能力，但是把传统服务器端 Web 编程的简单性与现代浏览器工具相结合，为平衡开发人员效率和用户响应能力提供了一种令人耳目一新的方式。

13. 非集群资源的 Operator 模式

评估

除了管理部署在集群上的应用程序，我们看到 Kubernetes Operator 模式越来越多地用于其他地方。非集群资源的 Operator 模式可以利用自定义资源和在 Kubernetes 控制面板中实现的事件驱动调度机制，来管理与集群外部相关的活动。该技术建立在由 Kube 管理的云服务的思想之上，并将其扩展到其他活动，例如持续部署或者及时响应外部存储库的变化。与专门构建的工具相比，这种技术的一个优势就是它开辟了一系列的工具，这些工具有的是 Kubernetes 自带的，有的则来自更广泛的生态社区。您可以使用 **`diff`**、**`dry-run`** 或 **`apply`** 等命令与 Operator 的自定义资源进行交互。Kube 的调度机制消除了以正确顺序编排活动的必要性，从而使开发更容易。如 Crossplane、Flux 和 ArgoCD 等开源工具都利用了这项技术。随着时间的推移，我们希望看到更多这样的工具出现。

14. 远程自发技术讨论

评估

我们观察到远程协作工具的持续创新。Slack 新增的 Huddles 特性提供了类似 Discord 的持久化语音通话，供用户随时加入或退出。Gather 提供了一种有创意的方式，通过头像和视频模拟虚拟办公室。很多集成开发环境 (IDE) 为结对编程和调试提供了直接的协作功能：我们之前列举了 Visual Studio Live Share，并在本期当中包含了 JetBrains Code With Me。由于包括视频会议在内的工具持续地演化协作方式，我们看到参与远程自发技

术讨论的团队数量与日俱增，重新创造了非正式对话的自发性，而非预定 Zoom 或 Microsoft Teams 会议的意向性。我们并不期待通过数字化工具完全重建面对面交流的丰富性，但我们确实看到，通过为团队提供多种协作渠道而非依赖一套工具链的方式，提升了远程团队的效率。

15. 软件物料清单

评估

2021 年 5 月，美国白宫发布了《关于改善国家网络安全的行政命令》。该文件提出了一些与我们在过去的技术雷达中展示的项目相关的技术要求，例如零信任架构以及使用安全策略即代码的自动合规性扫描。该文档的大部分内容都致力于提高软件供应链的安全性。特别引起我们注意的一项是要求政府软件应包含机器可读的软件物料清单（SBOM），它被定义为“包含构建软件使用的各种组件的详细信息和供应链关系的正式记录”。换句话说，它不仅应该详细说明交付的组件，还应该详细说明用于交付软件的工具和框架。这一秩序有可能开启软件开发透明和开放的新时代。这无疑会对以软件为生的我们产生影响。即使不是全部，今天生产的绝大部分软件产品都包含或在构建过程中使用了开源组件。通常，消费者无法得知哪个版本的软件包可能会影响其产品的安全性。于是他们不得不依赖零售供应商提供的安全警报和补丁。该行政命令确保向消费者提供所有组件的明确描述，使他们能够实施自己的安全控制方案。由于 SBOM 是机器可读的，这些控制可以自动化。从这一举措我们感受到了向拥抱开源软件的转变，在享受开源的同时我们也会实际解决它带来的安全风险。

16. Pull Request 等同于代码同行评审

暂缓

一些组织似乎觉得 pull request 等同于代码同行评审。他们认为唯一能够实现代码同行评审的方法就是 pull request。我们发现这种方法会造成严重的团队瓶颈，并且显著地降低反馈的质量，因为超负荷的评审人员会开始简单地拒绝 pull requests。虽然有观点认为，这是一种展示代码评审强制性的方式，但是我们的一个客户被告知，这种观点没有任何的依据，因为没有证据能够表明代码在接受之前被任何人实际阅读过。Pull requests 只是管理代码评审工作流的一种方式，我们敦促人们考虑其它的方法，特别是在需要仔细提供指导和反馈的情况下。

17. 测试环境中的生产数据

暂缓

我们一直认为测试环境中的生产数据是值得关注的领域。首先，它引发了许多最终导致了声誉受损的案例，例如从测试系统向整个客户群发送了不正确的警报。其次，测试系统的安全级别往往较低，尤其是围绕隐私数据的保护。当每个开发和测试人员都可以访问测试数据库中的生产数据副本时，对生产数据访问的精心控制就失去意义了。尽管您可以混淆数据，但这往往仅适用于特定字段，例如信用卡号。最后一点，当从不同国家或地区托管或访问测试系统时，将生产数据复制到测试系统可能会违反隐私法，这在复杂的云部署中尤其麻烦。为解决这些问题，使用假数据是一种更安全的策略。现存的工具也能帮我们创建假数据。在某些场景，例如重现错误或训练特定的机器学习模型时，我们承认确实有复制生产数据特定元素的必要。但我们建议此时一定要谨慎行事。

平台

采纳

—

试验

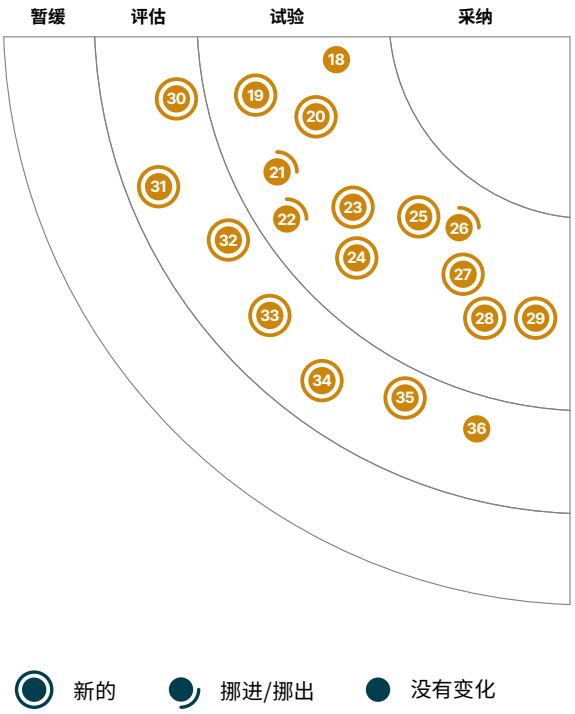
18. Backstage
19. ClickHouse
20. Confluent Kafka REST Proxy
21. GitHub Actions
22. K3s
23. Mambu
24. MirrorMaker 2.0
25. 用于 Kubernetes 的 OPA Gatekeeper
26. Pulumi
27. Sealed Secrets
28. Vercel
29. Weights & Biases

评估

30. Azure 认知搜索
31. Babashka
32. ExternalDNS
33. Konga
34. Milvus 2.0
35. Thought Machine Vault
36. XTDB

暂缓

—



18. Backstage

试验

随着各个组织持续关注改善开发人员体验和效率，我们看到 Backstage 越来越受欢迎，且组织开始采纳开发者门户。这些组织正在寻求支持和简化其开发环境。随着各种工具和技术数量的增加，某种形式的标准化，对于保持一致性变得越来越重要。这样开发人员可以专注于创新和产品开发，而不是陷入重新造轮子的泥潭。Backstage 是一个由 Spotify 创建的开源开发者门户平台。它基于软件模板、统一的基础设施工具和一致且集中的技术文档。其插件式架构，使其在组织的基础设施生态系统中，具有可扩展性和适应性。我们将持续关注新的 Backstage Service Catalog。它目前正处于 alpha 测试阶段，可以用来追踪组织生态系统中所有软件的所有权和元数据。

19. ClickHouse

试验

ClickHouse 是用于实时分析的开源柱状在线分析处理（OLAP）数据库。它起始于 2009 年的一个实验项目。从那以后，它已经发展成一个高性能且可线性扩展的分析数据库。它兼备高效查询处理引擎和数据压缩功能，使其适合在不进行预聚合的情况下，运行交互式查询。我们已经使用过 ClickHouse，并且对它的高性能印象深刻。

20. Confluent Kafka REST Proxy

试验

Kafka 是事件驱动架构的常规默认平台，但若调整其以适应遗留系统的环境，则会引入不匹配的问题。然而在一些案例中，我们成功地使用 Confluent Kafka REST Proxy，来将遗留系统环境的复杂度降至最低。该代理允许开发人员通过 HTTP 接口访问 Kafka，这在难以使用原生 Kafka 协议的环境中尤为有用。例如，只须让 SAP 团队通过预配置好的 SAP 远程函数调用，发出一条 HTTP POST 命令，就能消费到由 SAP 发出的这个事件。这避免了启用 SAP 相关的 Java 抽象（以及用来管理它的团队）。尽管此代理的功能非常全面，但与任何此类适配器工具一样，我们都建议应谨慎且清醒地了解其中所涉及到的利弊权衡。我们相信，该代理允许遗留系统的生产者发送事件这一点是很有价值的，但经由它来创建事件的消费者时要多加小心，因为抽象会变得更为复杂。该代理不会改变 Kafka 消费者是有状态的这一事实，这意味着由 REST API 创建出的消费者实例，会与特定的代理相绑定。此外，需要进行 HTTP 调用来消费主题中的消息，会改变 Kafka 事件的标准语义。

21. GitHub Actions

试验

虽然曾在技术雷达中提出慎用的建议，但我们依然看到了用户对于 GitHub Actions 的持续热情。尽管如此，之前的建议依然有效——对于复杂工作流来说，GitHub Actions 还不算成熟的 CI/CD 工具。例如，它不能重新触发工作流中单独的一个 job，不能在 composite action 中调用其他的 action，也不支持共享库。另外，虽然 GitHub Marketplace 生态圈具备显著的优势，但让第三方的 Github Actions 访问自己的构建流水线，会带来以不安全的方式共享机密的风险（我们建议遵循 GitHub 的安全加固建议）。尽管有种种顾虑，但在 GitHub 源代码旁直接创建构建工作流的便利性，对很多团队还是有着不小的吸引力。另外，还可以使用 act 在本地运行

Github Actions。和以前一样，我们建议审慎地权衡 Github Actions 的优缺点。但我们的很多团队还是非常喜欢它的简单性。

22. K3s

试验

K3s 是一个轻量级的用于物联网和边缘计算的 Kubernetes 发行版。它可以带来与 Kubernetes 相当的优势，同时又降低了运营的开销。它的增强部分包括轻量级的存储后端（K3s 默认使用 sqlite3 作为存储后端，而非 etcd）和一个单一且具有最少的操作系统依赖的二进制包，这些都使得 K3s 适用于资源受限的环境。我们已经在 POS 机中应用了 K3s 并对它十分满意。

23. Mambu

试验

Mambu 是一个 SaaS 云银行平台。它使客户能够轻松灵活地构建和更改他们的银行和借贷产品。与其他开箱即用但只能通过硬编码来集成的银行核心平台不同，Mambu 是专为不断变化的金融产品而设计的。它具备别具一格的工作流，同时使用 API 驱动的方法，来定制业务逻辑、流程和集成方式。我们目前有几个项目在使用 Mambu。凭借其基于云的可扩展性和高度可定制的特性，它正在成为构建金融产品时明智的默认业务领域系统。

24. MirrorMaker 2.0

试验

基于 Kafka Connect 框架构建的 MirrorMaker 2.0（也称为 MM2），弥补了以前 Kafka 复制工具中的许多短板。它可以成功地跨集群 geo-replicate 主题数据和元数据，包括偏移量、消费者组和授权命令行（authorization command lines，ACLs）。MM2 能够保留分区，并能检测新的主题和分区。我们很欣赏其所具备的下述逐步进行集群迁移的能力（当从本地集群迁移到云集群时，这种方法尤其有用）：同步主题和消费者组之后，先把客户端迁移到新集群里；然后把生产者迁移到新集群里；最后关闭 MM2，并将旧集群下线。此外，MM2 还可用于灾难恢复和高可用性场景。

25. 用于 Kubernetes 的 OPA Gatekeeper

试验

用于 Kubernetes 的 OPA Gatekeeper 是为 Kubernetes 实现的一个可定制的准入 webhook。它可以确保所有的规则都会被 Open Policy Agent（OPA）执行。我们正在使用 Kubernetes 平台的这个扩展，来为集群添加一个安全层，以便通过提供一个自动化的治理机制，来确保所有的应用都符合定义好的规则。我们的团队喜欢它的可定制化能力。使用 CRD（CustomResourceDefinitions），就可以定义 ConstraintTemplates 和 Constraints。这会使得定义规则和对象（例如 deployments，jobs，cron jobs 等）以及计算中的命名空间变得容易。

26. Pulumi

试验

我们已经看到在各种组织中，使用 Pulumi 的团队数量有所增加。尽管 Terraform 在基础设施编程世界中地位稳固，但 Pulumi 却填补了其中的一个空白。虽然 Terraform 是一个久经考验的常备选项，但其声明式的编程特质，却深受不足的抽象机制和有限的可测试性的困扰。如果基础设施完全是静态的，那么 Terraform 就够用了。但是动态基础设施的定义，则需要使用真正的编程语言。允许以 TypeScript/JavaScript、Python 和 Go 语言编写配置信息（无需标记语言或模板）这一点，就使 Pulumi 脱颖而出。Pulumi 专注于云原生架构，包括容器、无服务器函数和数据服务，并能良好支持 Kubernetes。最近，虽然面临着 AWS CDK 的挑战，但 Pulumi 仍然是该领域唯一的能独立于任何云平台厂商的工具。

27. Sealed Secrets

试验

Kubernetes 原生支持称为“机密”（secret）的键值对象。然而默认情况下，这些 Kubernetes 的“机密”其实并不真的那么机密。尽管它们与其他键值数据分开处理，可以单独采取预防措施或访问控制，且支持在将“机密”存储在 etcd 之前，对其进行加密，但在配置文件中，“机密”是以纯文本字段的形式保存的。Sealed Secrets 提供组合运算符和命令行实用程序，使用非对称密钥来对“机密”进行加密，以便仅在集群中用控制器将其解密。此过程可确保“机密”在 Kubernetes 用于部署的配置文件中不会泄漏。一旦加密，这些文件就可以安全地共享或与其他部署制品一起存储。

28. Vercel

试验

自从首次评估 JAMstack 以来，我们已经看到越来越多这种风格的 Web 应用。然而，当构建传统的动态网站和后端服务的基础设施对 JAMstack 来说太重的时候，我们的团队就会选择 Vercel。Vercel 是一个托管静态网站的云平台。更重要的是，它实现了开发、预览和发布 JAMstack 网站工作流程的无缝衔接。相关的部署配置非常简单。通过和 GitHub 集成，每个代码提交或 Pull Request，都可以触发一个新的带有预览链接的网站部署。这个操作极大地加快了开发过程中的反馈。Vercel 还使用 CDN 来扩展和加速生产环境的站点。值得一提的是，Vercel 背后的团队也在支持另一个广受欢迎的框架——Next.js。

29. Weights & Biases

试验

使用机器学习（ML）平台 Weights & Biases 的实验跟踪、数据集版本控制、模型性能可视化和模型管理功能，能够更快地构建模型。此外还可以将其与现有的机器学习代码集成，从而将实时指标、终端日志和系统统计数据快速传输到仪表盘中，用于进一步分析。我们的团队已经使用了 Weights & Biases，非常喜欢它在模型构建方面的协作功能。

30. Azure 认知搜索

评估

Azure 认知搜索为需要对异构内容进行文本检索的应用程序，提供搜索即服务的功能。它提供基于推送或基于拉取的 API，来上传图像。它能将图像、非结构化文本或结构化文档内容编入索引，也能有限地支持基于拉取的数据源类型。它能以 REST 和 .NET SDK 为基础，提供用来执行搜索查询的 API。搜索时既可以使用简单的查询语言，又可以使用更强大的 Apache Lucene 查询。后者具有字段范围查询、模糊搜索、中缀和后缀通配符查询、正则表达式搜索等功能。我们已经成功地将 Azure 认知搜索与其他的 Azure 服务一起使用，包括在 Cosmos DB 中搜索上传的内容。

31. Babashka

评估

即使现在已经有了各种开发和基础设施工具，但我们还是经常会需要使用脚本，将几项工作粘接起来，或者自动化重复的任务。当前编写这类脚本多使用 bash 或 Python。然而，我们很高兴地看到，Clojure 成为了这类工作令人兴奋的新选择。这都要归功于 Babashka，一个使用 GraalVM 实现的完整的 Clojure 运行时。Babashka 附带的库已经可以涵盖大多数的脚本工具使用场景，并且它还支持加载更多的库。使用 GraalVM 保证了脚本的启动时间与原生工具相当。Babashka 也是少见的支持多线程的脚本环境，可以用于少数特定的场景。

32. ExternalDNS

评估

ExternalDNS 将 Kubernetes 的 Ingress 和 Service 所对应的 DNS 记录，同步给外部的 DNS 提供商。而这项工作原本由 kops dns-controller、Zalando’ s Mate 或者 route53-kubernetes 来完成。由于人们更加青睐 ExternalDNS，后两者目前已被弃用。该工具使得用户可以通过公共 DNS 服务器，来发现 Kubernetes 的内部资源，从而减少了因 Ingress 主机或者 service IP 地址发生改变，而需要更新 DNS 时的一些人工步骤。它支持大量可以直接使用的 DNS 服务提供商，并且正在通过社区支持增加更多 DNS 服务提供商。就像那句老玩笑话所说，万事皆关乎 DNS。

33. Konga

评估

用来管理 Kong API 网关的 Konga，是一个开源 UI 平台。它曾在往期技术雷达中位于试验环。我们的团队喜欢它快捷的配置和丰富的功能，因为这可以使团队能轻松地试验和试用配置。另外它是开源的，这使团队不必担心软件使用许可成本。

34. Milvus 2.0

评估

Milvus 2.0 是一个云原生的开源向量数据库，用于检索和管理由机器学习模型和神经网络所生成的嵌入向量。它支持多种向量索引，以供对音频、视频、图像或其他任意非结构性数据的嵌入向量进行近似最近邻（ANN）搜索。如果有相似性搜索方面的需求，我们推荐对 Milvus 2.0 这款相对较新的数据库进行评估。

35. Thought Machine Vault

评估

我们很少在技术雷达中讨论商用软件，更不用说银行核心平台了。然而，讨论 Thought Machine Vault（与 Thoughtworks 没有任何关系）却是一次破例。这个产品旨在支持良好的软件工程实践，比如测试驱动开发、持续交付和基础设施即代码。开发者通过在 Vault 里用 Python 编写智能合约，来定义银行产品。这与通过 GUI 或专属配置文件或两者兼有，来实现定制化的标准无代码方式截然不同。因为产品是用常规的 Python 代码定义的，因此开发者可以使用一系列工具，包括测试框架和版本控制工具，以此来确保工作是安全而准确的。我们希望更多的金融服务平台的设计，能像上面那样多考虑提升开发者的效率。

36. XTDB

评估

具备双时态图查询的开源文档数据库 XTDB，每条记录都原生支持两个时间轴：*valid* time 时间轴（涉及事实发生时间），以及 *transaction* time 时间轴（涉及事实被数据库处理和记录的时间）。在很多场景中，支持双时态都是有益的，比如当进行用于执行时间感知查询的分析型用例的设计时，当审计事实的历史变化时，当支持分布式数据架构以保证时间点查询的全局一致性时（例如数据网格，当保持数据不变性时，等等。XTDB 以文档形式获取信息，而信息用可扩展数据表示法（EDN，是 Clojure 语言的一个子集）格式表示。XTDB 支持图和 SQL 查询，并可通过 REST API 层和 Kafka 连接等模块进行扩展。我们很高兴地看到，XTDB 的采用率有所增长，另外它也增加了对事务和 SQL 的支持。

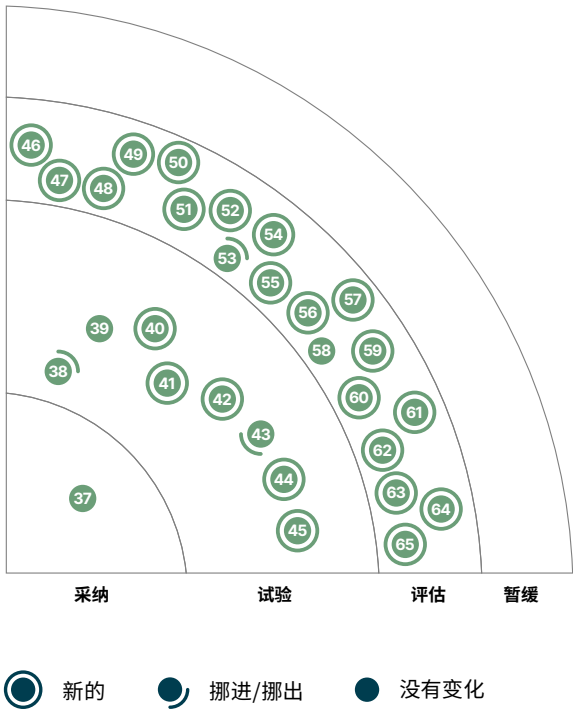
工具

采纳
37. fastlane

试验
38. Airflow
39. Batect
40. Berglas
41. Contrast Security
42. Dive
43. Lens
44. Nx
45. Wav2Vec 2.0

评估
46. cert-manager
47. Cloud Carbon Footprint
48. Code With Me
49. Comby
50. Conftest
51. Cosign
52. Crossplane
53. gopass
54. Micoo
55. mob
56. 现代 Unix 命令
57. Mozilla Sops
58. Operator Framework
59. Pactflow
60. Prefect
61. Proxyman
62. Regula
63. Sourcegraph
64. Telepresence
65. Vite

暂缓
—



37. fastlane

采纳

代码签名是发布 iOS 应用中的一步。虽然这一步已经有 Apple 的工具链支持，但过程仍易出错，充满了意外和不便之处。我们非常高兴地告诉你，fastlane 已经成为了我们自动发布移动应用时的选择。fastlane 为代码签名提供了一个更好的解决方案——match。它已被集成在 fastlane 顺滑的流程中，并且为管理团队的代码签名实现了一个新的方法。这种方法并不将签名密钥缺省存入开发者 macOS 的钥匙串中，而是将密钥与证书存入一个 Git 仓库中。这不仅让新上项目的成员更加方便地设置开发机器，而且在我们的体验中，这也是把代码签名融入持续交付流水线的最简单的方式了。

38. Airflow

试验

近年来，工作流管理工具越来越受到大家的关注，不管它是专注某个特定领域还是领域无关，这样的趋势归功于越来越多的数据处理流水线以及机器学习（ML）模型开发流程的自动化。Airflow 是早期开源的工作流编排引擎之一，它用代码将流水线定义为有向无环图（DAGs），这是对传统 XML/YAML 定义方式的一大改进。尽管 Airflow 仍然是被广泛使用的编排工具之一，但是我们还是鼓励你根据实际情况评估其他工具。例如，Prefect，它的关键特性是支持动态的数据处理任务，任务本身通过 Python 范型函数实现。如果你需要和 Kubernetes 深度集成，那么可以考虑 Argo。如果你需要编排机器学习（ML）工作流，那么 Kubeflow 和 MLflow 可能更合适。考虑到越来越多的新工具不断涌现，再加上 Airflow 本身的功能短板（比如缺少对动态工作流的支持，中心化的流水线调度机制），我们不再推荐 Airflow 作为首选的编排工具。

我们相信，随着流式处理越来越多地用于数据分析和数据流水线，以及通过去中心化的 Data Mesh 来管理数据，人们使用编排工具来定义和管理复杂数据流水线的需求会越来越少。

39. Batect

试验

Batect 持续赢得我们开发者的欢迎，并被很多人作为配置本地开发和测试环境的默认方式。这个开源工具（碰巧由一名 Thoughtworker 开发）基于 Docker，让设置和分享构建环境变得特别简单。这样，Batect 就可以成为你构建系统的入口，取代那个 “check out and go” 方法里无处不在的 go 脚本。Batect 会根据开发者的反馈持续演进，最近它新增了对 Docker BuildKit 和 Shell tab 补全的支持。

40. Berglas

试验

Berglas 是一款用来管理 Google 云平台（GCP）上私密信息的工具。我们此前已经推荐过“密码即服务”作为在现代分布式架构中存储和分享私密信息的技术，GCP 为此提供了 Secret Manager，而 Berglas 与 Secret Manager 配合使用效果良好。这对那些尚未直接集成 Secret Manager 的 GCP 服务来说尤其有用，在这种情况下，另一选择则是自行编写代码或脚本。Berglas 作为命令行工具和库，同时也能在密码即服务以外的场景中派上用场。Berglas 的作者——恰好也正是 HashiCorp Vault 的原作者，他目前正在 Google 工作，但 Berglas 并不是 Google 的官方工具。

41. Contrast Security

试验

Contrast Security 提供一个包括了静态应用安全测试 (static application security testing，SAST)、交互式应用安全测试 (interactive application security testing，IAST)、开源扫描、运行时应用自保护 (runtime application self-protection、RASP) 等多种组件的安全平台。至今它已经有几年历史了，我们也在多个项目上使用过它。关于 Contrast 平台，我们尤其喜欢的一点是它的运行时库分析。它帮助我们定位没被使用的库，这反过来帮助了我们对漏洞安排优先级，并移除掉潜在未使用的库。这和与日俱增的保障软件供应链安全的重要性密切相关。我们也相当喜欢它的 IAST 组件。我们发现它在持续交付 (continuous delivery，CD) 流水线中很有效率，误报更少，并且能发现相当范围的漏洞。

42. Dive

试验

Dive 是一个针对 Docker 镜像的分析工具，用于检查镜像文件中的每一层级、识别每一层上发生的变化。Dive 可以估测出镜像文件的镜像效率和已浪费空间，还能集成到持续集成 (CI) 流水线，根据效率得分或者已浪费空间让构建失败。我们已经在一些项目上使用了该工具，实践证明它的确非常有用，尤其在你构建镜像时，对额外的工具或空间消耗容忍度极低的情况下。

43. Lens

试验

我们的团队一直都在说使用 Lens 可视化和管理 Kubernetes 集群的良好体验。Lens 被称为 “Kubernetes 的 IDE”，它让与 Kubernetes 集群的交互成为可能，而不需要你记住命令或显示文件结构。Kubernetes 可能很复杂，我们知道集群指标和部署负载的可视化工具可以节省时间，并减少维护 Kubernetes 集群的人力成本。Lens 没有将维护 Kubernetes 的复杂工作隐藏在简单操作的 UI 界面之后，而是引入了管理员可以从命令行运行的工具。但是对于正在运行的集群，要谨慎使用 UI 交互方式修改，我们通常倾向于将基础架构变更用代码实现，这样它们是可重复的、可测试的，并且不容易出现人为错误。不过，Lens 确实是一款很好的一站式工具，可以交互式地浏览和理解集群状态。

44. Nx

试验

多年来，我们一直在争论是否应该把 monorepos 放入技术雷达中。每次我们得出的结论都是：monorepos 引入的代价需要进行细致入微的讨论，该技术 “太复杂了以至于无法进入技术雷达”。现在我们看到 JavaScript 社区对 monorepos 越来越感兴趣，例如，在这期播客中讨论的构建由微前端组成的应用程序。这是否是一个好主意，很大程度上取决于你的实际情况，我们当然不想给出一般性建议。我们想要评论的是工具。在团队中，我们看到了纷纷舍弃 Lerna，而强烈倾向于使用 Nx 来管理基于 JavaScript 的 monorepos。

45. Wav2Vec 2.0

试验

Wav2Vec 2.0 是用于语音识别的自我监督学习框架。在这个框架里，模型被分为两个阶段进行训练。首先，它从使用未标记数据的自我监督模式开始，并尝试实现最佳的语音表达。然后它进行有监督的微调整，在此期间，已被标记的数据指导模型去预测特定的单词或音素。使用 Wav2Vec 后，我们发现它的方法非常强大，即使有标记数据的可用性有限，它也可以为区域语言构建自动语音识别模型。

46. cert-manager

评估

cert-manager 是一款在 Kubernetes 集群里管理 X.509 证书的工具。它将证书和签发者建模为最高级的资源类型，并将证书作为服务安全地提供给工作在 Kubernetes 集群上的开发人员和应用程序。得益于对 Let's Encrypt，HashiCorp Vault 和 Venafi 的内置支持，cert-manager 是一款非常有趣的，值得评估的证书管理工具。

47. Cloud Carbon Footprint

评估

利益相关者越来越希望企业考虑到其决策对外部环境的影响，正如环境、社会和公司治理（ESG）投资和员工围绕气候变化积极行动的兴起所证明的那样。迁移到云提供了更高效使用能源的潜力——云提供商有更大的规模来证明对绿色能源和研发的投资是合理的——但云用户的软件抽象的缺点是，这些抽象也隐藏了对能源的影响，因为真实的数据中心被隐藏起来，并且是由另一家公司提供的资金。Cloud Carbon Footprint 是一款新的开源工具，它通过云 API 提供了基于 AWS、GCP 和 Azure 使用情况的碳排放估算的可视化。除了使用 Etsy 的 Cloud Jewels 等启发式方法来估算能源消耗，它还利用了像云区域所在地的基础能源网的碳强度（GCP 已经发布了此数据）这样的公共数据源，将能源消耗换算为碳排放量。这种云区域与基础能源网碳强度的关联，为将高能耗工作负载转移到提供更绿色能源的区域提供了推动力。

48. Code With Me

评估

JetBrains 的协作编码工具 Code With Me 在远程优先的世界越来越受欢迎，因为许多团队都在使用各种 JetBrains 工具。与其他远程协作工具如 VSCode Visual Studio Live Share 相比，Code With Me 为开发团队提供了更好的远程结对编程和协作体验。Code With Me 在邀请队友加入 IDE 项目并实时协作的能力值得探索。但是，我们已经看到了它在无缝重构方面的一些限制，以及高延迟环境中的一些问题。我们将持续在该领域关注这个工具。

49. Comby

评估

本期技术雷达引入了两款使用抽象语法树表示进行搜索和替换代码的工具。它们与 jscodeshift 有相似的定位，但包含适用于多种编程语言的解析器。尽管它们有一些相似之处，但它们在某些方面还是有所不同。其中

Comby 工具的独特之处，在于其简单的命令行界面，该命令行界面是根据 **awk** 和 **sed** 等 Unix 工具的精神设计的。虽然 Unix 命令基于操作匹配文本的正则表达式，但 Comby 使用特定于编程语言结构的模式语法，并在搜索之前解析代码。这有助于开发人员在大型代码库中搜索结构模式。和 **sed** 一样，Comby 可以用新的结构替换它匹配的模式。这对于大型代码库进行自动批量更改，或在一组微服务存储库中进行重复更改非常有用。由于这些工具相当新，我们希望看到一系列尚未发现的创造性用途。

50. Conftest

评估

Conftest 是一款为结构化配置数据编写测试的工具。它使用开放策略代理（OPA）中的策略定义语言 Rego，来为 Kubernetes 的配置文件、Tekton 的 pipeline 定义文件、甚至是 Terraform 的计划文件编写测试。配置是基础设施的的关键部分，我们推荐您评估使用 Conftest 来进行假设验证并得到快速反馈。

51. Cosign

评估

Sigstore 是云原生计算基金会（Cloud Native Computing Foundation，CNCF）旗下的项目，旨在简化软件签名和透明度。其中的 Cosign 用于容器签名及验证。Cosign 不仅支持 Docker 和开放容器计划（Open Container Initiative，OCI）镜像，还支持可以存储在容器注册表中的其他类型镜像。技术雷达介绍过功能类似的 Docker Notary。但 Notary v1 的问题在于需要维护单独的 Notary 服务器，而不能原生集成在容器注册表中。Cosign 将签名与镜像一起存储在注册表中，因此不存在这个问题。目前 Cosign 可以通过 Webhook 与 GitHub actions 及 Kubernetes 集成，并可以进一步集成在流水线中。我们已经在一些项目中使用了 Cosign，效果不错。

52. Crossplane

评估

Crossplane 是基于 Kubernetes Operator 模式实现的另一种类型的工具，但它的副作用延伸到了 Kubernetes 集群之外。在上一期技术雷达中，我们提到了将 Kube 管理的云服务作为一种技术使用，而 Crossplane 正是这样做的。其思想是利用 Kubernetes 控制平面来提供你部署所依赖的云服务，即使它们还没有部署在集群上，例如管理数据库实例、负载平衡器和访问控制策略等。这款工具有两个值得关注的理由。首先，它展示了 Kubernetes 底层控制平面强大而灵活的执行环境。支持的自定义资源范围没有限制。其次，Crossplane 为我们提供了除 Terraform、CDK 或 Pulumi 这些常用方案之外的另一种选择。Crossplane 为主要的云服务提供了一组预定义的 Provider，这些 Provider 涵盖最通用的配置服务。它并不是要试图成为一个通用的基础设施即代码（IaC）工具，而是要成为与 Kubernetes 部署工作相配套的工具。Crossplane 通常与 GitOps 的实践联系在一起，它是独立的，并且允许你在需要管理外部云资源时，仍然留在 Kubernetes 的生态系统中。然而，Crossplane 并不能帮助配置 Kubernetes，你至少需要一个其他的 IaC 工具来引导集群。

53. gopass

评估

gopass 是一个基于 GPG 和 Git 的团队密码管理器。它以 pass 为基础，并添加了多项功能，包括交互式搜索和单个树中的多密码存储。自提到 gopass 以来，我们已经在多个项目中使用它，有时甚至超出了它的极限。我们非常期待的一个功能是弃用保密信息的能力。可发现性是已知问题，但无法将保密信息标记为不再使用让这个问题变得更加复杂。不过，最大的问题是伸缩性。当你有 50 多人的团队多年使用同一个存储库时，我们发现存储库的大小可能会增长到数 GB。在新成员入职时重新加密可能需要半个多小时。潜在的问题似乎是，在我们的团队中，一切都在不断变化：人来人往，密钥转手，架构不断发展，添加新密钥，不再需要旧密钥。当变化较少时，即使对于大量用户，gopass 似乎能很好地工作。

54. Micoo

评估

Micoo 是拥挤的视觉回归测试工具赛道中又一新入竞争者，它是开源的解决方案，独立完备，通过提供 Docker 镜像实现简单快捷的环境设置。Micoo 还为 Node.js、Java 和 Python 提供了不同的客户端，以及 Cypress 插件，方便集成市面上大多数常见的前端 UI 自动化测试框架或解决方案。尽管 Micoo 没有提供某些基于 SaaS 或其他商业解决方案的所有功能，但我们的团队已经广泛使用 Micoo，并获得了积极的体验。他们特别强调了 Micoo 不仅适用于移动端、桌面应用程序，也适用于网页。

55. mob

评估

有时候，你会遇到一个工具，但在意识到之前，你不会觉得有多需要它；mob 就是这样的工具。对很多团队来说，远程结对编程已是常态，生活在这样的世界当中，如果能拥有一个可以在结对或 mob 编程期间帮助无缝切换的工具会非常有效。mob 将所有版本控制的命令和工具隐藏在一个命令行界面后，这让人参与 mob 编程更加容易。mob 也为如何远程加入提供了具体建议，例如在 Zoom 当中“偷偷接管屏幕共享”而不是中断屏幕共享，这能保证参与者的视频布局不会发生改变。这样一个有用的工具，这样周到的建议，还有什么不喜欢的？

56. 现代 Unix 命令

评估

爱上 Unix 有很多理由，其中一个对行业产生深远影响的是 Unix 的哲学：应用程序应当“只做一件事，而且做好它”。在 Unix 中，一组简单的命令可以被管道串在一起形成更复杂的解决方案，这正体现了 Unix 的哲学。近年来，开发者贡献出了越来越多的现代 Unix 命令，他们通常用 Rust 编写，试图把命令变得更小更快。这些命令引入了附加的功能，比如语法高亮，并且充分支持了现代化终端的特性。它们的目的在于，在原生层面帮助开发者更好的和 **git** 集成，并能识别出源码文件，例如，bat 是 **cat** 的替代品，它支持分页和语法高亮；exa 替代 **ls**，它支持显示文件的额外信息；还有 ripgrep，它默认忽略 gitignore、二进制和隐藏文件，是比 **grep** 更快的替代品。Modern Unix 仓库列举了其中的一些命令。我们很喜欢使用这些 Unix 命令，你也应该试试用它们改善命令行使用体验。不过，注意不要在脚本中用它们代替默认 OS 发行版中的标准命令，这会降低脚本在其他机器上的可移植性。

57. Mozilla Sops

评估

把密钥当作普通文本放到版本控制（通常是 Github）中是开发者最常犯的错误之一。在某些难以将误提交的密钥内容从遗留代码库移除的情况下，我们认为 Mozilla Sops 工具所提供的加密文本文件中的密钥功能是很有用的。我们在过去也多次提到有类似功能的工具（Blackbox，git-crypt），而 Sops 因几项功能脱颖而出。比如，Sops 集成了 AWS KMS，GCP KMS，以及 Azure Key Vault 等全托管密钥存储服务，可将其作为加密密钥的来源。Sops 也支持跨平台使用，并且支持 PGP key。这使得用户可以在单个文件级别上对密钥进行细粒度的访问控制。同时，Sops 也以纯文本的方式留下了识别密钥，以便通过 git 来定位和比对文本中的原始密钥。我们始终支持能更容易确保开发者安全的事情。但是，也请记住，你并不需要在一开始就把密钥放到源码中。请参阅我们于 2007 年 11 月发布的雷达内容 Decoupling secret management from source code。

58. Operator Framework

评估

我们持续看到 Kubernates 在新场景中的应用。比如，Kubernetes 被扩展为在集群之外或跨多个基础设施提供商管理运行的资源，或者被用来管理超出 Kubernetes 初始范围的有状态应用程序。这些扩展功能可能正在使用 Kubernetes Operator 模式：构建具有被管理自定义资源所特定领域知识的控制器。比如，管理有状态应用程序的 operator 可以使用 Kubernetes 原始类型，自动执行应用程序部署之外的特定任务，例如恢复、备份和升级数据库。

Operator Framework 是一组开源工具，可简化构建和管理 Kubernetes operators 的生命周期。尽管有多种框架可以帮助你构建 Kubernetes operator，但 Operator Framework 仍然是一个不错的选择。它通过 Operator Lifecycle Manager 模块支持丰富的 operator 生命周期管理；通过 Operator SDK，它可以支持多种语言，自行构建 operator 代码，并提供用于发布和共享 operator 的目录。如果你计划构建 Kubernetes operator，我们建议你尝试使用 Operator Framework，从而可靠地加速你的开发。

59. Pactflow

评估

我们认为对于那些拥有大型复杂 API 生态系统的组织，特别是当他们已经在使用 Pact 的情况下，非常值得评估一下 Pactflow。Pactflow 可以管理那些使用 Pact 编写的工作流程和持续部署测试代码，从而降低通往消费者驱动的契约测试的门槛。多个生产者和大量不同的消费者之间协作的复杂性往往会让人望而却步。很多团队投入了大量的精力，尝试通过手动的方式来解决这个问题，因此我们认为很值得评估一下 Pactflow，看它是否可以帮我们解决这个问题。

60. Prefect

评估

Prefect 是一款数据工作流管理工具，可以轻松地在数据管道中添加语义，如重试、动态映射、缓存和失败通知。你可以将 Python 函数标记为任务，并通过函数调用将它们串联起来构建数据流水线。将 Python API 与一些为

通用数据操作预定义的任务相结合，这样在评估你的数据流水线工作流需求时，Prefect 会是一个值得考虑的选择。

61. Proxyman

评估

它可能并不是你每天都需要的工具，但是当你在诊断令人讨厌的网络问题手忙脚乱时，它能帮你获得功能丰富的 HTTP 调试代理。Proxyman 就是这样一个工具。我们的一些团队已经使用它有段时间了，作为在 macOS 上 Charles 的直接替代品，我们非常喜欢它最新型的界面和证书管理。

62. Regula

评估

基础设施即代码（infrastructure as code，IaC）的其中一个信条是自动化测试。假设我们的测试金字塔强壮，在底层的代码层面的测试有良好的覆盖，那我们能产出更好而且更安全的基础设施。不幸的是，支持这个领域的工具相当稀缺。Conftest 经常被用于测试 Terraform 的 JSON 和 HCL 代码，然而它是一个普通用途的工具。Regula 是一个有吸引力的替代品。和 Conftest 类似，Regula 以 Open Policy Agent 的 Rego 语言编写的规则，来检查基础设施代码的合法性，但它同时提供了一系列的基础类型来对基础设施配置进行特别的验证。因为两个工具都是基于 Rego 语言的，Regula 规则也可以由 Conftest 运行。然而，Regula 有自己的命令行工具，它可以在流水线上不依赖 Conftest 或者 OPA 来运行测试。我们的开发者认为 Regula 可以节约时间并产出更可读、可维护和简练的测试代码。即便如此，两个工具都只验证基础设施代码。一个完整的测试集应该也去测试基础设施本身，以确保代码被正确地翻译了。

63. Sourcegraph

评估

另一个引起我们注意的基于抽象语法树的代码搜索工具是 Sourcegraph。与开源的 Comby 相比，Sourcegraph 是一个商业工具（有 10 个用户的免费套餐）。Sourcegraph 特别适合在大型代码库中搜索、导航或交叉引用。你可以通过 Sourcegraph 的网站访问云托管版本，进而搜索公开可用的开源存储库。Comby 是用于自动执行重复性任务的轻量级命令行工具，而 Sourcegraph 的重点在于理解和导航大型代码库的交互式开发人员工具。与 Comby 类似 **sed** 的界面不同，Sourcegraph 的自动代码重写功能是由 UI 驱动的，允许用户在修改之前查看变更。由于 Sourcegraph 是一项托管服务，因此它还能够持续监控代码库并在匹配发生时发送警报。

64. Telepresence

评估

Telepresence 是一款帮助缩短代码变更反馈周期的工具，在这之前，这些变更通常需要进行部署才能进行适当的测试。开发人员可以用它将本地机器上运行的进程插入到远程 Kubernetes 集群中，这样本地进程可以访问远程集群的服务和特性，本地服务也可以临时替代集群中的服务。

在服务集成设置变得有些笨拙的情况下，Telepresence 可以提高开发人员的生产力，并实现更有效的本地测试。然而，如果你习惯了使用这样一个聪明的工具，那可能会遭遇到更大的问题。例如，如果你是因为无法为本地开发设置所有必要的依赖而使用了 Telepresence，那更应该去调查设置和架构的复杂度。如果它是你进行服务集成测试的唯一方法，那请参考**消费者驱动的契约测试**或者其他自动化的集成测试方式。

65. Vite

评估

快速反馈是优质开发体验的关键。代码更改后，得到反馈前等待的一两分钟可谓最影响开发流程的。可惜随着应用规模和复杂性的增长，流行的前端构建工具大多已经不够快了。之前我们介绍了**esbuild**，它通过编译为本机语言而非 JavaScript 来实现，显著地改善了性能。**Vite** 基于 esbuild 构建，相比其他工具带来了**重大改进**。它主要由两个部分组成：一个开发服务器，基于原生 ES 模块提供了丰富的内建功能，如极快的模块热更新（HMR）；以及一套构建指令，它使用 Rollup 打包你的代码。不同于大多数旧工具，Vite 依赖于 ES 模块，它不提供 shim 和 polyfills，这表示它不兼容那些不支持 ES 模块的旧浏览器。如果要支持旧浏览器，我们部分团队会在开发时使用 Vite，而生产构建时使用其他工具。

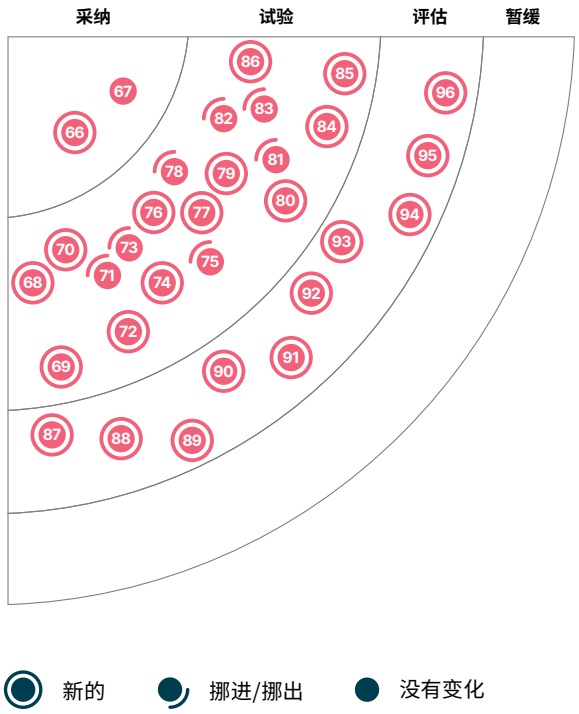
语言和框架

采纳
66. Jetpack Compose
67. React Hooks

试验
68. Arium
69. Chakra UI
70. DoWhy
71. Gatsby.js
72. Jetpack Hilt
73. Kotlin Multiplatform Mobile
74. lifelines
75. Mock Service Worker
76. NgRx
77. pydantic
78. Quarkus
79. React Native Reanimated 2.0
80. React Query
81. Tailwind CSS
82. TensorFlow Lite
83. Three.js
84. ViewInspector
85. Vowpal Wabbit
86. Zap

评估
87. Headless UI
88. InsightFace
89. Kats
90. ksqlDB
91. Polars
92. PyTorch Geometric
93. 乾坤
94. React Three Fiber
95. Tauri
96. Transloco

暂缓
—



66. Jetpack Compose

采纳

仿照苹果推出的 SwiftUI，Google 也为现代 Android 应用提供了完全不同的全新用户界面构建方式 Jetpack Compose。Compose 提供了更强大的工具和一组直观的 Kotlin API，这在多数情况下可以减少代码量，并且比起先定义静态 UI 再填充数据的传统方式，在应用运行时直接创建用户界面会更容易。随着 Compose Multiplatform 和 Kotlin Multiplatform 的出现，开发者现在可以使用统一的工具套件来构建桌面、网页以及原生 Android 应用。此外，Compose 支持 Wear OS 3.0+，而随着 Kotlin Multiplatform Mobile 提供对 iOS 的支持，将来 Compose 也可能会支持 iOS。

67. React Hooks

采纳

React Hooks 引入了一种管理状态逻辑的新方法；鉴于 React 组件相比较类来说更接近于函数，Hooks 接受了这一点并将状态传给函数，而不是将函数作为方法传给带有状态的类。React 应用中状态管理的另一个主要内容是 Redux，我们注意到它已经受到审查，可以看到，在某些时候 Redux 的复杂性并不值得，对于这些情况，使用 Hooks 的简单方法是更可取的。完全靠自己去引入这种实现很快会变得棘手；因此我们推荐考虑结合 React Context 以及 useContext 和 useReducer Hooks，并根据这篇博客文章中解释的路线来实现。

68. Arium

试验

Arium 是一个 Unity 3D 应用测试框架。功能测试是健康的测试金字塔中的一个重要组成部分。以包装 Unity Test framework 而成的 Arium，让您能够在多个平台上为 3D 应用编写功能测试。我们已经在一些项目上成功地使用了它。

69. Chakra UI

试验

Chakra UI 是专门为无障碍设计的一个 React.js 的 UI 组件库。我们很喜欢它关于无障碍设计的功能，比如深色模式以及与无障碍网页倡议（WAI-ARIA）的兼容。此外，它易于测试和定制，为开发提供了良好的体验，加速 UI 解决方案在生产环境中的开发过程。

70. DoWhy

试验

DoWhy 是一个 Python 库，用于执行端到端的因果推理和分析。尽管利用当时存在的变量的相关性，机器学习模型可以根据事实数据进行预测。但在我们需要问如果和为什么问题的场景中，它们是不够的。如果一个变量改变了呢？对结果会有什么影响？因果推理是回答此类问题的一种方法。它估计了因果效应，也就是说，如果我们改变一个原因变量，结果会发生多大的变化。由于实验的成本或限制，当我们无法通过观察和收集 A/B 测试的

数据来得出答案时，就可以采用这种方法。基于使用过去收集的事实和数据以及人们可以在了解领域时做出的假设，DoWhy 库对因果效应进行估计。它使用了一个四步流程，即根据假设对因果关系图进行建模，确定结果的原因，估计因果效应，最后通过反驳结果来挑战这些假设。我们已经在生产中成功地使用了这个库，它是因果估计用例中常用的库之一。

71. Gatsby.js

试验

尽管有几个框架承诺保持与静态站点生成器相同的开发简易性和可扩展性，但我们仍然认为 Gatsby.js 拥有良好的体验。特别是我们已经使用它来构建和部署了可扩展到大量用户的网站，而无需担心容量规划或部署基础设施。我们的开发人员也对其关注可访问性和支持旧系统的特性印象深刻，并且他们可以重用自己的 React.js 经验。总而言之，我们觉得 Gatsby 已经相当成熟，是这个领域的可靠选择。

72. Jetpack Hilt

试验

Jetpack Hilt 最近发布了 1.0 版，我们的使用体验良好。Jetpack Hilt 提供了用于将 Hilt 与各种其他 AndroidX 库（例如 WorkManager 和 Navigation）集成的扩展。它进一步扩展了 Hilt 的范围，为开发人员提供了一种将 Dagger 依赖注入集成到 Android 应用程序中的标准方法。在之前 Radar 中我们提到过作为 Kotlin 原生依赖注入框架的 Koin。虽然在已有的大型代码库中，我们并不建议尝试替换 Koin。但当启动一个新项目时，Hilt 似乎是应该采取的方案。

73. Kotlin Multiplatform Mobile

试验

随着跨平台移动开发的端到端体验日益高效愉悦，构建跨平台移动应用成为了许多组织的一项强有力选择。Kotlin Multiplatform Mobile（KMM）是 JetBrains 提供的 SDK，在利用 Kotlin 跨平台能力的同时，它还提供了工具和特性以优化开发体验。使用 KMM 时，您只需用 Kotlin 对业务逻辑和核心应用程序编写一份代码，然后即可被 Android 和 iOS 应用共享。仅在必要时（如利用原生 UI 元素），您才需要针对特定的平台编写代码，且这些特定代码保存在各个平台的不同视图中。基于 KMM 的迅速发展我们将其移至“试验”环。我们也看到一些组织已经将其作为了默认选项。

74. lifelines

试验

lifelines 是一个用 Python 编写的进行生存分析的库。它最初是为出生和死亡事件而开发的，现在已演变成一个用来预测任何持续时间的完整的生存分析库。除了医疗使用案例（例如用来回答：这个人能活多久？），我们也已经在零售和制造业中用它来回答类似用户对一个服务订阅了多久？或我们应该什么时候做下一个预防性的维护？这样的问题。

75. Mock Service Worker

试验

Web 应用，特别是企业内部应用，通常分为两个部分。用户界面和小部分业务逻辑运行在浏览器中，而大部分业务逻辑、认证和持久化工作运行在服务器中。这两部分一般会通过在 HTTP 上传输 JSON 进行通讯。但请不要误认为这些端点（endpoint）是真正的 API，它们只是当应用要穿越两个运行环境时的实现细节而已。与此同时，它们也提供了一个合适的“接缝”，以便独立测试这些“零件”。当测试 JavaScript 部分时，可以通过像 Mountebank 这样的工具在网络层对其服务端进行打桩（stub）和模拟（mock）。Mock Service Worker 则提供了另一种选择——在浏览器中拦截各种请求，从而进一步简化了手工测试。和 Mountebank 一样，为了测试网络交互，Mock Service Worker 运行在浏览器外部的 Node.js 进程中。由于 GraphQL 的复杂性，此前我们一直难以对它在网络层进行手工模拟。而在 REST 交互之外，Mock Service Worker 对 GraphQL API 的模拟能力也为它锦上添花。

76. NgRx

试验

React 应用中的状态管理一直是技术雷达中反复出现的话题。最近我们阐明了对 Redux 这一流行框架的立场。NgRx 本质上是 Angular 的 Redux。它是一个使用 Angular 构建响应式应用的框架，可以进行状态管理并隔离副作用。我们的团队认为使用 NgRx 很简单，不仅仅因为它由 RxJS 构建，他们还强调了我们从 Redux 中了解到的一个权衡：增加响应式状态管理就会增加复杂性，而只有大型应用才能利大于弊。NgRx 提供的脚手架库 Schematics 以及一组支持可视化状态追踪和回溯调试的工具提高了开发体验。

77. pydantic

试验

起初，Python 添加了类型注解用以支持静态分析。然而，考虑到类型注解和一般注解在其他编程语言中的广泛使用，开发者开始将 Python 的类型注解用于其他目的只是时间问题。pydantic 就是其中一种，用类型注解进行运行时数据验证和设置管理。当接收到 JSON 数据，并需要将其解析为复杂的 Python 结构时，pydantic 确保传入的数据与预期类型匹配，或在不匹配时报告错误。虽然 pydantic 可以被直接使用，许多开发者将它作为最流行的 Python web 框架之一——FastAPI 的一部分来使用。事实上，在 FastAPI 中使用 pydantic 被认为非常有必要，以至于最近有一个 Python 更改提议——旨在降低加载带注解的代码到内存中的成本——被重新考虑，因为它会在运行时破坏类型注解的使用。

78. Quarkus

试验

我们在两年前开始评估 Quarkus，现在我们团队在这方面对它有了更多的经验。Quarkus 是为 OpenJDK HotSpot 和 GraalVM 量身定制的 Kubernetes 原生 Java 技术栈。在过去的两年里，Quarkus 已经连接了 Java 世界中最好的库，并简化了代码配置，给我们的团队提供了一个很好的开发体验。Quarkus 的启动时间非

常快（几十毫秒）并且具有较低的 RSS 内存占用；这归功于它的 container-first 构建方法：它使用提前编译技术在编译时进行依赖注入，这样就避免了反射的运行时间成本。使用 Quarkus 的同时我们的团队也不得不做出一些妥协：在流水线上构建 Quarkus 需要将近 10 分钟；一些依赖注解和反射的功能（如 ORM 和序列化器）也受到了限制。这些妥协一部分是使用 GraalVM 造成的。因此，如果您的应用程序不是为了 FaaS 运行的，那么使用 Quarkus 和 HotSpot 也是一个不错的选择。

79. React Native Reanimated 2.0

试验

如果我们想在 React Native 应用中制作动画，可是使用 React Native Reanimated 2.0。我们之前有 Reanimated 1.x，但它存在与 Reanimated 声明式语言的复杂性相关的问题，并且还有一些与 React Native JavaScript 线程和 UI 线程之间的初始化和通信相关的额外性能成本。Reanimated 2.0 尝试重新构想如何在 UI 线程中绘制动画；它允许我们使用 JavaScript 编写动画代码并在使用名为 animation worklets 的新 API 的 UI 线程上运行它们。它通过在 UI 线程上生成一个辅助 JavaScript 上下文来使之可以运行 JavaScript 函数。我们在需要动画的 React Native 项目中使用它，并且非常喜欢它。

80. React Query

试验

React Query 通常被描述为 React 缺失的数据获取库。数据获取，缓存，同步和服务器状态更新是很多 React 应用中非常普遍的需求，虽然这些需求很容易理解，但在程序中正确的实现却是众所周知的困难。React Query 提供了一种基于 React Hooks 的简便直接的方案。作为应用的开发者，只需要传入一个解析数据的函数，然后把其他的工作都交给这个框架即可。我们喜欢它能开箱即用，又能在需要时进行多样化配置。它的开发工具可以帮助新接触 React Query 的开发人员理解该框架的工作原理，但遗憾的是，它还无法在 React Native 中使用。根据我们的经验，该框架的第三版拥有了线上产品所需要的稳定性。

81. Tailwind CSS

试验

我们的开发人员一直在使用 Tailwind CSS 提高工作效率，并且对其在大型团队和代码库的可扩展性印象深刻。Tailwind CSS 通过较低层次的 CSS 样式类降低复杂性，为 CSS 工具和框架提供了一种替代方法。Tailwind CSS 易于定制，可以适应任何客户的视觉设计。我们还发现它与 Headless UI 搭配得很好。Tailwind CSS 使开发者不需要编写任何新的样式类或 CSS 样式。从长远来看，这将产出更易于维护的代码。Tailwind CSS 在可重用性和自定义创建可视化组件之间，提供了适当的平衡。

82. TensorFlow Lite

试验

自 2018 年雷达中首次提到 TensorFlow Lite 以来，我们已经在一些产品中使用了它。很高兴它与宣传的效果一致。除了将预训练模型集成到移动应用程序中的标准用法，TensorFlow Lite 还支持设备上训练，从而进一步开

辟应用领域。TensorFlow Lite 网站不仅展示了许多常见的应用示例（如图像分类和目标检测），还隐含了诸如姿态估计和手势识别等新交互方式的应用。

83. Three.js

试验

我们第一次提到 Three.js 是 2017 年，在雷达的“评估”环上。在这之后，这个 web 3D 渲染库得到了快速的改进和发展。随着 WebGL API 标准的改进，以及对 WebXR 的支持，Three.js 成为了一个可以用来营造沉浸式体验的工具。与此同时，浏览器对 3D 渲染和 WebXR 设备 API 的支持也得到提升，使得 web 成为一个越来越有吸引力的 3D 内容平台。尽管还有别的 3D 渲染库，我们团队还是更喜欢 Three.js，特别是和 React Three Fiber 一起用的时候，能抽象掉一些底层细节。不过，开发者仍然需要注意性能问题，有时还需要重构数据来优化渲染速度。

84. ViewInspector

试验

当我们使用 SwiftUI 创建界面时，其背后理念是创建一个可以轻松映射到界面元素的视图模型。在这种场景下，大多数测试都可以在模型上完成，而我们只需使用标准的单元测试框架就能编写直观、高效的测试。为了测试模型和视图之间的数据绑定，开发者会使用 XCUITest，这个自动化框架会启动完整的应用并远程控制界面，它能达到目的，测试效果也还算稳定，但是耗时很长。

想要更快地给 SwiftUI 编写单元测试，您可以试试 ViewInspector 开源库，它利用 Swift 开放的反射 API 访问 SwiftUI 创建的底层视图。因此，基于 ViewInspector 的测试只需要实例化一个 SwiftUI 视图，定位到需要测试的界面元素，就可以对元素进行断言测试，而像点击这种基本的交互也可以被测试到。如同许多别的 UI 测试框架，ViewInspector 提供了定位界面元素的 API，您可以指定路径，通过视图层级结构找到元素，也可以使用一系列查找方法定位元素。基于 ViewInspector 的测试往往比用 XCUITest 的测试更简单，运行起来也要快得多。不过，需要注意的是，尽管用 ViewInspector 写测试很简单，您也可能因此过度测试——测试简单的一对一映射逻辑就像在复印代码，没有太大意义。此外，即便 ViewInspector 降低了测试 SwiftUI 代码的难度，也请记住要把大部分逻辑放在模型中实现。

85. Vowpal Wabbit

试验

Vowpal Wabbit 是一个多功能的机器学习库，尽管它最初由雅虎研究院于十多年前创立，但我们仍想强调许多最新的机器学习技术时至今日依旧最早被添加于此。如果对机器学习感兴趣，那么您最好对 Vowpal Wabbit 上的更新多加留意。值得注意的是，近年来微软也对 Vowpal Wabbit 展现出更为浓厚的兴趣，除聘请了一位主要贡献者外，还将该库集成到他们的 Azure 产品中，例如 machine-learning designer 和 Personalizer。

86. Zap

试验

Zap 是一个用于 GoLang 的超高性能的结构化日志库，它比标准的日志实现和其他日志库更快。Zap 既有一个“漂亮”的记录器，提供了结构化和 **printf** 风格的接口，也有一个更快的记录器，仅提供了结构化的接口。我们的团队已经大规模地使用了它，并且很高兴将其推荐为他们的首选解决方案。

87. Headless UI

评估

Headless UI 是为 React.js 和 Vue.js 提供的无样式组件库，它和 Tailwind CSS 诞生于同一个团队。和其他自带默认样式的组件库不同，开发人员不需要订制或者调整默认样式就能开始使用，这一点非常讨人喜欢。组件丰富的功能和完全的可访问性，再加上可以随意添加样式，开发人员能够更高效地专注于业务问题和用户体验。毫无疑问，Headless UI 也可以很好的与 Tailwind CSS 类结合使用。

88. InsightFace

评估

基于 PyTorch 及 MXNet 的 InsightFace 是开源的 2D 及 3D 深度人脸分析工具集。InsightFace 使用前沿且精确的方法进行人脸检测、人脸识别和人脸对齐。InsightFace 最吸引我们的是它为 ArcFace——用于检测两个图像相似性的前沿机器学习模型——提供了一个的最佳实现。带有 ArcFace 的 InsightFace 在 LFW 数据集上获得了 99.83% 的准确率。我们正在使用它进行面部重复数据剔除，效果很好。

89. Kats

评估

Facebook Research 最近发布的 Kats 是一个轻量级的时间序列分析框架。时间序列分析是数据科学中的一个重要领域，包含预测、检测（周期性、异常值和变化点）、特征提取和多变量分析等问题域。在时间序列分析中，一般会使用不同的库进行不同的分析。而 Kats 则致力于提供一组算法和模型，为时间序列分析的所有问题域提供一站式服务。雷达已经介绍过同样来自 Facebook Research 的 Prophet——Kats 提供的用于预测的模型。我们期待在涉及时间序列分析的问题中试用 Kats。

90. ksqlDB

评估

如果你正在使用 Apache Kafka 并构建流处理应用程序，ksqlDB 是一个不错的框架，用来以类似 SQL 的语句编写简单的应用程序。ksqlDB 并不是一个传统的 SQL 数据库。然而，它允许你在现有的 Kafka topics 上使用轻量级类似 SQL 的语句来构建新的 Kafka streams 或 tables。和从传统数据库读取数据类似，Queries 能够拉取数据，或者当有增量变化发生时把结果推送到应用程序。你可以选择将其作为现有 Apache Kafka 原生部分安装并以 standalone server 的方式运行，或者作为 Confluent Cloud 上一个完全托管的服务。我们正在简

单的数据处理场景中使用 ksqlDB。在更复杂的场景中，比如当应用程序需要在代数 SQL 查询之外进行编码的情况下，我们会继续在 Kafka 之上使用诸如 Apache Spark 或 Apache Flink 这些数据处理框架。我们建议在应用程序的简单性允许的情况下尝试使用 ksqlDB。

91. Polars

评估

Polars 是在 Rust 中实现的一种内存 DataFrame 库。与其他 DataFrame 库（如 Pandas）不同，Polars 是多线程、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。

92. PyTorch Geometric

评估

PyTorch Geometric 是 PyTorch 的几何深度学习扩展库。几何深度学习旨在构建能从非欧几里得数据（比如，图网络）中学习的神经网络。基于图网络的机器学习方法在社交网络建模和生物医药领域，特别是在药物发现领域，越来越受到人们关注。PyTorch Geometric 提供了一个易于使用的库，用于设计复杂的图网络问题，比如蛋白质结构的表示。这个扩展库同时支持 GPU 和 CPU，还在最新研究成果的基础上，内置了大量基于图的机器学习算法。

93. 乾坤

评估

微前端自推出以来，一直很受欢迎。然而，如果团队未能保持应用程序从样式技术到状态管理的一致性，则很容易陷入微前端的无序。乾坤，在中文中代表天地，是一个旨在为此提供开箱即用解决方案的 JavaScript 库。乾坤基于 single-spa，允许不同的框架共存于单一应用程序中。它还提供了样式隔离和 JavaScript 沙箱，以确保微应用的样式或状态不会相互干扰。乾坤在社区中获得了一定的关注；我们的团队也在评估它，希望它可以支持更为友好的调试。

94. React Three Fiber

评估

随着 3D 和扩展现实（XR）应用在网页浏览器中的可行性提升及人们对其兴趣的增加，我们的团队已经尝试使用 React Three Fiber 在网页端开发 3D 体验。React Three Fiber 是一个将 React.js 组件和状态模型转译为由 Three.js 渲染的 3D 对象的库。这一方法将 3D 网页编程开放给了更广泛的开发者群体，这部分开发者已经熟悉 React.js 和其丰富的工具以及库的生态系统。但是，在使用 React Three Fiber 开发应用时，我们的团队经常需要命令式地操作 3D 场景。这一点与 React 提供的响应式组件范式兼容得不太好，且开发者仍然必须理解基础的 3D 渲染机制。React Three Fiber 是否能够提供足够的抽象来吸引开发者学习其特性，或者直接使用 Three.js 开发其实更为容易？目前尚无定论。

95. Tauri

评估

Tauri 是 Electron 的替代品，它使用 Rust 工具以及在操作系统 WebView 中渲染的 HTML、CSS 和 JavaScript 组合来构建桌面应用程序。与捆绑 Chromium 的 Electron 不同，用 Tauri 构建的应用程序利用了底层的 WebView，即 MacOS 上的 WebKit，Windows 上的 WebView2 和 Linux 上的 WebKitGTK。这种方法有一些有意思的利弊：一方面，你可以得到运行迅速而小巧的应用程序二进制文件；但另一方面，你需要验证不同系统中 WebView 的兼容性问题。

96. Transloco

评估

Transloco 是一个用来构建多语言应用程序的 Angular 库，它可以在模板中使用，并提供了处理更复杂情况的功能。由于翻译是在运行时按需加载的，Transloco 可以轻松地在 Web 浏览器中实现语言切换。Transloco 还可以通过模板管道对数字、日期等进行本地化处理。

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 成立于 1993 年，如今已从一个小团队发展成为在 17 个国家拥有超过 10,000 名员工的全球领先的软件及咨询公司。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，20 多年来，他们为全球各地的众多合作伙伴倾力服务。

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