ThoughtWorks®

TECHNOLOGY RADAR

有态度的前沿技术解析

Vol.22

thoughtworks.com/radar

#TWTechRadar

贡献者

技术雷达通常由 ThoughtWorks 技术委员在面对面的会议交谈中梳理而成，但由于我们此刻处于疫情期间，因此首次尝试用线上虚拟活动来创建雷达。

技术雷达中国区技术咨询顾问组：

黄进军/伍斌/姚琪琳/张凯峰/包欢/边晓琳/陈璐/陈孟/樊卓文/韩盼盼/李光毅/梁凌锐/梁若琳/刘先宁/闵锐/阮焕/王亦凡/邬倩/吴瑞峰/向博/徐瑾/徐培/杨旭东/易维利/尹尚维/喻晗/张丽/张霄翀/张扬

Rebecca Parsons (CTO)

Martin Fowler (Chief Scientist)

Bharani Subramaniam

Birgitta Böckeler

Camilla Crispim

Erik Dörnenburg

Evan Bottcher

Fausto de la Torre

Hao Xu

Ian Cartwright

James Lewis

Jonny LeRoy

Lakshminarasimhan Sudarshan

Mike Mason

Neal Ford

Ni Wang

Rachel Laycock

Scott Shaw

Shangqi Liu

Zhamak Dehghani

关于技术雷达

ThoughtWorker 酷爱技术。我们对技术进行构建、研究、测试、开源、描述，并始终致力于对其进行改进，以求造福大众。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和对行业有重大影响的技术趋势。

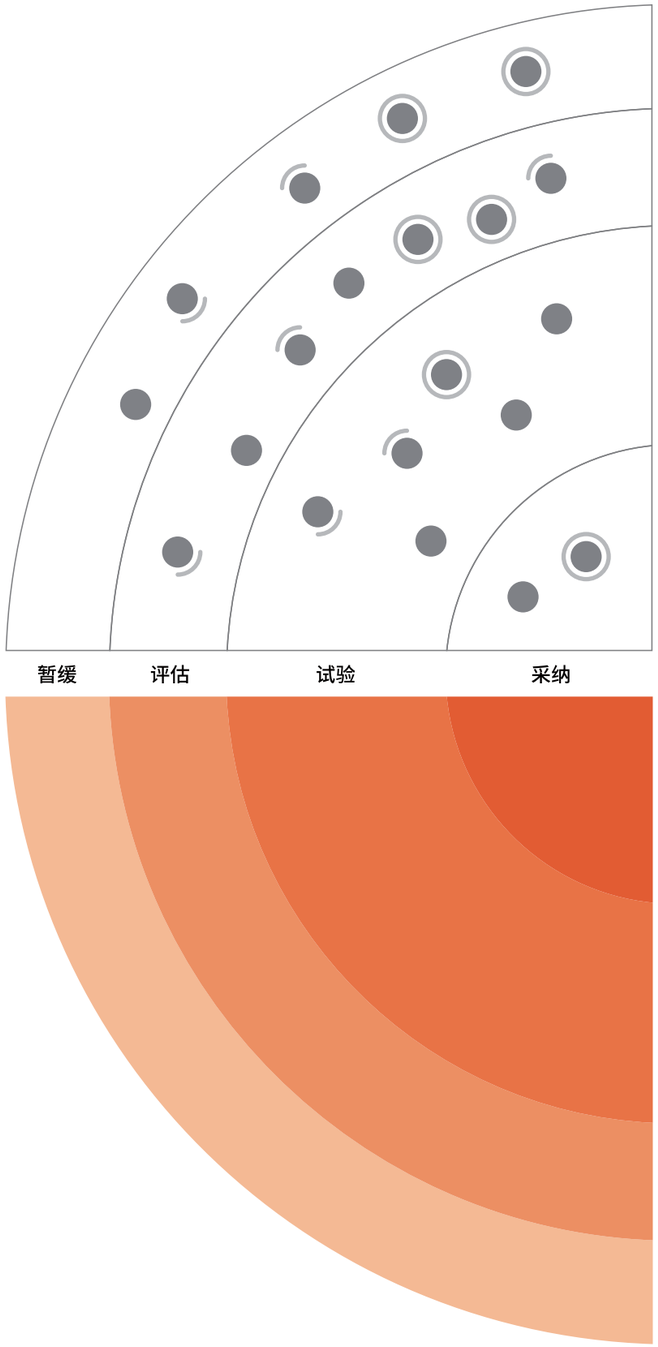
这个雷达以独特的形式记录技术顾问委员会的讨论结果，从首席技术官到开发人员，雷达为各路利益相关方提供价值。这些内容只是简要的总结，我们建议你探究这些技术以了解更多细节。这个雷达的本质是图形性质，把各种技术项目归类为技术、工具、平台和语言及框架。如果雷达技术可以被归类到多个象限，我们选择看起来最合适的一个。我们还进一步将这些技术分为四个环以反映我们目前对其的态度。

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，因此你会发现它们在雷达中的位置也会改变。



新的

移进/移出

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，我们挪走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。

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

本期主题

Zoom 里的大象

“需求是发明之母”——谚语

随着相关技术缓慢成熟，许多公司已经实践了远程办公的想法。但是非常突然地，一场全球瘟疫大流行迫使全世界的公司迅速从根本上改变了他们的工作方式，以此来保护一些生产力。正如很多人已经注意到，“居家办公”与“在疫情期间被迫居家办公”是截然不同的，并且我们认为在这种新的背景下要实现完全的生产力，前方还会有一段路要走。

我们从未相信远程制作一期雷达是可能的，但我们现在已经做到了——这是有史以来第一次在技术雷达委员会成员并未会面的情况下制作的雷达。许多被提议的条目都指向了为一流的远程协作赋能的迫切需求。我们不想忽略房间里的大象（译注：Elephant in the room，是一个英语惯用表达，指显而易见却被集体选择性忽略的问题），以回避谈论这场危机，但做好远程优先的合作是一个深邃而微妙的话题，因此可以确定的是并非我们的所有建议都能符合雷达的传统格式。所以伴随这一期雷达，你还会拥有一期讨论我们以远程优先的方式制作雷达的体验的播客、一份关于远程优先生产力的建议的报告、一场涉及危机中的技术策略的网络研讨会，还有一些指向其他 ThoughtWorks 材料的链接，包括我们的 Remote Working Playbook。我们希望这些材料与网络上其他材料一道，帮助组织在这片未知水域中安全航行。

X 也是软件

我们经常鼓励软件交付生态系统中的其他团队，采用敏捷软件开发团队所引入的、有益的工程实践。但我们也发现这些工程实践在部分领域的应用进展缓慢，所以我们经常讨论这个主题。在本期技术雷达中，我们决定再次强调基础设施即代码以及流水线即代码，并讨论了基础设施配置、机器学习流水线等相关的领域。我们发现，通常负责这些领域的团队并不采用软件设计原则、自动化、持续集成及测试等久经考验的工程实践。我们理解有很多因素，如软件的（本质及偶然）复杂性、缺乏知识，政治原因、缺乏合适工具等，都会对部分工程实践的快速发展造成阻碍。但是，组织可以从敏捷软件交付实践中获得的好处也是显而易见的，所以值得为此付出一些努力。

数据视角的成熟和扩展

在本期技术雷达中，关于数据成熟度的主题横跨了多个条目和象限，特别是围绕分析数据和机器学习的技术和工具。我们注意到自然语言处理（NLP）领域中的许多持续 创新；我们也欢迎机器学习的全生命周期工具套件的出现和持续发展，将经久不衰的 工程实践与迭代中表现良好的工具组合结合起来，表明“机器学习也是软件”。最后，对于像微服务这样的分布式架构，我们看到了对数据网格的极大兴趣，这是一种能在分布式系统中有效地服务和使用大规模分析数据的方法。随着业界更加用心地思考数据在现代系统中的工作方式，我们也为这个领域的总体方向和开放的视角感到欢欣鼓舞，并期待在不久的将来能看到振奋人心的创新。

Kubernetes 生态系统的寒武纪爆发

Kubernetes 市场主导地位的持续巩固，必然促成其生态系统的蓬勃发展。我们在工具、平台和技术象限中，围绕 Kubernetes 讨论了若干雷达条目。这表明该主题已无处不在。例如，Lens 和 k9s 简化了集群管理，kind 有助于本地测试，Gloo 则是一种可选的 API 网关。Hydra 是为在 Kubernetes 上运行而优化了的 OAuth 服务器，而 Argo CD 则使用 Kuberenetes 的原生预期状态管理功能，来实现一个持续交付服务器。这些发展表明，围绕 Kubernetes 完全可以建立一个支持性的生态系统。Kubernetes 虽然提供了关键功能，但其抽象程度对于大多数用户而言，不是太细碎，就是太高深。为解决这些复杂性，相关工具就应运而生，以简化 Kubernetes 的配置和使用，或提供 Kubernetes 核心功能中所缺失的特性。随着 Kubernetes 继续占据主导地位，我们看到其繁荣的生态系统在不断发展和壮大，并能扬长避短。这个生态系统正日趋成熟，我们希望它会朝着一组新的更高层次的抽象去演化，以发挥 Kubernetes 的优势，而不是给出众多选择，让人莫衷一是。

The Radar

技术

采纳
1. 将产品管理思维应用于内部平台
2. 基础设施即代码
3. 微前端
4. 流水线即代码
5. 务实的远程结对
6. 最简特性开关

试验
7. 机器学习下的持续交付
8. 道德偏见测试
9. GraphQL 用于服务端资源聚合
10. 移动微前端
11. 平台工程产品团队
12. 安全策略即代码
13. 半监督学习循环
14. NLP 的迁移学习
15. 使用原生的远程工作方法
16. 零信任架构

评估
17. 数据网格
18. 去中心化身份识别
19. 声明式数据管道定义
20. DeepWalk
21. 通过容器编排管理有状态系统
22. 预检构建

暂缓
23. 云平移
24. 遗留系统迁移的功能一致性
25. 用于业务分析的日志聚合
26. Gitflow 的长期分支
27. 仅快照测试

平台

采纳
28. .NET Core
29. Istio

试验
30. Anka
31. Argo CD
32. Crowdin
33. eBPF
34. Firebase
35. Hot Chocolate
36. Hydra
37. OpenTelemetry
38. Snowflake

评估
39. Anthos
40. Apache Pulsar
41. Cosmos
42. Google BigQuery ML
43. JupyterLab
44. Marquez
45. Matomo
46. MeiliSearch
47. Stratos
48. Trillian

暂缓
49. Node 泛滥

工具

采纳
50. Cypress
51. Figma

试验
52. Dojo
53. DVC
54. 用于机器学习的实验跟踪工具
55. Goss
56. Jaeger
57. k9s
58. kind
59. mkcert
60. MURAL
61. Open Policy Agent (OPA)
62. Optimal Workshop
63. Phrase
64. ScoutSuite
65. 视觉回归测试工具
66. Visual Studio Live Share

评估
67. Apache Superset
68. AsyncAPI
69. ConfigCat
70. Gitpod
71. Gloo
72. Lens
73. Manifold
74. Sizzy
75. Snowpack
76. tfsec

暂缓

语言&框架

采纳
77. React Hooks
78. React Testing Library
79. Vue.js

试验
80. CSS-in-JS
81. Exposed
82. GraphQL Inspector
83. Karate
84. Koin
85. NestJS
86. PyTorch
87. Rust
88. Sarama
89. SwiftUI

评估
90. Clinic.js Bubbleprof
91. Deequ
92. ERNIE
93. MediaPipe
94. Tailwind CSS
95. Tamer
96. Wire
97. XState

暂缓
98. Enzyme

新的　移进/移出　没有变化

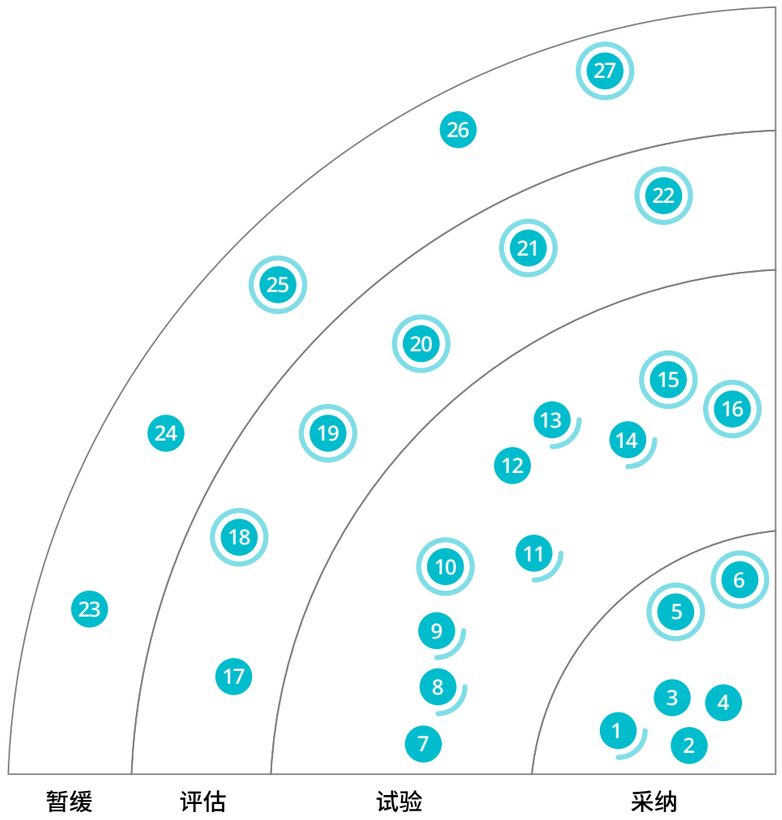
TECHNOLOGY RADAR Vol. 22

技术

技术

将产品管理思维应用于内部平台
采纳

越来越多的公司正在构建内部平台，借此快速有效地推出新型数字化解决方案。成功实施这一战略的企业正将产品管理思维应用于内部平台。这意味着与内部消费者（开发团队）建立共情，并在设计上彼此协作。平台的产品经理要建立路线图，确保平台为业务交付价值，为开发者改善体验。不幸的是，我们也见到了一些不太成功的方式，团队在未经验证的假设、没有内部客户的情况下，打造出的平台犹如空中楼阁。这些平台尽管采用了激进的内部策略，但往往无法充分利用，还耗尽了组织的交付能力。和其他产品一样，好的产品管理就是为消费者创造喜爱的产品。



基础设施即代码
采纳

尽管基础设施即代码是一种相对旧的技术（我们早在 2011 年的技术雷达中就已经介绍过它），但在现代云时代，构建基础设施的行为已经成为了将配置指令传递到云平台的重要组成部分，因此它变得非常重要。当我们说“即代码”时，是指我们在软件领域学到的所有良好实践都应应用于基础设施。使用源代码控制，遵守 DRY 原则、模块化、可维护性以及使用自动化测试和部署都是关键的实践。我们当中具有深厚软件和基础设施背景的人，需要体谅和支持那些没有做过这些的同事。仅仅说像对待代码一样处理基础设施还不够，我们需要确保从软件世界学到的来之不易的经验教训，也能够同样应用到整个基础设施领域。

微前端
采纳

引入微服务令我们受益匪浅，使用微服务，团队可以扩展那些独立部署及维护的服务的交付。遗憾的是，我们也看到许多团队创建了单体前端——一个建立在后端服务之上的大而混乱的浏览器应用程序——这在很大程度上抵消了微服务带来的好处。自从问世以来，微前端持续变得流行。我们已经看到，许多团队采用这种架构的某种形式，来管理多开发人员和多团队的复杂性，以提供相同的用户体验。在去年的六月份，这个技术的发起人之一，发表了一篇介绍性的文章，可以起到微前端参考文献的作用。它展示了这种设计是如何通过各种 Web 编程机制实现的，以及使用 React.js 构建了一个示例应用程序。我们有理由相信，随着大型组织尝试在跨多团队中分解 UI 开发，这种风格将越来越流行。

采纳
1. 将产品管理思维应用于内部平台
2. 基础设施即代码
3. 微前端
4. 流水线即代码
5. 务实的远程结对
6. 最简特性开关

试验
7. 机器学习下的持续交付
8. 道德偏见测试
9. GraphQL 用于服务端资源聚合
10. 移动微前端
11. 平台工程产品团队
12. 安全策略即代码
13. 半监督学习循环
14. NLP 的迁移学习
15. 使用原生的远程工作方法
16. 零信任架构

评估
17. 数据网格
18. 去中心化身份识别
19. 声明式数据管道定义
20. DeepWalk
21. 通过容器编排管理有状态系统
22. 预检构建

暂缓
23. 云平移
24. 遗留系统迁移的功能一致性
25. 用于业务分析的日志聚合
26. Gitflow 的长期分支
27. 仅快照测试

技术

我们坚持结对编程，但是处在后 COVID 的远程工作时代，我们需要使用更多务实的方式来保证效率。

（务实的远程结对）

我们建议使用最简特性开关来代替不必要的复杂的特性切换框架。

（最简特性开关）

流水线即代码
采纳

流水线即代码技术强调，用于构建、测试和部署我们应用程序或基础设施的交付流水线配置，都应以代码形式展现。这些代码应置于版本控制系统中，并切分成包含自动化测试和部署的可复用组件。随着组织逐渐演变为构建微服务或微前端的去中心化自治团队，人们越来越需要以代码形式管理流水线这种工程实践，来保证组织内部构建和部署软件的一致性。这种需求使得业界出现了很多交付流水线模板和工具，它们可以以标准的方式构建、部署服务和应用。这些工具用大多采用声明式交付流水线的形式，采用一个流水线蓝图，来执行一个交付生命周期中不同阶段的任务，如构建、测试和部署，而不用关心实现细节。以代码形式来完成构建、测试和部署流水线的能力，应该成为选择 CI/CD 工具的评估标准之一。

务实的远程结对
采纳

我们坚信结对编程可以提高代码质量，在团队中传播知识，并可以在总体上更快地交付软件。然而在后 COVID 的世界中，许多软件团队将是分布式的或完全远程工作的。因此在这种情况下，我们建议采用务实的远程结对，即根据手头的工具调整结对策略。考虑使用 Visual Studio Live Share 这样的的工具来实行高效、低延迟的协作。仅当结对双方居住的地理位置相对靠近且网络带宽足够高时，才使用高清的屏幕共享。让所处时区相近的开发人员结对工作，而不要无视地理位置。如果出于客观原因导致结对难以进行，也有一些应对措施可以执行，例如独立编码辅以代码审查、基于 pull-request 的协作（但要提防 Gitflow 长期分支），或仅在代码关键部分进行短期结对。以我们多年的经验看来，采用实用主义的远程结对编程是有效的。

最简特性开关
采纳

我们遗憾地发现，人们很少使用特性开关，且经常混淆其类型和使用场景。为了从持续集成中受益，一些团队会使用 LaunchDarkly 等重量级平台来实现特性切换（包括发布切换），即使他们所需要的仅仅是简单地 if/else 条件控制。因此，除非你需要 A/B 测试、金丝雀发布或将特性发布的职责交给业务人员，我们建议使用最简特性开关 来代替不必要的复杂的特性。

机器学习下的持续交付
试验

利用机器学习使业务应用和服务智能化，并不仅仅是训练模型并为其提供服务。它需要实现一整套端到端、持续可重复的模型训练、测试、部署、监控和运维周期。机器学习下的持续交付（CD4ML）是一种可靠的端到端开发、部署和监控机器学习模型的技术。支撑 CD4ML 的基础技术栈包括数据访问和探索工具、工件（例如数据、模型和代码）的版本控制、持续交付流水线、用于各种部署和实验的自动化环境设置、模型性能评估和跟踪，以及模型运作的可观测性。公司可以根据现有的技术栈选择自己的工具集。CD4ML 强调自动化和避免手工交接。CD4ML 是我们开发机器学习模型的默认方法。

道德偏见测试
试验

在过去的一年中，人们对机器学习，尤其是深度神经网络的兴趣发生了转变。到目前为止，人们对各个模型无限潜能的兴奋，驱动了多种工具及技术的发展。然而近期以来，人们越来越担心这些模型可能无意间造成的伤害。例如，模型可以无意中被训练做出有利可图的信用决策，粗暴地排除处于不利地位的申请人。幸运的是，道德偏见测试越来越受关注，这将有助于发现潜在的有害决定。lime、AI Fairness 360 或 What-If Tool 等工具可以帮助发现训练数据中由少数群体导致的偏差，Google Facets 或 Facets Dive 等可视化工具可用于发现训练数据集内的子组。除了道德偏见测试，我们还将 lime（local interpretable model-agnostic explanations，局部可知、模型不可知的解释）用于理解机器学习分类器的预测以及分类器（或模型）的功能。

GraphQL 用于服务端资源聚合
试验

我们看到越来越多的工具（例如 Apollo Federation）支持将多个 GraphQL 端点聚合到一个图中。但必须提醒的是，不要滥用 GraphQL，尤其是作为服务间的协议时。我们的实践是仅将 GraphQL 用于服务端资源聚合。使用这种模式时，微服务会持续发布明确定义的 RESTful API，而聚合服务或 BFF（Backend for Frontends）模式则使用 GraphQL 解析器集成其他服务资源。图的形状需由领域建模实践驱动，以确保在必要时（在每个限界上下文都是一个微服务的情况下）将统一语言局限在子图内。该技术简化了聚合服务或 BFF 的内部实现，同时鼓励对服务进行良好的建模以避免贫血 REST。

移动微前端
试验

自 2016 年雷达对其进行介绍以来，我们已经看到微前端在 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。尽管我们已经看到有的团队在编写自己的框架来支持这种开发风格，但是现有的模块化框架（例如 Atlas 和 Beehive）也可以简化多团队应用开发的集成问题。

平台工程产品团队
试验

采用云计算和 DevOps，虽然提高了团队生产力，减少了对集中式运维团队和基础设施的依赖，但也制约了那些缺乏自管理完整应用和运维技巧的团队。一些组织通过创建平台工程产品团队 来应对这些挑战。这些团队维护着一个内部的应用平台，该平台使交付团队能够自助部署和运维系统，从而减少交付时间和降低技术栈的复杂度。这里的重点是 API 驱动的自服务和支持工具，而交付团队仍然需要对部署在该平台上的应用负责。当组织考虑组建这样一个团队的时候应当非常小心，避免无意中创建一个 独立的 DevOps 团队，也不要把现有托管及运维设施重新打上平台的标签。关于如何最佳地组建平台团队，我们一直在使用团队拓扑中的概念，将项目中的平台团队划分为赋能团队、核心“平台中的平台”团队和关注流量的团队。

安全策略即代码
试验

安全策略是保护我们的系统免受威胁和破坏的规则和程序。例如，访问控制策略定义并强制谁可以在什么情况下访问哪些服务和资源；或者网络安全策略可以动态地限制特定服务的流量速率。当今技术环境的复杂性要求将安全策略视为代码；我们需要定义安全策略脚本、将其加入版本控制中、自动验证、自动部署并监测其性能。Open Policy Agent 这样的工具或者 Istio 之类的平台提供了灵活的策略定义和实施机制，它们都可以支持安全策略即代码的实践。

半监督学习循环
试验

半监督学习循环是一类迭代式的机器学习工作流，它们利用未标记数据中尚待发现的关系，来提升学习性能。这些技术通过不同方式组合已标记和未标记的数据集，从而改进模型。此外，它们还对在不同数据子集上训练出来的模型进行对比。与机器从未标记数据中推断分类的无监督学习，以及训练集完全标记的有监督技术不同，半监督技术利用的是一小部分已标记数据和大量的未标记数据。半监督学习还与主动学习技术密切相关，在主动学习技术中，人们被引导至选择性标记的模糊数据点。因为能够精确标记数据的专家是稀缺资源，并且标记通常是机器学习工作流中最耗时的活动，所以半监督技术不仅降低了训练成本，还使机器学习对于新型用户而言是可行的。我们还看到了弱监督技术的应用，它使用了机器标记的数据，但其可信度低于人工标记的数据。

技术

人们越来越担心机器学习模型可能在无意间造成的伤害。幸运的是，道德偏见测试越来越受关注，这将有助于发现潜在的有害决定。

（道德偏见测试）

我们已经看到微前端在 Web UI 中得到了广泛应用。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。

（移动微前端）

技术

零信任架构（Zero trust architecture，ZTA）是组织针对其所有资产实施零信任安全原则的策略和流程，以及对最佳实践的践行。

（零信任架构（ZTA））

NLP 的迁移学习
试验

该技术之前处于技术雷达的评估维度。NLP（Natural Language Processing），自然语言处理领域的创新在持续快速发展，并且由于无处不在的迁移学习，使得我们可以将这些创新应用到项目中。GLUE 基准测试（一套语言理解任务）的得分在过去几年里有了显著的进步，平均分数从刚发布时的 70.0 上升到 2020 年 4 月处于领导地位的 90.0。我们在 NLP 领域的很多项目，从 ELMo、BERT 和 ERNIE 等预训练模型开始，然后根据项目需求进行微调，可以取得重大进展。

使用原生的远程工作方法
试验

分布式团队有多种形态和设置；对于我们来说，100% 位于同一地点的交付团队反而已经成为例外。我们大多数团队都是多点团队，或者至少有一些团队成员在异地工作。因此，默认情况下使用原生的远程工作方法可以极大地帮助团队提高整体流程和效率。首先要确保每个人都可以访问必要的远程系统。此外，请使用例如 Visual Studio Live Share、MURAL 或 Jamboard 等工具，将在线工作坊和远程结对变成惯例，而不是偶尔为之。但是，“使用原生的远程工作方法”不只是将本地的协同实践平移到数字世界：拥抱更多的异步沟通、围绕决策文档的更多纪律以及“始终远程”会议是我们团队默认采用的其他方法，用于应对随时可能发生的位置变换。

零信任架构
试验

随着资产（数据、功能、基础架构和用户）跨越安全边界（本地主机，多云提供商以及各种 SaaS 供应商），组织的技术版图正在变得越来越复杂。这就要求企业安全计划和系统架构进行范式转变：从基于信任区和网络配置的静态、缓慢改变的安全策略，转变为基于临时访问权限的动态、细粒度的安全策略。

零信任架构（Zero trust architecture，ZTA）是组织针对其所有资产（设备、基础设施、服务、数据及用户）实施零信任安全原则的策略和流程，以及对最佳实践的践行（包括不论网络位置确保所有访问和通信的安全、强制基于最小权限原则并尽可能细粒度控制的策略即代码、持续监控和自动缓解威胁等）。技术雷达介绍了很多赋能技术，例如安全策略即代码，端点安全边车和 BeyondCorp。如果准备进一步了解 ZTA，请参阅 NIST ZTA 和 Google 在 BeyondProd 上发表的文章，以了解更多关于原则、赋能技术组件及迁移模式的信息。

数据网格
评估

数据网格是一种架构和组织范式，它挑战了我们的传统观念，即必须将大量的可分析数据集中起来才能使用，将数据放在一起或让专门的数据团队来维护。数据网格声称，为了让大数据推动创新，数据的所有权必须和将数据作为产品提供服务的领域数据所有者联合起来（在自助数据平台的支持下，抽象数据产品服务所涉及的技术复杂性）；它还必须通过自动化的方式实现一种新的联合治理形式，以支持面向领域的数据产品间的互操作性。去中心化、互操作性以及数据消费者体验，是数据创新民主化的关键。

如果你的组织有大量的领域，包括大量产生数据的系统和团队，或者多种数据驱动的用户场景和访问模式，我们建议你评估数据网格。构建数据网格需要的投资包括构建自服务的数据平台、支持对领域进行组织结构变更以长期维护其数据产品，以及一个激励机制，来奖励将数据作为产品提供和使用的领域团队。

去中心化身份识别
评估

自互联网诞生以来，技术领域已朝着去中心化的方向加速发展。虽然如 HTTP 之类的协议和如微服务或数据网格这样的的架构模式，使得去中心化的实现成为可能，但身份管理仍然保持着中心化的实现。然而，分布式账本技术（DLT）的出现使去中心化身份识别的概念成为可能。在一个去中心化身份识别系统里，实体是离散的可识别单元，例如人、组织和事件都可以自由使用任何共享的信任根。相反，常规的身份管理系统基于集中的权限和注册表，例如公司目录服务、证书颁发机构或域名注册表。

去中心化身份标识符的开发是主要的启用标准，它是全球独特的、持久的的且可加密验证

的自我主权标识符。尽管规模化的去中心化身份标识符的实现仍然很少见，但我们对这一运动的前提感到兴奋，并已开始在架构中使用这一概念。有关最新的实验和行业合作，请查看 Decentralized Identity Foundation。

声明式数据管道定义
评估

许多数据管道都或多或少地使用了 Python 或 Scala 编写的命令式脚本来定义。这样的脚本中包含了各个步骤的逻辑以及将这些步骤串联起来的代码。在使用 Selenium 测试时，出现过类似的情况，而后开发人员发现了 Page Object 模式，后来许多行为驱动开发（BDD）框架都实现了步骤定义与步骤组合之间的分离。现在，一些团队正在尝试为数据工程引入相同的思路。一个独立的声明式数据管道定义（可能是用 YAML 编写的）仅包含一些步骤的声明和顺序。它定义了输入和输出的数据集，并且在需要更复杂的逻辑时决定是否需要以及何时引入脚本。我们发现了该领域的第一个开源工具——A La Mode。

DeepWalk
评估

DeepWalk 是一个有助于将机器学习应用于图的算法。在处理以图表示的数据集时，关键问题之一就是从图中提取特征。这就是 DeepWalk 可以发挥作用的地方。DeepWalk 使用 SkipGram 来构造嵌入式节点，它将图视为一种语言，图中的每一个节点都是该语言中的唯一单词，DeepWalk 通过随机遍历图中有限的节点来将单词构造成句子。这些图嵌入随后便可应用于各种机器学习模型中。我们有些项目需要在图上应用机器学习，DeepWalk 是我们尝试的技术之一。

通过容器编排管理有状态系统
评估

我们建议在利用容器编排平台（例如 Kubernetes）管理有状态系统时要谨慎。有些数据库不能为容器的编排提供原生支持——它们不期望被调度器终结并重新部署到另一台主机。在这样的数据库上构建高可用服务并非易事。因此我们仍然建议在裸机或虚拟机（VM）上运行它们，而不是将其强行适配到容器平台上。

预检构建
评估

尽管相比 Gitflow 来说，我们强烈推荐 CI，但我们知道直接向主干提交然后在 master 分支上跑 CI，在团队过大的时候是低效的。构建会变慢或不稳定，也可能团队会缺乏在本地运行完整测试集的纪律。在这种情况下，一次红色的构建可以阻塞多名或多对开发者的工作。许多团队通常依赖功能分支来绕过这些问题，而不是解决潜在的根本原因——构建缓慢、不能本地运行测试或迫使许多人在同一位置工作的单体架构。我们不鼓励功能分支，因为它可能需要巨大的努力来解决合并冲突，并且还可能在解决冲突的过程中拉长了反馈周期和引入缺陷。取而代之的是，我们提议使用预检构建作为替代方案：它是基于 Pull Request 的构建，针对只在流水线运行期间存在的为每个 commit 建立的微型分支。为了自动化这一工作流，我们已经见到了类似 Bors 这样的机器人程序，它将合并 master 分支和删除迷你分支（当构建成功时）的工作自动化。我们正在评估这个流程，你也应该试试看，但请不要用它去解决错误的问题，因为这样的话可能会带来分支滥用，导致弊大于利。

云平移
暂缓

奇怪的是，有了十多年的云迁移行业经验之后，我们仍然认为有必要呼吁大家警惕云平移。因为它只将云视为托管的解决方案，并且在云上直接简单复制现有的架构、安全实践和 IT 运营模式。这种方式并未意识到云在敏捷性和数字创新方面的优势。云迁移需要有意地跨多个轴向云原生状态转变，并且根据独特的迁移环境，每个组织最终的结果可能会处于从云平移到云原生迁移这样一个波谱中的某个位置。例如，系统架构作为交付敏捷性的支柱之一，我们通常需要对其进行修改。简单地将现有系统平移为容器具有很强的诱惑性。尽管此策略可以加快云迁移的速度，但在创建敏捷性以及交付功能和价值方面却存在不足。云上的企业安全与传统的通过防火墙和分区的基于边界的安全从根本上是不同的，它需要企业迈向零信任架构。同时，它还需要 IT 运营模式的改革，以便于通过自助式

技术

分布式账本技术（DLT）的出现使去中心化身份识别的概念成为可能。

（去中心化身份标识）

在处理以图表示的数据集时，关键问题之一就是从图中提取特征。这就是 DeepWalk 可以发挥作用的地方。

(DeepWalk)

技术

旨在提供技术可观察性的日志通常很难对用户有深刻的理解。

(用于业务分析的日志聚合)

快照测试在遗留系统中的价值是不可否认的，但不应该作为主要测试机制。

(仅快照测试)

的自动化平台安全地提供云服务，使团队能够承担更多的运营责任并获得自治权。最后，组织必须建立起能支撑持续变化的基础，例如同样迁移为应用和基础设施进行持续测试而创建的的流水线。这些将有助于迁移过程，并最终构建一个更健壮和完善的系统，同时也为组织提供了持续演进和改进自身系统的方式。

遗留系统迁移的功能一致性

暂缓

我们发现，越来越多的组织需要替换陈旧的遗留系统，以适应其客户（内部和外部）的需求。我们持续看到的一种反模式是遗留系统迁移的功能一致性，即保留旧版本同样功能的愿望。我们认为这错失了一个巨大的机会。随着时间流逝，旧系统往往会变得臃肿，包含了许多不被用户使用的功能（根据 2014 年 Standish Group 的报告，这一比例为 50%）和随着时间而发展的业务流程。替换这些功能是一种浪费。我们的建议：说服你的客户退一步思考，了解他们的用户当前需要什么，并根据业务结果和指标，对这些需求进行优先排序——这通常说起来容易做起来难。这意味着需要进行用户调研，并采用现代产品开发实践，而不是简单地替换现有的系统。

用于业务分析的日志聚合

暂缓

几年前，出现了新一代的日志聚合平台，该平台能够存储和搜索大量的日志数据，用来发掘运营数据中的趋势和洞见。这种工具有很多，Splunk 是其中最著名的。由于这些平台可在整个应用程序范围内提供广泛的运营和安全可见性，因此管理员和开发人员越来越依赖于它们。当干系人发现可以使用“日志聚合进行业务分析”时，他们便开始乐于此道。但是，业务需求可能会很快超越这些工具的灵活性和可用性。旨在提供技术可观察性的日志通常很难对用户有深刻的理解。我们更喜欢使用专门为用户分析而设计的工具和指标，或者采用更具事件驱动性的可观察性方法，其中收集、存储业务和运营事件的方式可以用更多专用工具来重现和处理。

Gitflow 的长期分支

暂缓

五年前，我们强调了 Gitflow 长期分支的问题。从本质上讲，长期分支与将所有更改持续集成到源代码中的方法背道而驰。并且，根据我们的经验，持续集成对大多数软件开发来说是更好的方法。后来，因为看到不少团队几乎只用长期分支，我们将警惕的态度扩大到 Gitflow 本身。时至今日，我们依然能看到以 Web 系统持续交付为既定目标的团队沉迷于长期分支。因此当看到 Gitflow 的作者现在已经在他的原有文章中添加了一条注释，解释说 Gitflow 当初不是为此类场景而设计时，我们非常高兴。

仅快照测试

暂缓

在遗留系统中工作时，为了保证系统继续运行且不破坏遗留代码功能，快照测试的价值是不可否认的。然而，我们看到了使用仅快照测试作为主要测试机制这种常见但有害的做法。快照测试可以验证组件在 DOM 中生成的确切结果，但不能验证组件的行为。因此，它可能是脆弱且不可靠的，还会催生“仅删除快照后重新生成快照”这样不好的实践。与此相反，您应该通过模拟用户的操作，对组件的逻辑和行为进行测试。Testing Library 系列中的工具也鼓励这种思维方式。

TECHNOLOGY RADAR Vol. 22

平台

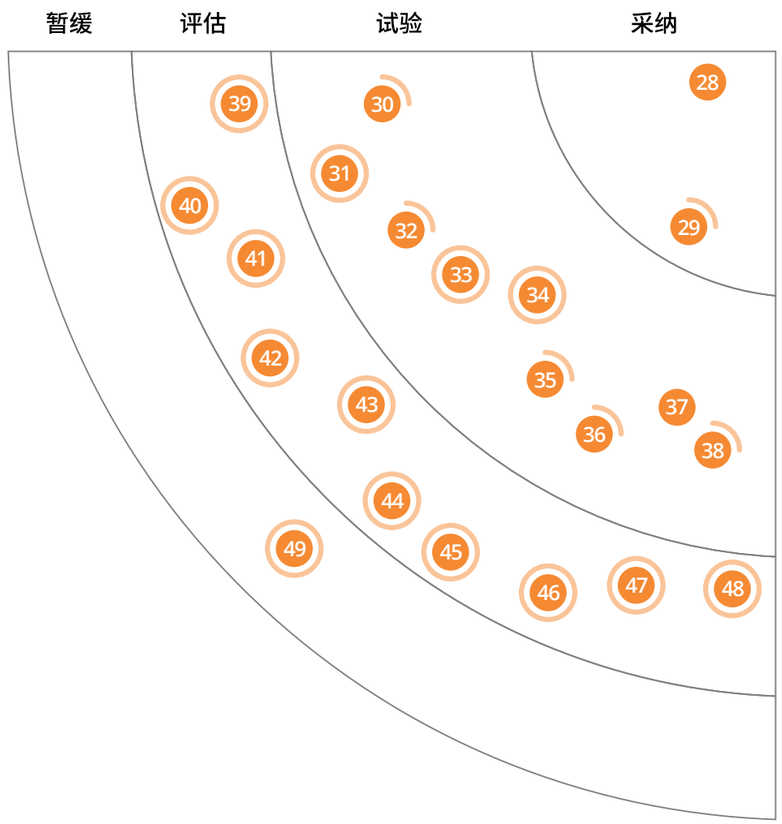
平台

.NET Core
采纳

虽然 .NET Core 之前已进入雷达的采纳环，表明它已成为我们 .NET 项目的默认平台，但我们觉得有必要再次关注一下 .NET Core。随着去年 .NET Core 3.x 的发布，.NET Framework 的大部分特性，现在都已移植到 .NET Core 中。微软在.NET Framework 最新一次发布中，强调.NET Core 是 .NET 的未来。微软已经做了大量工作，来使 .NET Core 达到容器友好。我们大多数基于 .NET Core 的项目，都是针对Linux的，并通常以容器形式进行部署。即将发布的 .NET 5，看起来很有前途。我们对此充满期待。

Istio
采纳

如果正在构建和运行规模化的微服务架构，且已采用 Kubernetes，那么使用服务网格来管理所有架构切面，应当是一个默认选择。在众多服务网格实现中，Istio 是最主流的。它的功能十分丰富，包含服务发现、流量管理、服务到服务以及源到服务的安全性、可观察性（包括遥测和分布式追踪）、滚动发布及韧性机制等。其最新版本易于安装，并提供了控制面板架构，用户体验得到了改善。尽管我们承认维护自己的 Istio 和 Kubernetes 实例，不仅需要足够的知识，还需要一定的内部资源，可能并不适合能力不足的团队，但在我们的诸多项目中，Istio 在保证运维质量的同时，的确降低了大规模微服务的实现门槛。



Anka
试验

Anka 是一组辅助 iOS 和 macOS 应用开发的工具，用于创建、管理、分发、构建和测试可复制的 macOS 虚拟环境。它为 macOS 环境带来了类似于 Docker 的体验，包括即时启动，用于管理虚拟机的命令行工具，以及用于对虚拟机进行版本化和打标记以便进行分发的注册表。我们已经使用 Anka 为客户构建了 macOS 私有云。在需要虚拟化 iOS 和 macOS 的环境时，这是一个值得考虑的工具。

Argo CD
试验

在不评判 GitOps 技术的情况下，我们想针对在 Kubernetes 环境中对应用的部署和监控，来讨论 Argo CD。鉴于其能在

采纳
28. .NET Core
29. Istio

试验
30. Anka
31. Argo CD
32. Crowdin
33. eBPF
34. Firebase
35. Hot Chocolate
36. Hydra
37. OpenTelemetry
38. Snowflake

评估
39. Anthos
40. Apache Pulsar
41. Cosmos
42. Google BigQuery ML
43. JupyterLab
44. Marquez
45. Matomo
46. MeiliSearch
47. Stratos
48. Trillian

暂缓
49. Node 泛滥

平台

尽管这不是新技术，但随着容器化部署微服务的流行，其价值逐渐显露出来。

(eBPF)

这种清晰地将身份管理与 OAuth2 框架其余部分相分离的设计，使 Hydra 与现有身份验证生态系统的集成更加容易。

(Hydra)

Kubernetes 指定的目标环境中，自动将应用部署至所期望的状态，以及在下述方面带给我们的良好体验——排查失效部署的故障原因、验证日志以及监控部署状态，我们建议尝试使用 Argo CD。此外，用户甚至可以通过图表化方式，实时查看集群运行状态、变更如何传播以及 pod 如何被创建和销毁。

Crowdin

试验

大多数需要支持多语言的项目，都始于先以一种语言构建功能，然后再通过电子邮件和电子表格进行离线翻译，来管理其余语言的翻译工作。尽管这种简单的方法可行，但事情很快就会失控。因为接下来，会不得不持续为不同的语言翻译者回答相同的问题，导致翻译者、校对人员和开发团队之间的协作失去活力。在为数不多的几个有助于简化项目本地化工作流程的平台中，Crowdin 是其中之一。使用 Crowdin，在开发团队持续构建功能的同时，平台可以将需要翻译的文本转变为在线工作流。我们喜欢 Crowdin 能促使团队持续和增量地完成翻译工作，而不是在最后阶段大批量地管理这些工作。

eBPF

试验

几年来，Linux 内核已经内置 eBPF（extended Berkeley Packet Filter）虚拟机，并提供将 eBPF 过滤器挂载到特定套接字（socket）的功能。但是 eBPF 能做的远不止包过滤。它允许以很少的开销，在内核中不同的地方，触发自定义脚本。尽管这不是新技术，但随着容器化部署微服务的流行，其价值逐渐显露出来。在这些系统中，服务到服务的通信可能很复杂，因此很难将延迟或性能问题与 API 调用关联起来。现在一些工具会内置 eBPF 脚本，用于收集和可视化数据包流量，或报告 CPU 利用率。随着 Kubernetes 的兴起，出现了基于 eBPF 脚本的新一代安全实施和检测工具，以降低大规模微服务部署的复杂性。

Firebase

试验

谷歌的Firebase 自2016年被我们纳入无服务器架构以来，发生了重大的演变。Firebase 是一个综合性平台，可用于构建移动和 Web 应用，并运行在谷歌可伸缩基础设施上。我们尤其喜欢 Firebase App Distribution 和 Firebase Remote Config。前者可通过持续部署流水线，轻松发布应用程序的测试版本。而后者可以将配置更改，动态地推送给应用程序，而无须重新发布应用程序。

Hot Chocolate

试验

GraphQL 生态系统和社区正不断发展。其中，Hot Chocolate 是用于 .NET（包括 .NET Core 和 .NET Classic）的 GraphQL 服务器。该平台可用于构建和托管 schema，并能使用与 GraphQL 相同的基本组件（数据加载器，解析器，schema，操作和类型）对这些 schema 进行查询。Hot Chocolate 的开发团队最近增添了 schema 拼接功能，允许从单个入口点跨多个 schema（从不同位置聚合而成）进行查询。尽管该功能有被滥用的可能，我们的团队仍对 Hot Chocolate 感到满意，因为它提供了详尽的文档，并且能让我们迅速为客户提供价值。

Hydra

试验

并非每个人都需要自托管的 OAuth2 解决方案，但若需要，可关注 Hydra。这是一款完全符合规范的开源 OAuth2 服务器及 OpenID connect 提供方。Hydra 具有用于开发环境的内存存储支持，以及用于生产环境的关系数据库（PostgreSQL）。无状态的 Hydra 平台，拥有类似九头蛇海德拉的魔力（在希腊神话中，Hydra 意指九头蛇海德拉。即使斩断它的一颗头，也会生出新的头。与之类似，Hydra 平台中无状态的服务器实例出一旦失效，平台会创建新的实例。——译者注），且易于在 Kubernetes 等平台上进行横向伸缩。根据性能要求，在对 Hydra 实例进行横向伸缩时，需要调整数据库实例的数量。由于 Hydra 并未提供任何开箱即用的身份管理解决方案，所以可以通过一个整洁的 API，将现有的任何身份管理系统，与 Hydra 集成在一起。这种清晰地将身份管理与 OAuth2 框架其余部分相分离的设计，使 Hydra 与现有身份验证生态系统的集成更加容易。

OpenTelemetry
试验

OpenTelemetry 是一个开源可观测性项目，它合并了 OpenTracing 和 OpenCensus。OpenTelemetry 项目包含规范、库、代理和其他组件，以便从服务中捕获遥测信息，更好地观察、管理和调试服务。它涵盖了可观测性的三大支柱——分布式跟踪、指标和日志记录（目前处于 beta 测试版），且其规范通过 correlations 连接这三大支柱。这样就可以使用指标 来查明问题，找到相应的跟踪信息，以定位问题，并最终研究相应的日志，以查找确切的根本原因。OpenTelemetry 组件可以连接到后端可观察性系统，例如 Prometheus 、Jaeger 及其他系统。OpenTracing 的形成，是朝着融合标准化与工具简化而迈出的积极一步。

Snowflake
试验

对我们的很多客户来说，Snowflake 已被证明是一个健壮的 SaaS 大数据存储、数仓或数据湖解决方案。它拥有先进的架构，来扩展存储、算力和服务，以加载、卸载和使用数据。它也非常灵活，支持结构化、半结构化和非结构化数据。它为不同的访问模式（如用于数据科学的 Spark 和用于分析的 SQL），提供相应的 connector （数量正不断增加）。它也能运行在多种云平台上。对许多客户而言，我们建议公共系统（如大数据存储）要使用托管服务。然而，如果出于风险和法规的考虑，无法使用托管服务的话，那么 Snowflake 对于具有大数据量和大处理量的公司，是个不错的选择。我们已经成功地在中型项目中使用了 Snowflake，但尚未在数据需要跨组织部门存储的大型生态系统中尝试过。

Anthos
评估

我们看到了企业云策略的以下转变，即从无意而为的混合云，或从整个云的迁移，转向有意而为且精密复杂的混合云、多云或可移植云。在这种转变中，企业会从多个维度，来建立和执行下述云策略——基于风险、控制能力和性能情况，来决定将在何处托管其各种数据和功能；如何在降低运维成本的同时，充分利用企业内部的基础设施；如何既充分利用多个云提供商及其独特的差异化服务，又不会给构建和运行应用程序的用户带来复杂性和摩擦。

Anthos 是谷歌针对上述转变而创建的平台。它通过一系列开源技术（例如 GKE、Service Mesh 和基于 Git 的配置管理系统），为实现混合云和多云策略，提供高级管理和控制平面（plane）。它支持在不同的托管环境（包括 Google Cloud 和本地硬件）上，运行可移植的数据流量和相关功能。尽管其他云提供商拥有类似平台，但 Anthos 打算在支持混合云的基础上更进一步，使用开源组件来让可移植云成为可能（但这还有待观察）。我们看到人们对 Anthos 的兴趣正在上升。尽管谷歌的托管混合云方案看似很有前景，但这并不是灵丹妙药，因为需要对现有云和本地系统进行更改。针对考虑使用 Anthos 的客户，我们的建议是，在 Google Cloud 生态系统的服务和其他选项之间，进行权衡取舍，以保持所选的服务，能得到合理的中立和控制程度。

Apache Pulsar
评估

Apache Pulsar 是一个开源的 pub-sub（发布-订阅）消息与流媒体平台，与 Apache Kafka 在同一领域展开竞争。它提供了我们所期望的功能，如低延迟的异步及同步消息传递，可进行容量伸缩的消息持久化存储，以及多种客户端程序库。Pulsar 吸引我们的地方，是能轻易实现容量伸缩，尤其适合在多用户类型的大型组织中使用。Pulsar 原生支持多租户、异地备份、基于角色的访问控制和计费隔离。此外，我们还期望 Pulsar 能解决高容量数据系统中，消息日志无限增长的问题。在这些系统中，事件一旦持久化，就需要无限期地保存，并且订阅者可以从历史节点开始订阅消息。这是通过一个分层存储的模型来实现的。尽管 Pulsar 对于大型组织是一个有前景的平台，但依然有提升的空间。目前，安装 pulsar 需要管理 ZooKeeper 和 BookKeeper 以及其他工具。我们期待随着 Pulsar 的日益普及，用户很快能够得到更广泛的社区支持。

平台

OpenTelemetry 项目包含规范、库、代理和其他组件，以便从服务中捕获遥测信息，更好地观察、管理和调试服务。

（OpenTelemetry）

Anthos 是谷歌针对企业云策略转变而创建的平台。它支持在不同的托管环境（包括 Google Cloud 和本地硬件）上运行相关功能。

（Anthos）

平台

BigQuery ML 降低了使用机器学习做出预测和推荐的门槛，特别是针对一些需要快速探索的场景。

（BigQuery ML）

Cosmos
评估

自从我们在技术雷达中，对区块链这一领域作出初始评估以来，该领域技术的性能有了极大的提升。然而，目前仍然没有哪个区块链平台能够达到“互联网级别”的吞吐量。各种区块链平台相继发展，产生了不少新的数据和价值孤岛。这使得区块链社区的关键主题一直是跨链技术。区块链的未来，可能是由若干独立且并行的区块链而组成的网络。这也是 Cosmos 的愿景。Cosmos 发布了 Tendermint 和 CosmosSDK，以使开发人员可以定制独立的区块链。这些并行的区块链，可以通过 IBC（Inter-Blockchain Communication，区块链间通信协议）和 Peg Zone 来交换价值。对于 CosmosSDK，我们的团队拥有丰富的经验。而 IBC 协议正在日趋完善。这种架构可以解决区块链的互操作性和可伸缩性的问题。

Google BigQuery ML
评估

通常需要先编写代码将数据带入机器学习模型，才能得出训练和预测结果。而 Google BigQuery ML 能将模型带入数据，从而反转了这一模式。Google BigQuery 是一个数据仓库，能针对数据分析场景，使用 SQL 进行大规模查询。Google BigQuery ML 扩展了此功能及其 SQL 接口，通过利用 BigQuery 数据集，来创建、训练和评估机器学习模型，并最终运行模型预测，以创建新的 BigQuery 数据集。Google BigQuery ML 默认支持部分模型，例如用于预测的线性回归（linear regression），或用于分类的二元和多类回归（binary and multiclass regression）。另外，它还能导入已经训练好的 TensorFlow 模型（但功能有限）。尽管 BigQuery ML 及其基于 SQL 的方式，降低了使用机器学习做出预测和推荐的门槛（尤其针对一些需要快速探索的场景），但导致需要作出艰难的权衡——这种方式不利于模型训练的其他方面，例如道德偏差测试（ethical-bias-testing），可解释性和机器学习的持续交付。

JupyterLab
评估

JupyterLab 是 Jupyter 项目的下一代基于 web 的用户操作界面。对于 Jupyter Notebook 的长期用户来说，JupyterLab 也值得一试，因为它提供一个用于 Jupyter Notebook、代码和数据的交互式环境。我们将 JupyterLab 视作 Jupyter Notebook 的一次进化。JupyterLab 通过扩展其原有功能（允许将代码、可视化和文档存于一处），来提供更好的体验。

Marquez
评估

Marquez 是相对年轻的开源项目，用于数据生态系统中元数据的采集和托管。它使用简单的数据模型，来表现诸如世袭（lineage）、上下游数据处理作业及其状态之类的元数据，并用灵活的标签标记数据集的属性。Marquez 提供了简单的 RESTful API 来管理元数据，从而简化了与数据生态系统中其他工具集的集成。

我们已将 Marquez 作为起点，轻松对其进行扩展，以适应需求，例如执行安全策略，并改用其领域语言等。如果需要一个小巧简单的工具，来完成数据处理作业和数据集的存储和可视化，那么 Marquez 是一个很好的开始。

Matomo
评估

Matomo（前身为 Piwik）是一款开源的网站分析平台，可以完全控制对分析数据的访问。可以自托管 Matomo，并保护网站分析数据的安全，防止第三方非授权访问。Matomo 还可以轻松地将网站分析数据与内部数据平台集成，为用户需求量身定制使用模型。

MeiliSearch
评估

MeiliSearch 是一个快捷、易用且易部署的文本搜索引擎。多年来，Elasticsearch 在具备容量伸缩特点的文本搜索领域，已成为流行选择。但是，如果数据量不大，无须进行分布式的文本搜索，但仍然想实现一个快捷的容错搜索引擎，那么就建议评估一下 MeiliSearch。

Stratos
评估

Ultraleap（前身为 Leap Motion）在 XR（Extended Reality，扩展现实）领域一直处于领先地位。它创造了出色的手部跟踪硬件，使用户的手可以融入虚拟现实世界中。Stratos 是 Ultraleap 的底层触觉、传感器和软件平台。它可以使用定向超声波，在空中产生触觉反馈。比如响应驾驶员的手势，以调节车内空调，并在交互界面上实现触觉反馈。看到这项技术，以及富有创造力的技术人员将该技术应用于各种场景，我们十分欣喜。

Trillian
评估

Trillian 是一种可加密验证的集中式数据存储。在去信任化和去中心化的环境中，可以使用基于区块链的分布式账本。然而，在企业应用环境中，当不需要使用会耗费大量 CPU 资源的共识协议时，建议尝试一下 Trillian。

Node 泛滥
暂缓

技术都有被滥用的趋势，尤其是广为流行的技术。比如现在所见到的“Node 泛滥”现象，就是一种随意或误用 Node.js 的趋势。其中有两点很突出。首先，我们经常听到这样的说法——应该使用 Node，这样只用一种编程语言，就能完成前后端的所有代码。对此，我们还是坚持认为多语言编程是一种更好的方法，尽管我们也将 JavaScript 作为一等语言。其次，我们经常听到团队将性能作为选择 Node.js 的理由。这种观点已经被大量比较合理的基准数据所否定，但其来源还是有历史原因的。当初，Node.js 变得流行时，还是首个使用非阻塞编程模型的主流框架。该模型让 Node.js 非常适合 IO 密集型任务（我们在 2012 年的 Node.js 文章中提到了这一点）。由于单线程的特点，Node.js 从来就不是计算密集型任务的理想选择。而现在，其他平台已经拥有功能强大的非阻塞框架（其中一些已经配备既优雅又现代的 API）。所以性能已不再是选择 Node.js 的理由。

平台

MeiliSearch 是一个快捷、易用且易部署的文本搜索引擎。如果数据量不大，无须进行分布式的文本搜索，那它是一个理想的选择。

（MeiliSearch）

Stratos 是 Ultraleap 的底层触觉、传感器和软件平台，在 XR（Extended Reality，扩展现实）领域一直处于领先地位。

(Stratos)

TECHNOLOGY RADAR Vol. 22

工具

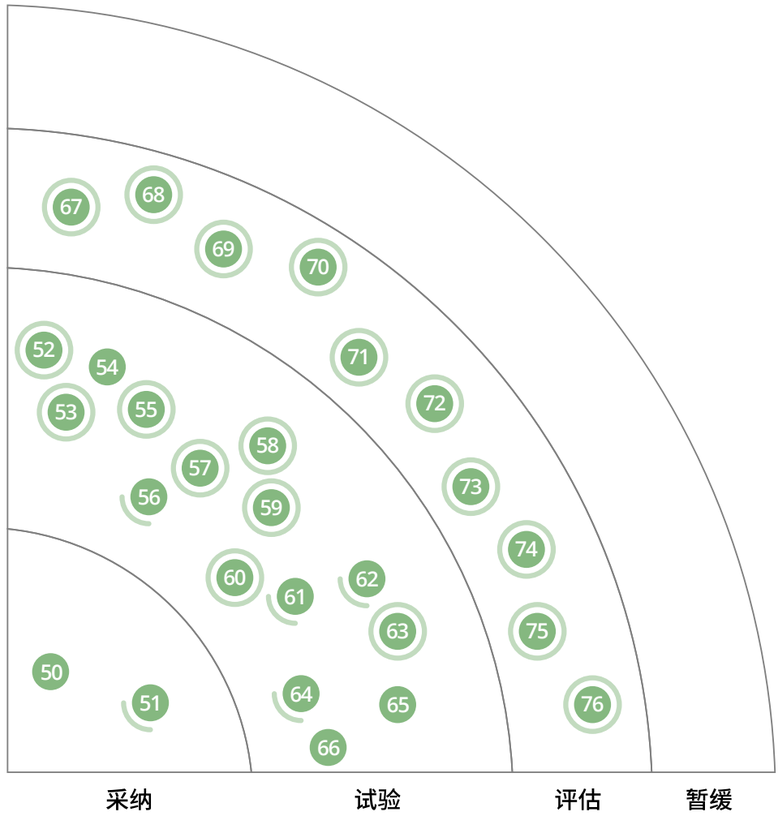
工具

Cypress
采纳

Cypress 在我们的团队中备受青睐，作为健康的测试金字塔的一部分，我们的端到端测试一直是由开发人员自行管理。我们之所以决定在本期技术雷达中再次提起它，是因为 Cypress 最近的版本增加了对 Firefox 的支持，我们强烈建议在多种浏览器上进行测试。Chrome 和基于 Chromium 的浏览器所处的主导地位，已经产生了一个令人担忧的趋势，即团队似乎只使用 Chrome 进行测试，而这可能会导致一些糟糕的意外。

Figma
采纳

不论是对设计师，还是多角色团队而言，Figma 都已被证明是协作设计的首选工具。它允许开发人员和其他角色通过浏览器查看和评论设计，而无需使用桌面版本。和它的竞争对手（如 Invision 或 Sketch）相比，Figma 将版本控制、协作设计和设计分享这些功能都集中到一个工具上，这使得我们的团队更容易一起想出新点子。我们的团队发现 Figma 十分有用，特别是在开启和促进远程分布式设计工作方面。除了实时设计和协作功能外，Figma 还提供了一个 API 以帮助改善 DesignOps 流程。



Dojo
试验

几年前，Docker（通常是跟容器一起）彻底改变了我们对于打包、部署和运行应用程序的看法。尽管在生产环境中情况有所改善，但是开发人员依然需要花费大量时间去设置开发环境，并且经常遇到“但它在我的机器上是好的”这类问题。Dojo 旨在通过 Docker 镜像的版本化和发布来创建标准的开发环境，并以此来解决这个问题。我们的几个团队已经借助 Dojo 来简化从本地开发到生产流水线中的开发、测试和构建过程。

DVC
试验

在 2018 年，我们在可重复分析的版本化数据中提到了 DVC。时至今日，它已成为机器学习（ML）项目中管理实验的热门工具。由于 DVC 是基于 Git 的，因此对于软件开发人员

采纳
50. Cypress
51. Figma

试验
52. Dojo
53. DVC
54. 用于机器学习的实验跟踪工具
55. Goss
56. Jaeger
57. k9s
58. kind
59. mkcert
60. MURAL
61. Open Policy Agent (OPA)
62. Optimal Workshop
63. Phrase
64. ScoutSuite
65. 视觉回归测试工具
66. Visual Studio Live Share

评估
67. Apache Superset
68. AsyncAPI
69. ConfigCat
70. Gitpod
71. Gloo
72. Lens
73. Manifold
74. Sizzy
75. Snowpack
76. tfsec

暂缓

工具

Jaeger 是一个开源的分布式追踪系统。我们在 Kubernetes 上成功的将 Istio 和 Envoy 与 Jaeger 集成，并且很喜欢 Jaeger 的 UI。

(Jaeger)

k9s 为 kubectl 的所有功能都提供了一个交互界面。此外，它并不是一个 Web 应用程序，而是运行在终端窗口中的。

(k9s)

来说，DVC 无疑是一个备感熟悉的环境，他们可以很容易地将以往的工程实践应用于 ML 实践中。由于 DVC 实现了对处理数据的代码以及数据本身的版本管理，而且还实现了对流水线的各个阶段的追踪，因此能在建立建模活动秩序的同时可以不扰乱分析师的工作流程。

用于机器学习的实验跟踪工具
试验

机器学习的日常工作通常可以归结为一系列的实验，包括选择建模方法和网络拓扑，训练数据以及优化调整模型。数据科学家必须利用经验和直觉来做出一些假设，然后去评估这些假设对模型的整体性能的影响。随着这种实践的成熟，我们的团队发现对“用于机器学习的实验跟踪工具”的需求日益增长。这些工具可以用于帮助研究人员跟踪实验，从而使这些实验变得更加的有条不紊。尽管这个领域还没有出现明确的赢家，但是已经出现了 MLflow 这类的工具以及诸如 Comet 和 Neptune 这样的平台，它们使得整个机器学习的工作流程变得更加的严谨和可重复。

Goss
试验

Goss 是一个供应测试工具，往期的技术雷达在介绍 TDD 开发容器脚本时曾提到过它。与 Serverspec 相比，Goss 的功能还不够完备，因此还不足以作为 Serverspec 的替代方案。可是如果它的功能恰好能满足需求，考虑一下它也未尝不可，更何况它非常小巧，开箱即用，不像 Serverspec 一样还需要专门的 Ruby 运行环境。使用 Goss 这种工具的一个常见的反模式就是分开维护代码和测试，这样在实际的基础设施即代码文件发生变化的时候，也必须相应地修改测试断言。这种测试的维护工作量非常大，而且因为代码和测试之间的严格一致性，当工程师更新了一个而忘记更新另外一个的时候往往会导致测试失败。而且这些测试很少能发现真正的问题。

Jaeger
试验

Jaeger 是一个开源的分布式追踪系统。类似于 Zipkin，它的灵感来自于谷歌的 Dapper 论文，并且遵循 OpenTelemetry 规范。我们在 Kubernetes 上成功的将 Istio 和 Envoy 与 Jaeger 集成，并且很喜欢 Jaeger 的 UI。Jaeger 暴露了 Prometheus 格式的追踪指标，以便其他工具使用它。然而，新一代的工具，如 Honeycomb，将追踪和度量集成到单个可观测性流中以支持更简单的聚合分析。Jaeger 在 2017 年加入了 CNCF，并且最近被提升到 CNCF 的最高成熟度级别，这表明它已被广泛部署到生产系统中。

k9s
试验

我们仍然是基础设施即代码的热心拥护者，并且我们始终坚信，一个健壮的监控解决方案是运营好一个分布式应用的前提。有时候，一个类似 AWS Web 控制台这样的交互式工具是很好的补充。这类工具使我们能够即兴地浏览各种资源，而无需记住每一个似是而非的命令。但是，使用交互式工具频繁地进行手动修改仍然是一个值得商榷的实践。对于 Kubernetes，我们现在有了 k9s，它为 kubectl 的所有功能都提供了一个交互界面。此外，它并不是一个 Web 应用程序，而是运行在终端窗口中的，它的操作界面会唤起我们中的某些人关于 Midnight Commander 的美好回忆。

kind
试验

kind 是一个用于在 Docker 容器节点中运行本地 Kubernetes 集群的工具。通过与 kubetest 集成，kind 使 Kubernetes 中的端到端测试变得很简单。我们已经借助 kind 创建临时性的 Kubernetes 集群，在持续集成（Continuous Integration，CI）管道里测试 Kubernetes 中的资源，例如控制器和自定义资源（Custom Resource Definitions，CRDs）。

mkcert
试验

mkcert 是一个用于创建本地信任的开发证书的便捷工具。在本地开发环境中使用真实的 CA（Certificate Authority，证书颁发机构）签发的证书，是非常困难的，特别是对于像 example.net、localhost 或者 127.0.0.1 这样的主机来说，使用真实的 CA 签发的证书是不可能的。在这样的情况下，自签发的证书可能是唯一的选择。mkcert 可以生成自签发的证书，并把本地 CA 安装到系统根证书库中。对于本地开发和测试以外的所有情况，我们

强烈建议使用真实的 CA 签发的证书以避免信任问题。

MURAL
试验

MURAL 自诩为“视觉协作的数字工作空间”，并允许团队在基于白板和便利贴构建的共享工作空间内进行交互。它的功能包括投票、评论、注释和“跟随演讲者”。我们特别喜欢它的模板特性，它允许主持人设计并与团队重用引导会话。所有主流的办公协作软件在这个领域都有工具可以使用（例如，谷歌 Jamboard 和微软白板），这些都是值得研究的，但我们发现 MURAL 使用起来非常顺畅、方便和灵活。

Open Policy Agent（OPA）
试验

在我们为客户构建的许多分布式云原生解决方案中，Open Policy Agent（OPA）已经迅速地展现了它的价值。OPA 提供了一套统一的框架和语言，用于声明，实施和控制云原生解决方案中各个组件的策略。它是实现安全策略即代码的工具中的一个很好的例子。无论是在 K8s 集群中部署资源，还是在服务网格中跨服务执行访问控制，抑或是通过代码精准地控制应用资源访问，在很多场景中 OPA 都给我们提供了非常顺畅的体验。最近的一个商业产品 Styra 声明式授权服务「Styra’s Declarative Authorization Service（DAS）」，通过向 K8s 的 OPA 中添加管理工具（或者说控制平面），预先构建的策略库，策略影响分析和日志记录等功能，使企业采用 OPA 变得更加简单。我们期待 OPA 的成熟和扩展，希望它可以从一个可用的服务演变成一个（大型的）以数据为中心的解决方案。

Optimal Workshop
试验

为了更好地构建我们的产品，用户体验研究需要对数据进行收集和分析。我们的团队发现 Optimal Workshop 非常实用，因为它让验证产品原型和配置数据收集的相关测试变得很简单，因此也能帮助我们做出更好的决策。诸如首次点击，卡片排序，用户交互热力图等功能，都能帮助我们在验证产品原型的同时，优化网站导航和信息展示。因为它支持分布式团队的远程用户体验研究，因此也成为了分布式团队的一个理想工具。

Phrase
试验

正如在 Crowdin 的描述中所提到的，现在可以使用多种平台来管理产品的多语言翻译，而无需再通过电子邮件来发送大型的电子表格。在这些平台中，我们的团队报告了使用 Phrase 的良好体验，并强调它对所有关键用户群体都非常易用：翻译人员使用的是一个非常方便的浏览器 UI；管理者也可以在这个 UI 上添加新的字段，并与其他团队的翻译进行同步；开发人员可以在本地或通过构建流水线来访问 Phrase。另一个值得一提的特性是，Phrase 还通过使用标签实现了翻译的版本管理，这使得在实际产品中对比不同版本的翻译成为可能。

ScoutSuite
试验

ScoutSuite 是基于 Scout2（在2018年技术雷达上出现过）的一个增强版工具，提供跨 AWS、Azure、GCP 和其他云提供商的安全状态评估。它的工作原理是自动聚合环境的配置数据，并应用规则对环境进行审计。我们已经在多个项目中发现它对即时安全评估非常有用。

视觉回归测试工具
试验

自我们在 2014 年第一次提及视觉回归测试工具以来，这项技术已经得到了广泛的应用，工具的前景也得到了发展。BackstopJS 仍然是一个很好的选择，它会定期添加新特性，包括支持在 Docker 容器中运行。在上期技术雷达中我们精选出了 Loki。SaaS 解决方案则有 Applitools、CrossBrowserTesting 和 Percy。另一个值得一提的工具是 Resemble.js，一个图像比较库。尽管许多团队只是间接地把它作为 BackstopJS 的一部分来使用，我们的一些团队已经直接使用它来分析比较网页中的图片了。总之，我们的经验表明，在界面发生重大变化的早期阶段，视觉回归工具的用处不大，但是随着产品的成熟和界面的稳定，它们肯定会证明自己的价值。

工具

OPA 提供了一套统一的框架和语言，用于声明，实施和控制云原生解决方案中各个组件的策略。

（Open Policy Agent（OPA））

我们的团队发现 Optimal Workshop 非常实用，因为它让验证产品原型和配置数据收集的相关测试变得很简单，因此也能帮助我们做出更好的决策。

（Optimal Workshop）

工具

AsyncAPI 是一项开源计划，旨在构建急需的事件驱动和异步 API 标准以及开发工具。

(AsyncAPI)

ConfigCat 支持简单的特性开关、用户细分和 A/B 测试，它还对少量使用或起步者慷慨地提供了免费服务。

(ConfigCat)

Visual Studio Live Share
试验

Visual Studio Live Share 是用于 Visual Studio Code 及 Visual Studio 的扩展套件。如果你的团队正在寻找一种良好的远程协作方式，可以考虑这款出色的工具。Live Share 提供了良好的、低延迟的远程结对体验，并且所需的带宽比粗暴地共享整个桌面要少得多。更重要的是，开发人员可以在结对过程中使用他们各自的配置、扩展和快捷键。Live Share 不仅支持编辑、调试代码的实时协作，还支持语音呼叫，以及共享终端和服务器。

Apache Superset
评估

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。它可以与 Presto、Amazon Athena 和 Amazon Redshift 协同工作，并且可以很好地与企业身份验证集成。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。需要指出的是，Apache Superset 目前正在 Apache 软件基金会（ASF，Apache Software Foundation）进行孵化，这意味着它还没有得到 ASF 的完全认可。

AsyncAPI
评估

开放标准是构建分布式系统的基础支柱之一。例如，OpenAPI（以前被称为 Swagger）规范作为定义 RESTful API 的行业标准，对微服务等分布式架构的成功至关重要。基于它涌现出了大量用于构建、测试和监控 RESTful API 的工具。然而在使用事件驱动 API 的分布式系统中，这种标准化在很大程度上是缺失的。

AsyncAPI 是一项开源计划，旨在构建急需的事件驱动和异步 API 标准以及开发工具。AsyncAPI 规范受 OpenAPI 规范的启发，以一种机器可读的格式描述和记录了事件驱动 API。它与协议无关，因此可以适用于包括 MQTT，WebSocket 和 Kafka 在内的许多协议的 API 上。我们期待看到 AsyncAPI 的持续改进以及其工具生态系统的进一步成熟。

ConfigCat
评估

如果你正在寻找一个支持动态特性开关的服务（切记简单的特性开关也是可行的），那么请看一下 CongfigCat。我们对它的描述是“类似于 LaunchDarkly，但更便宜，且少了些花哨”。我们认为它能满足我们的绝大部分需求。ConfigCat 支持简单的特性开关、用户细分和 A/B 测试，它还对少量使用或起步者慷慨地提供了免费服务。

Gitpod
评估

大多数软件可以通过简单的两步进行构建：签出代码库，然后运行一个构建脚本。不过，建立一个完整编码环境的过程仍然很繁琐。Gitpod 通过为 Github 或 GitLab 仓库提供基于云的、现成的代码环境来解决这个问题。它提供了一个基于 Visual Studio 代码的 IDE，可以在 web 浏览器中运行。默认情况下，这些环境是在谷歌云平台上启动的，当然也可以部署内部解决方案。它的价值是显而易见的，特别是对于开源软件，这种方法可以降低临时贡献者的门槛。然而，这种方法在企业环境中的可行性还有待观察。

Gloo
评估

随着 Kubernetes 和 service mesh 的日益普及，API 网关在云原生分布式系统中一直面临着生存危机。毕竟，许多 API 网关的功能，如流量控制，安全性，路由和可观察性等，现在都是由集群的入口控制器和网格网关提供。Gloo 是一个支持这种变化的轻量级 API 网关，它使用 Envoy 作为其网关技术，同时为外部用户和应用程序提供附加价值，如 API 的内聚视图等。Gloo 还提供了一套管理界面用于控制 Envoy 网关，并运行和集成了多个服务网格的实现，如 Linkerd，Istio 和 AWS App Mesh。尽管它的开源版本已经提供了 API 网关的基本功能，但它的企业版有一组更成熟的安全控件，如 API 密钥管理，OPA 集成等。Gloo 是一个很有前途的轻量级 API 网关。它很好地适应了云原生技术和架构的生态系统，同时避免了在 API 网关中引入业务逻辑以迎合最终用户的陷阱。

Lens
评估

Kubernetes 的一个优势是灵活性和可配置的范围，以及 API 驱动的、可编程的配置机制和使用 manifest 文件的命令行可见性与控制。然而，福祸相依，当部署非常复杂或者需要管理多个集群时，单纯通过命令行属性和 manifest 文件很难清楚地了解总体状态。Lens 试图通过一个集成环境来解决这个问题，这个集成环境可以用以查看集群的当前状态和工作负载，可视化集群指标，并通过内嵌的文本编辑器修改配置。与简单的点击界面不同，Lens 把管理员可能在命令行里运行的工具整合到了一个可导航的界面中。这个工具是试图驯服 Kubernetes 管理复杂性的几种方法之一。在这个领域，我们还没有见到一个明确的胜利者，不过 Lens 在用户图形界面和纯命令行工具之间找到了一个有趣的平衡。

Manifold
评估

Manifold 是机器学习的一个与模型无关的可视化调试工具。模型开发人员通常会花费大量的时间用于迭代和改进现有模型，而不是创建一个新的模型。通过将焦点从模型转移到数据，Manifold 对影响模型性能的数据集特征进行可视化，并以此补充现有的性能指标。我们认为 Manifold 将会是机器学习生态系统中一个值得考虑的实用工具。

Sizzy
评估

构建一个在多种设备和屏幕尺寸上看起来都符合预期的网页应用并不是一件简单的事情。Sizzy 是一个 SaaS 解决方案，用于在一个浏览器窗口内展示多个视窗。应用会被同时渲染到所有的视窗中，并且对应用的交互也会同步到所有视窗中。根据我们的经验，以这种方式与应用交互，可以更容易地在视觉回归测试工具于构建流水线中标记问题之前发现潜在的问题。尽管如此，我们也应该指出，我们的一些开发人员在尝试使用了一段时间 Sizzy 之后，总的来说，还是更倾向于使用 Chrome 提供的工具。

Snowpack
评估

Snowpack 是 JavaScript 构建工具领域中的一个有趣的新成员。与其他解决方案相比，Snowpack 的关键的改进是可以使用 React.js，Vue.js 和 Angular 等现代框架来构建应用程序，而无需打包器。由于省去了打包的环节，对代码的任何修改都几乎可以立即显示在浏览器上，因此开发过程中的反馈周期得到了极大的改善。为了达到这个神奇的效果，Snowpack 将 `node_modules` 中的依赖转换为单个的 JavaScript 文件，并将其放置于一个新的 `web_modules` 目录中，从这个目录中可以将它们作为 ECMAScript 模块（ESM）导入。对于 IE11 和其他不支持 ESM 的浏览器，它也支持一种变通方法。遗憾的是，目前还没有任何浏览器可以从 JavaScript 中导入 CSS，因此使用 CSS 模块并不简单。

tfsec
评估

安全是所有人都关心的问题，尽早捕获风险总好过于后面再去面对由此导致的各种问题。在基础设施即代码领域，Terraform 是管理云环境时一个显而易见的选择。现在我们又有了 tfsec。它是一个静态分析工具，可以用来扫描 Terraform 模板并查找潜在的安全问题。它针对不同的云提供商（包括 AWS 和 Azure）都预设了安全规则。我们一直都很喜欢那些有助于降低安全风险的工具，而 tfsec 不仅擅长识别安全风险，安装和使用也非常简单。

工具

Lens 是试图驯服 Kubernetes 管理复杂性的几种方法之一。

（Lens）

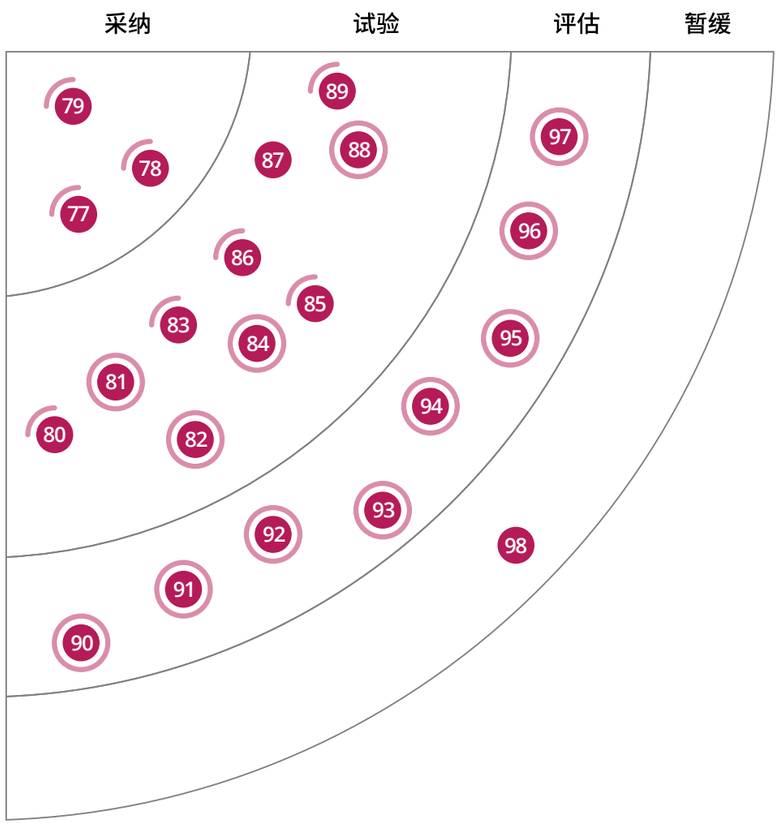
tfsec 是一个静态分析工具，可以用来扫描 Terraform 模板并查找潜在的安全问题。

（tfsec）

TECHNOLOGY RADAR Vol. 22

语言&框架

语言&框架



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。基于我们的经验，Hooks 提高了组件之间功能的重用性和代码的可读性。考虑到 Hooks 使用 React Test Renderer 和 React Testing Library 改进了可测试性，以及不断增长的社区支持，我们将其作为我们的首选方法。

React Testing Library
采纳

JavaScript 的世界日新月异，随着我们使用框架的经验越来越多，我们的倾向也在改变。我们深入使用某些框架，其他备选框架自然黯然失色。在 React 前端测试方面，React Testing Library 就是这样一个例子。我们团队很喜欢的是，用这个框架写的测试比其他框架（如 Enzyme）更健壮，因为它鼓励独立测试组件间的关系，而不是测试全部实现细节。这种思维源自于测试库，React Testing Library 是它的一部分，它还为像 Angular 和 Vue.js 这样的框架提供了一整套库。

Vue.js
采纳

Vue.js 在我们的社区中已经成为一个值得信赖且备受喜爱的成功的前端框架。虽然还有其他被广泛采用的同类型产品，比如 React.js，但简明的 API 设计、职责清晰的指令及模块（一个模块对应一个文件的设定）和更简单的状态管理，都让 Vue.js 成为有力的竞争者。

CSS-in-JS
试验

自 2017 年，我们把 CSS-in-JS 作为新兴技术提出后，它变得越来越流行了，在我们的工作中也可以看到明显的趋势。基于一些扎实的

采纳
77. React Hooks
78. React Testing Library
79. Vue.js

试验
80. CSS-in-JS
81. Exposed
82. GraphQL Inspector
83. Karate
84. Koin
85. NestJS
86. PyTorch
87. Rust
88. Sarama
89. SwiftUI

评估
90. Clinic.js Bubbleprof
91. Deequ
92. ERNIE
93. MediaPipe
94. Tailwind CSS
95. Tamer
96. Wire
97. XState

暂缓
98. Enzyme

语言&框架

Koin 是一个 Kotlin 框架，用于处理软件开发中的常规问题之一：依赖注入。

(Koin)

NestJS 是一个让 Node.js 应用开发更安全、更不易出错的基于 TypeScript 优先风格的框架。

(NestJS)

生产经验，我们现在推荐 CSS-in-JS 为试用技术。在上次的技术雷达中我们提到过，要了解这个技术，样式化组件框架是一个很好的出发点。在使用 CSS-in-JS 技术的时候，除了它的种种优点，通常也会遭遇到一个缺点：运行时计算样式会导致用户可感知延迟。通过 Linaria，我们看到了一类新的在创建时就考虑到这个问题的框架。Linaria 引入了许多技术把大部分性能开销转移到构建时间。不过这也带来了一些妥协，最常见的就是不支持 IE11 的动态样式。

Exposed
试验

在长期使用 Kotlin 的过程中，我们的开发团队没有将 Java 框架与 Kotlin 混用，从而获得了更多针对 Kotlin 设计的框架的经验。作为轻量级的对象关系映射器（ORM），Exposed 过了好一阵子才引起我们的注意。Exposed 有两种数据库访问方式：类型安全的内部 DSL 包装 SQL，以及数据访问对象（DAO）模式的实现。它具有一个成熟的 ORM 所能被期待的功能，例如对多对多引用的处理，急切加载（eager loading）以及对跨实体联接的支持。Exposed 受到我们团队青睐的地方还在于，方法实现无需代理，并且不依赖于反射，这无疑对性能有所帮助。

GraphQL Inspector
试验

GraphQL Inspector 可以比较两个 GraphQL 模式（schema）之间的变更。我们曾经警告要谨慎使用 GraphQL。自那以后，我们很高兴看到 GraphQL 相关的工具有了一些改进。ThoughtWorks 的大多数团队会继续将 GraphQL 用于服务器端资源聚合。通过将 GraphQL Inspector 集成到团队的 CI 流水线中，我们已经能够捕获 GraphQL 模式中潜在的重大变化。

Karate
试验

基于认为测试是唯一真正重要的 API 规范的经验，我们一直在寻找对测试有帮助的新工具。Karate 是一种 API 测试框架，其独特之处在于它不依赖通用编程语言，而直接使用基于 Gherkin 的语法编写测试。Karate 使用一种领域特定语言，来描述基于 HTTP 的 API 测试。我们的团队喜欢 Karate 为 API 规范带来的易读性，并建议将其应用于测试金字塔的较高层次，而非过量应用在细节的断言中。

Koin
试验

随着 Kotlin 被越来越多地用于移动和服务端开发，其相关生态系统也在不断发展。Koin 是一个 Kotlin 框架，用于处理软件开发中的常规问题之一：依赖注入。尽管有多种 Kotlin 依赖注入框架可供选择，我们的团队更喜欢 Koin 的简单性。Koin 避免使用注解，而是通过构造函数或模仿 Kotlin 的延迟初始化，从而仅在需要时才注入对象。这与 Android 基于静态编译的 Dagger 注入框架形成鲜明对比。我们的开发人员喜欢此框架的轻量级本质及其内置的可测试性。

NestJS
试验

随着 Node.js 越来越流行，诸如 Node 滥用等趋势逐渐显现，这导致了许多开发者使用 Node.js 开发业务系统。我们经常看到大型 JavaScript 系统在可伸缩性、可维护性方面出现各种问题。NestJS 是一个让 Node.js 应用开发更安全、更不易出错的基于 TypeScript 优先风格的框架。NestJS 是一个有态度的框架。它的架构直接受 Angular 启发，符合 SOLID 原则。我们的团队常用 NestJS 构建 Node.js 微服务，它给开发者赋能，能让他们编写出可测试、可伸缩、低耦合、易维护的应用。

PyTorch
试验

我们的团队一直在使用并且很认可 PyTorch 机器学习框架，并且有几支团队对 PyTorch 的喜爱甚于 TensorFlow。PyTorch 暴露了 TensorFlow 隐藏的 ML 内部工作原理，使其更易于调试，并包含了程序员熟悉的结构，例如循环和动作。PyTorch 最新版本提高了性能，我们已在生产项目中成功使用了它。

Rust
试验

Rust 日益受到欢迎。曾经，Rust、C++ 和 Go 哪门语言更好的讨论火了一段时间，谁是赢家却尚未明确。近期，令人感到高兴的是，Rust 改善显著，添加并稳固了更多内置 API，包括上一期技术雷达中提到的高级异步支持。此外，Rust 还启发了新语言的设计。例如，Libra 区块链上的 Move 语言借鉴了 Rust 的内存管理方式来管理资源，从而确保了数字资产永远不会被复制或隐式丢弃。

Sarama
试验

Sarama 是 Apache Kafka 的 Go 客户端库。如果你在 Go 中开发 API，你会发现 Sarama 非常容易设置和管理，因为它不依赖于任何原生库。Sarama 有两种类型的 API——一种高层 API 用于轻松地生产和消费消息，另一种底层 API 用于控制网络上的字节。

SwiftUI
试验

苹果已在其新的 SwiftUI 框架上迈出了一大步，该框架用于在 macOS 和 iOS 平台上实现用户界面。我们很高兴 SwiftUI 跨越了 Interface Builder 和 XCode 之间略显混乱的关系，并采用了一致的、声明性的以及以代码为中心的方式。现在，你可以在 XCode 11 中并排查看代码和生成的可视化界面，从而获得更好的开发人员体验。SwiftUI 框架还从近年来主导Web 开发的 React.js 世界汲取了灵感，它利用视图模型的不可变值和异步更新机制，构成了统一的反应式编程模型。这为开发人员提供了一个完全原生的替代品，以替代类似 React Native 或 Flutter 这样的反应式框架。SwiftUI 无疑代表了 Apple UI 开发的未来，尽管很新，但它已经显示出优势。它和它平缓的学习曲线给予我们非常棒的体验。值得注意的是，鉴于它不支持 iOS 12 以及更低版本，因此在使用 SwiftUI 之前，你应该首先了解客户的使用场景。

Clinic.js Bubbleprof
评估

在提升代码性能的过程中，定位代码的瓶颈和延迟点通常比较困难，特别是在异步操作中。性能剖析工具此时就十分重要。Clinic.js Bubbleprof 会可视化 Node.js 进程中的异步操作，绘制程序调用流中的延迟图。我们很喜爱这类工具，因为它帮助开发人员轻松的定位和确定代码改进的优先级。

Deequ
评估

在数据工程中使用良好的软件工程实践，也仍然存在一些工具空白。我们的一个团队尝试在数据管道中的不同步骤之间自动执行数据质量检查时，惊讶地发现，在该领域中只有很少的工具。他们选择了 Deequ，这是一个用来为数据集编写类似单元测试的库。Deequ 建立在 Apache Spark 之上，虽然它是由 AWS 实验室发布的，但也可以被用在 AWS 以外的环境。

ERNIE
评估

在上一期技术雷达中，我们加入了 BERT——NLP（Natural Language Processing，自然语言处理）领域中的一个关键里程碑。去年，百度发布了 ERNIE 2.0（Enhanced Representation through kNowledge IntEgration），它在 7 个 GLUE（General Language Understanding Evaluation）语言理解任务和全部 9 个中文 NLP 任务上的表现均优于 BERT。和 BERT 一样，ERNIE 也提供了无监督预训练语言模型。它可以通过添加输出层的方法来进行微调，以创建多种 NLP 任务的当前最优模型。ERNIE 与传统预训练方法不同之处在于，它是一个连续的预训练框架。它可以不断地引入各种各样的预训练任务，以帮助模型有效地学习语言表达，而不是仅使用少量的预训练目标进行训练。我们对 NLP 的进步感到非常兴奋，并期待在我们的项目中尝试。

MediaPipe
评估

MediaPipe 是一个用于构建多模态（例如视频，音频，时间序列数据等），跨平台（例如 Android，iOS，Web 和边界设备）的应用类机

语言&框架

Deequ 是一个用来为数据集编写类似单元测试的库，可以在数据管道中的不同步骤之间自动执行数据质量检查。

（Deequ）

ERNIE 提供了无监督预训练语言模型。它可以通过添加输出层的方法来进行微调，以创建多种 NLP 任务的当前最优模型。

（ERNIE）

语言&框架

为了便于定制，它仅提供了较低层次的 CSS 样式类来构建模块，且没有自带任何多余的复杂样式。

(Tailwind CSS)

Wire 是一种编译时依赖注入工具，可以同时生成代码并将组件连接在一起。

(Wire)

器学习流水线。它提供包括面部识别，手部识别，手势识别以及物体识别在内的多种能力。尽管 MediaPipe 主要部署在移动设备上，但多亏了 WebAssembly 和 XNNPack 机器学习推理类库的帮助，使得它也能在浏览器上运行。就像目前所看到的一样，我们正在探索一些将 MediaPipe 用于增强现实的用例。

Tailwind CSS

评估

CSS 工具和框架提供了预先设计的组件，帮助开发者快速实现想要的页面效果。但是随着开发的进行，它们变得难以定制。Tailwind CSS 提供了一种有趣的方法。为了便于定制，它仅提供了较低层次的 CSS 样式类来构建模块，且没有自带任何多余的复杂样式。较低层次 CSS 样式类广泛的覆盖，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。这样来看 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。

Tamer

评估

如果需要将关系数据库中的数据收集到 Kafka 的 Topic 中，可以考虑使用 Tamer，它将自己标榜为“驯化的 Kafka JDBC数据源连接器”。尽管 Tamer 是一个相对较新的框架，但它比 Kafka JDBC 连接器更高效，尤其是在处理大量数据时。

Wire

评估

Golang 社区中有相当一部分人是依赖注入怀疑论者，部分原因是他们将模式与特定框架相混淆，并且具有系统编程背景的开发人员本能地不喜欢反射引起的运行时开销。Wire 应运而生，这是一种编译时依赖注入工具，可以同时生成代码并将组件连接在一起。Wire 没有额外的运行时开销，并且更易于推断静态依赖关系图。无论你手写代码还是使用框架，我们都建议使用依赖注入来鼓励模块化和可测试的设计。

XState

评估

我们在之前的技术雷达中收录了几个状态管理库，但是 XState 采用的方法略有不同。这是一个简单的 JavaScript 和 TypeScript 框架，用于创建有限状态机并将其可视化为状态图。它与一些流行的响应式 JavaScript 框架 (Vue.js, Ember.js, React.js 和 RxJS) 集成，并且是基于 W3C 标准的有限状态机。另一个值得注意的功能是状态机定义的序列化。我们发现有一点很有用，那就是在其他上下文中创建有限状态机 (特别是在编写游戏逻辑) 时，可视化状态及其可能的转换的能力。我们确实很喜欢 Xstate 的 visualizer 可以很容易做到这一点。

Enzyme

暂缓

我们通常不会将已经移除的工具保留在技术雷达上，但是我们的团队强烈感受到 Enzyme 应该替换为 React Testing Library 来用于测试 React 界面组件。使用 Enzyme 的团队发现它对于被测试组件内部的聚焦会导致脆弱的、无法维护的测试。

ThoughtWorks®

ThoughtWorks 是一家软件咨询公司，也是一个充满热情、以目标为导向的社区。我们帮助客户以技术为核心，推动其商业变革，与他们并肩作战解决最核心的技术问题。我们致力于积极变革，希望能够通过软件技术创造更美好的社会，与此同时我们也与许多志向相投的组织合作。

创办 25 年以来，ThoughtWorks 已经从一个小团队，成长为现在拥有超过 7000 人，分布于全球 14 个国家、拥有 43 间办公室的全球企业。这 14 个国家是：澳大利亚、巴西、加拿大、智利、中国、厄瓜多尔、德国、印度、意大利、新加坡、西班牙、泰国、英国、美国。

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 洞见

ThoughtWorks®

thoughtworks.com/radar

#TWTechRadar