ThoughtWorks®

TECHNOLOGY RADAR VOL. 20

有态度的前沿技术解析

thoughtworks.com/radar
#TWTechRadar

贡献者

技术雷达由ThoughtWorks技术顾问委员会筹备

本期技术雷达是基于ThoughtWorks技术顾问委员会
在2019年3月深圳会议上的讨论所得出的。


Rebecca Parsons (CTO)


Martin Fowler (Chief Scientist)


Bharani Subramaniam


Erik Dörnenburg


Evan Bottcher


Fausto de la Torre


Hao Xu


Ian Cartwright


James Lewis


Jonny LeRoy


Ketan Padegaonkar


Lakshminarasimhan Sudarshan


Marco Valtas


Mike Mason


Neal Ford


Ni Wang


Rachel Laycock


Scott Shaw


Shangqi Liu


Zhamak Dehghani

技术雷达中国区技术咨询顾问组：

边晓琳	樊卓文	韩盼盼	胡晨	蒋帆	梁凌锐	刘先宁	马伟	闵锐
阮焕	万学凡	王柏元	王亦凡	魏喆	吴邦	伍斌	向博	徐培
杨琛	姚琪琳	姚小玉	易维利	张凯峰	张霄翀	张扬	赵琪琪	

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

1 采纳

我们强烈主张业界采纳这些技术。我们会在适当时候将其用于我们的项目。

2 试验

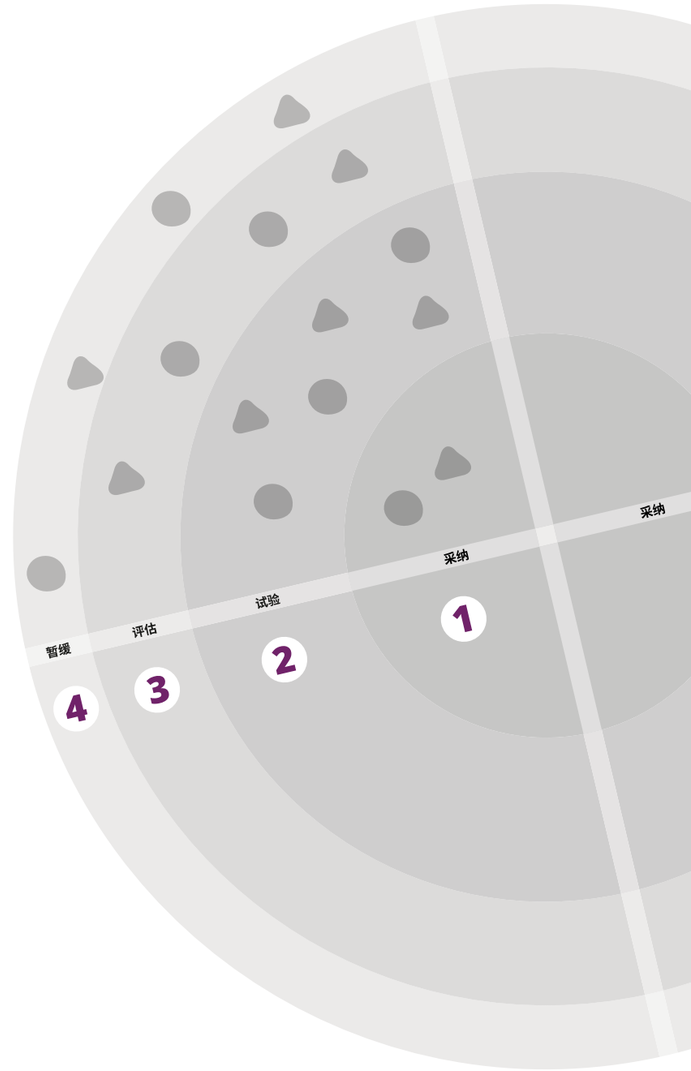
值得追求。重要的是理解如何建立这种能力。企业应该在风险可控的项目中尝试此技术。

3 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。

4 暂缓

谨慎推行。



新的或发生变化的
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。

最新动态

本期精彩集锦

日新月异的数据形态

十年前，数据几乎等同于关系数据库。如今，数据则可能呈现出各种形态，包括NoSQL、时间序列、像CockRoachDB和Spanner这样提供全局一致性的SQL存储，以及提供聚合日志文件查询功能的事件流。随着数据源不断增长，数据规模越来越大，种类越来越多，变化速度越来越快，企业想要对这些数据做出更实时的响应，上述情况也就应运而生了。对于开发人员，每种形式的数据使用方法都存在固有的优缺点，如何取舍是个难题。架构师和开发人员应该密切关注各种工具和模型提供的新功能，同时保持勤奋好学，不要完全以对待现有工具的常用方法来使用新工具。我们必须认识到数据形势正在发生重大变革，并坚持寻找合适的策略和工具。

Terraform生态系统建设

开发人员喜欢抽象层，原因很明显，因为他们可以将复杂问题封装到抽象层中，从而集中精力处理更高层级的问题。在前几期的技术雷达中，我们都阐述了这种发展趋势，很多团队在同时使用云和容器时都会采纳这种方法。一开始他们关注的是Docker及其生态系统。然后焦点开始转向Kubernetes。现在，我们发现主要活动总体上都集中在基础设施即代码方面，尤其是集中在Terraform生态系统上。虽然除了Terraform，我们还推荐了其他工具，但是它在提供商社区中的采纳率仍然让人印象深刻。本期雷达的内容重点包括Terratest（用于测试基础设施代码），以及GoCD的新提供商（可以使用Terraform配置GoCD）。

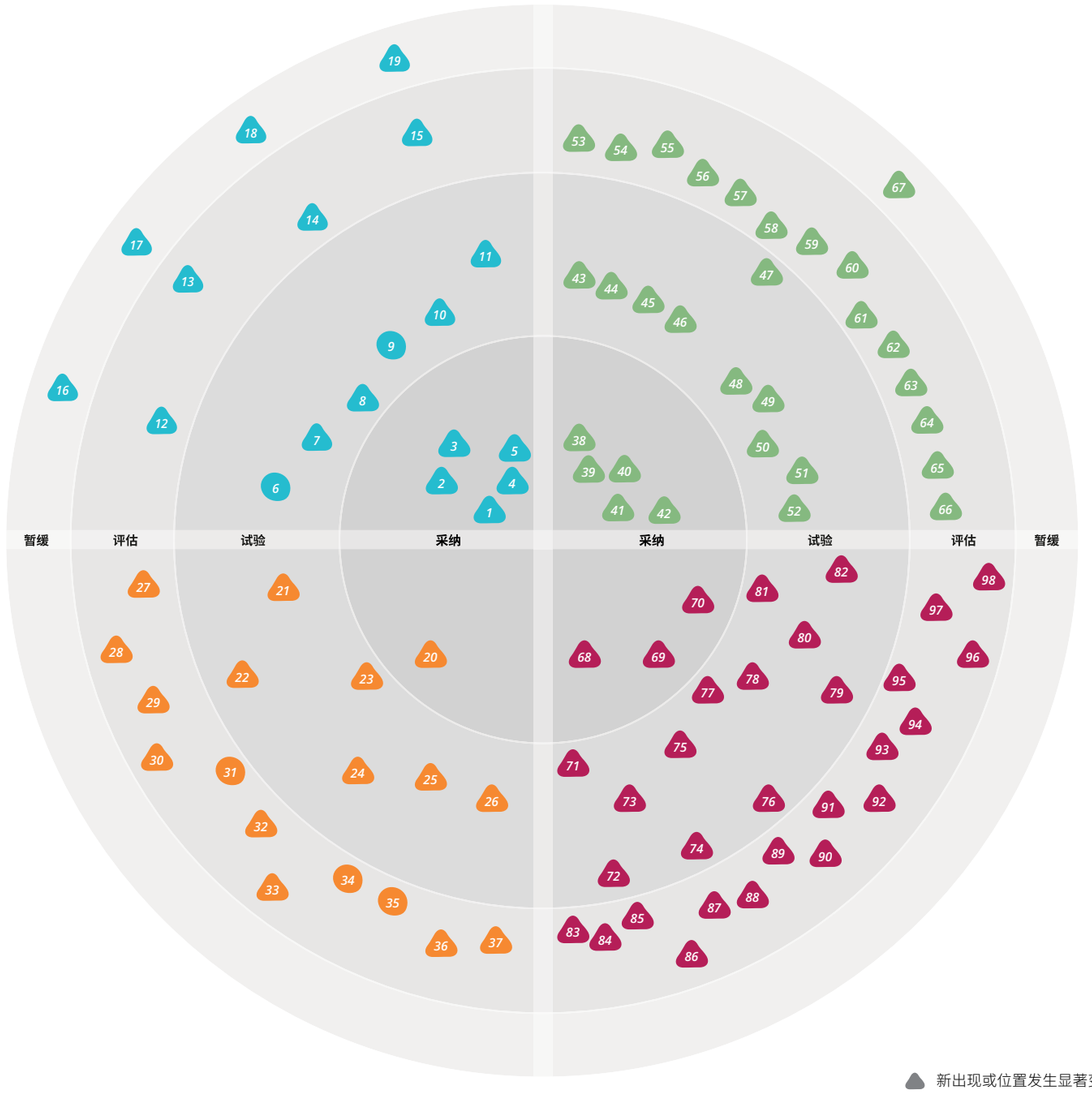
Kotlin方兴未艾

Kotlin作为一项开源语言，不仅在Android环境中获得了广泛应用，而且还在向其他环境拓展，也在技术雷达中受到了持续密切的关注。由于不喜欢现有的语言选择，JetBrains在内部开发了Kotlin，并随后开源。Kotlin似乎在各种开发人员中广受欢迎。它经常在各种平台和工具中用作通用甚至专用开发语言，而且在技术雷达中出现的频率也越来越高。此外，我们的项目团队也在采纳该语言（Ktor、MockK、Detekt、HTTP4K）。这个新兴语言在实用性设计和先进工具中获得了广泛应用，并且形成了一个蓬勃发展的生态系统，取得了巨大的成功，对此我们深感欣慰。

封装边界的泄漏

随着“一切即代码”理念的盛行，以前难以改变的绝大多数环节（基础设施、安全、合规性和运营），几乎都变得可以通过编程来处理，这就意味着开发人员可以采纳更完善的工程实践。然而，我们仍然经常看到，要么配置子系统异常复杂，要么过度依赖于可视化编排工具，逻辑渗透到配置文件中，以YAML编写的条件语法晦涩复杂，还有各种技术需要使用大量编排框架等情况。随着多语言编程、基础设施即代码和一切皆服务技术的出现，我们不再需要将各种组件都组合到单一内聚的系统中。因此，原本应位于系统边界内的逻辑就会泄漏到编排工具、配置文件和其他管道中。虽然有时候确有这种必要，但是我们建议各团队保持谨慎，考虑将此类代码放在开发人员执行测试、版本控制、持续集成和其他完善的工程实践的位置。请避免将业务逻辑放在配置文件中（并且避免使用要求将业务逻辑放在配置文件中的工具），尽可能减少必须执行的编排操作，不要让编排功能主导你的系统。

THE RADAR



新出现或位置发生显著变化的条目
没有变化的条目

技术

采纳
1. Four key metrics
2. Micro frontends
3. Opinionated and automated code formatting
4. Polyglot programming
5. Secrets as a service

试验
6. Chaos Engineering
7. Container security scanning
8. Continuous delivery for machine learning (CD4ML) models
9. Crypto shredding
10. Infrastructure configuration scanner
11. Service mesh

评估
12. Ethical OS
13. Smart contracts
14. Transfer learning for NLP
15. Wardley mapping

暂缓
16. Productionizing Jupyter Notebooks
17. Puncturing encapsulation with change data capture
18. Release train
19. Templating in YAML

平台

采纳
20. Contentful

试验
21. AWS Fargate
22. EVM beyond Ethereum
23. InfluxDB
24. Istio
25. Kafka Streams
26. Nomad

评估
27. CloudEvents
28. Cloudflare Workers
29. Deno
30. Hot Chocolate
31. Knative
32. MinIO
33. Prophet
34. Quorum
35. SPIFFE
36. Tendermint
37. TimescaleDB

暂缓

工具

采纳
38. Cypress
39. Jupyter
40. LocalStack
41. Terraform
42. UI dev environments

试验
43. AnyStatus
44. AVA
45. batect
46. Elasticsearch LTR
47. Helm
48. InSpec
49. Lottie
50. Stolon
51. TestCafe
52. Traefik

评估
53. Anka
54. Cage
55. Cilium
56. Detekt
57. Flagr
58. Gremlin
59. Honeycomb
60. Humio
61. Kubernetes Operators
62. OpenAPM
63. Systems
64. Taurus
65. Terraform provider GoCD
66. Terratest

暂缓
67. Handwritten CloudFormation

语言&框架

采纳
68. Apollo
69. MockK
70. TypeScript

试验
71. Apache Beam
72. Formik
73. HiveRunner
74. joi
75. Ktor
76. Laconia
77. Puppeteer
78. Reactor
79. Resilience4j
80. Room
81. Rust
82. WebFlux

评估
83. Aeron
84. Arrow
85. Chaos Toolkit
86. Dask
87. Embark
88. fastai
89. http4k
90. Immer
91. Karate
92. Micronaut
93. Next.js
94. Pose
95. react-testing-library
96. ReasonML
97. Taiko
98. Vapor

暂缓

技术

Four key metrics
采纳

详尽的DevOps现状报告侧重于对高绩效组织的数据驱动型统计分析。这项研究持续多年，结果发表在Accelerate，证明了组织绩效和软件交付效能之间存在直接关联。研究人员证实，只需四个关键指标就能区分低绩效、中绩效和高绩效人员：前置时间、部署频率、平均修复时间(MTTR)和变更失败率。的确，我们已经发现这四个关键指标是个简单却强大的工具，可帮助领导和团队专注于衡量并改进重要的环节。实施构建流水线是一个良好开端，以便你能够捕获四个关键指标（Four key metrics），并使软件交付价值流可见。例如，作为GoCD Analytics的一等公民，GoCD流水线能够衡量这四个关键指标。

Micro frontends
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和维护的服务的交付。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。自从第一次将微前端（Micro frontends）描述为能够解决这一问题的技术以来，我们在微前端领域积累了近乎普遍的积极经验，并且随着越来越多的代码从服务器转移到浏览器，我们还发现了微前端的一些使用模式。但到目前为止，Web组件在这个领域仍然令人难以捉摸。

Opinionated and automated code formatting
采纳

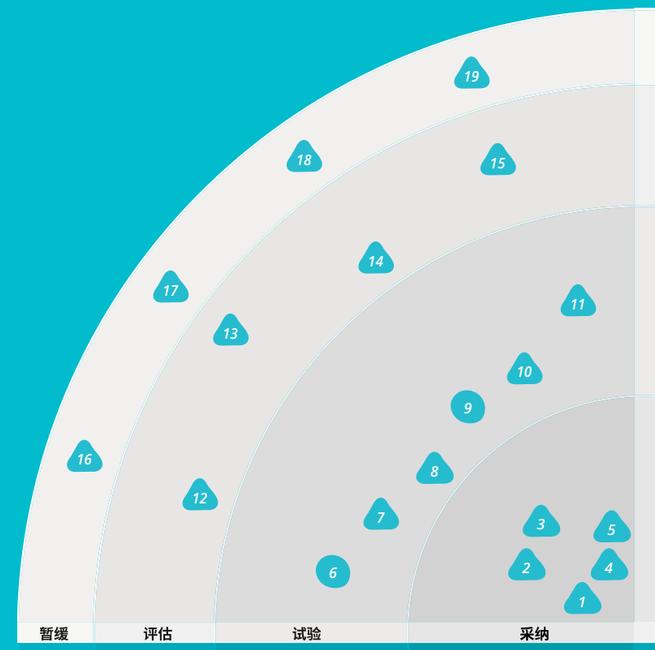
在我们的记忆里，对代码进行格式化所用的风格取决于个人品味、公司政策和激烈的辩论。最终，行业似乎厌倦了这无休止的争论，停止这些讨论、简单地采纳有态度的代码格式化工具（Opinionated and automated code formatting），能够为团队节省大量时间。即使你不完全同意某个工具的代码风格，但大多数团队都应该能够理解专注于代码用途（而非其外观）所带来的好处。对于JavaScript，Prettier一直在争取我们的一票支持，类似的还有Python的Black，而且越来越多的语言开始内置这样的工具，比如：Golang，Elixir。此处的关键在于不要花费数小时来讨论要强制执行哪些规则，而是选择一个有态度的、最低可配置的自动工具，最好是把这个自动化过程配置到pre-commit hook中。

采纳
1. Four key metrics
2. Micro frontends
3. Opinionated and automated code formatting
4. Polyglot programming
5. Secrets as a service

试验
6. Chaos Engineering
7. Container security scanning
8. Continuous delivery for machine learning (CD4ML) models
9. Crypto shredding
10. Infrastructure configuration scanner
11. Service mesh

评估
12. Ethical OS
13. Smart contracts
14. Transfer learning for NLP
15. Wardley mapping

暂缓
16. Productionizing Jupyter Notebooks
17. Puncturing encapsulation with change data capture
18. Release train
19. Templating in YAML



技术

在工作中选择正确的语言可以大幅提高生产效率，推广几种支持不同生态系统或特性的语言能够帮助企业及开发人员更好地解决问题。

(Polyglot programming)

机器学习的持续交付 (CD4ML) 模型，将持续交付实践应用于开发机器学习模型上，以便于随时应用在生产环境中。

(Continuous delivery for machine learning (CD4ML) models)

Polyglot programming
采纳

我们将多语言编程（Polyglot Programming）归入最早一期技术雷达中的“试验”部分，是为了表明为工作选择正确的语言可以大幅提高生产效率，并且出现了值得考虑的新语言。我们想要重申这一点，因为我们最近看到了开发人员和企业在共同发起的标准化语言技术栈方面的努力。虽然我们承认，如果毫无约束的使用语言，所产生的问题要比解决的问题更多，但如果企业要加快流程和上线速度，以及开发人员需要合适的工具来解决手头的问题，推广几种支持不同生态系统或特性的语言就显得非常重要。

Secrets as a service
采纳

在构建和运维软件的价值流中，人和机器都会使用密码凭据。构建流水线需要使用密码凭据来与容器注册中心等安全基础设施进行交互，应用程序需要使用API密钥作为密码凭据来获得业务功能访问权限，而服务间通信则需要以证书和密钥作为密码凭据来保护其安全。这些密码凭据可以使用不同的方式进行设置和检索。我们经常提醒开发人员注意不要使用源代码管理方法来存储密码凭据。我们建议将密码凭据与源代码分开管理，并且建议使用git-secrets和Talisman等工具，以免将密码凭据凭据存储在源代码中。我们习惯于使用密码凭据即服务（Secrets as a service）作为存储和访问密码凭据的默认技术。利用这种技术，你可以使用Vault或AWS Key Management Service（KMS）等工具来读写HTTPS端点上的密码凭据，同时实现精细的访问控制。密码凭据即服务使用外部“身份提供方”（例如AWS IAM）来确定请求访问密码凭据的“行为方”的身份。“行为方”利用“密码凭据即服务”来认证自己的身份。要使这种流程有效运行，就必须使“行为方”、“密码凭据即服务”和应用程序的身份引导过程实现自动化。很多基于SPIFFE的平台已经提高了向服务分发身份的自动化水平。

Chaos Engineering
试验

在去年，我们看到混沌工程（Chaos Engineering）从一个备受关注的想法，转变成公认的主流方法，来改善并保证分布式系统的弹性。随着大大小小的组织开始将混沌工程作为运营流程实施，我们正在学习如何安全地大规模应用这些技术。该方法当然并非适合所有人，为确保高效和安全，需要大规模的组织支持。随着Gremlin等商业服务以及Spinnaker这样实现了一些混沌工程理念的部署工具的出现，行业认可度和可用的专有技能也将得到明显的提升。

Container security scanning
试验

围绕Docker的容器革命显著减少了应用在跨环境迁移时的阻力，并推动持续交付和持续部署的采纳。但尤其是后者，对于传统的投产控制带来了相当大的漏洞。容器安全扫描（Container security scanning）技术是对该威胁载体的必要响应。构建流水线中的工具，会自动检查流水线中的容器是否存在已知漏洞。自从我们第一次提及该技术以来，工具全景已成熟，并且该技术已被证实对我们客户的开发工作也非常有用。

Continuous delivery for machine learning (CD4ML) models
试验

机器学习的持续交付（CD4ML）模型（Continuous delivery for machine learning -CD4ML models），将持续交付实践应用于开发机器学习模型上，以便于随时应用在生产环境中。该方法解决了传统机器学习模型开发的两个主要问题：一个是训练模型和将模型部署到生产环境之间的周期过长，此过程通常包括将模型手动转换为可上线的代码；另一个问题是使用被过时数据训练过的产品环境模型。

机器学习模型的持续交付流水线有两个触发因素：(1) 模型结构的变动；(2) 训练与测试数据集的变动。要使其发挥作用，我们需要对数据集和模型源代码进行版本化。流水线通常包括用测试数据集来测试模型、按需使用H2O等工具来对模型作自动转换、以及将模型部署到生产环境以交付价值。

Crypto shredding
试验

对敏感数据保持适当的控制是相当困难的，尤其是出于备份及恢复的目的，将敏感数据复制到主数据系统外的时候。密钥销毁是指主动覆

盖或删除用于保护敏感数据的加密密钥，以保护敏感数据不被读取。对于审计应用程序或区块链这样不应该或不能删除历史记录的系统来说，密钥销毁（Crypto shredding）技术对于隐私保护和GDPR合规非常有用。

Infrastructure configuration scanner

试验

交付团队应该获得包括基础设施在内的整个技术栈的所有权，我们很早就提出了这个建议。这意味着交付团队担负了更大的责任，例如要以安全、可靠且兼容的方式来配置基础设施。所以在采纳云策略时，大多数组织对于基础设施的配置，都默认选择严格控制和集中管理的方式，以期降低风险。但这也会造成严重的生产效率瓶颈。此时另一种做法是，允许团队管理自己的配置，并使用基础设施配置扫描工具（Infrastructure configuration scanner），来确保配置的安全性和可靠性。这类可供选择的开源扫描工具，有适用于AWS的prowler和适用于Kubernetes的kube-bench。如需持续的检测，可以关注AWS Config Rules，及其他商业服务。

Service mesh

试验

服务网格（Service mesh）是一种安全、快速、可靠的运行微服务生态系统的方式。这种方式为轻松地大规模采纳微服务奠定了基础。它提供了检测、保障、跟踪、监控和故障处理功能。它提供的这些跨功能能力无需共享API网关等资产或将很多依赖库纳入到每个服务中。一种典型的实施方法是，和每个服务的进程一起在一个单独的容器中部署轻量级反向代理进程（又称为Sidecar）。Sidecar会拦截每项服务的入站和出站流量，并且提供上述的跨功能能力。利用这种方式，分布式服务团队就无需在其服务中使用代码来构建和更新网格所提供的功能。这样就可以更轻松地在微服务生态系统中采纳多语言编程。我们团队已成功的基于Istio这样的开源项目实现这种方式，而且我们将密切关注其他开源服务网格实现项目（例如Linkerd）。

Ethical OS

评估

作为ThoughtWorks的开发人员，我们对于工作中可能涉及到的道德问题十分敏感。随着社会对科技的依赖程度日益增长，软件开发团队在制定决策时必须考虑道德问题。目前已经出现的一些工具，可以帮助我们思考自己所构建的软件会在未来产生什么影响。此类工具包括技术塔罗牌和道德风险手册（Ethical OS），这两类工具都获得了广泛好评。道德风险手册为我们提供了一个思维框架和一系列工具，可以促进我们围绕软件构建方面存在的道德问题展开讨论。该手册由Institute for the Future（未来研究所）和科技与社会解决方案实验室（Tech and Society Solutions Lab）联合编制。其中探讨了一系列切实的风险，例如网瘾、多巴胺经济，而且还分析了很多值得探讨的场景。

Smart contracts

评估

我们在使用分布式账本技术（DLTs）方面积累的经验越多，遇到的当前智能合约（Smart contracts）未完善之处就越多。从理论上来看，在账本上自动添加不可否认、不可逆的合约是个好主意。但如果在你考虑如何使用现代化软件交付技术来开发它们，以及出现实施方式之间的差异时，问题就出现了。不可变数据是一回事，但不可变业务逻辑则完全是另外一回事！一定要想清楚是否在智能合约中包含逻辑，这一点真的非常重要。我们已经发现，不同的实施方式之间存在截然不同的运营特征。例如，即使合约可以演变，不同平台对这种演变的支持程度也不一样。我们的建议是，在智能合约中加入业务逻辑之前，请认真考虑，并权衡不同平台的利弊。

Transfer learning for NLP

评估

迁移学习在计算机视觉领域变得非常高效，它通过重用已有模型来加速模型训练的时间。令从事机器学习工作的人感到兴奋的是，随着ULMFiT及经过预训练的开源模型和代码示例的发布，该技术可应用于自然语言处理（NLP）。我们认为，在创建用于处理文本分类的系统时，NLP的迁移学习（Transfer learning for NLP）将显著减少所需的工作。

技术

随着社会对科技的依赖程度日益增长，软件开发团队在制定决策时必须考虑道德问题。道德风险手册为我们提供了一个思维框架和一系列工具，可以促进我们围绕软件构建方面存在的道德问题展开讨论。

(Ethical OS)

不可变数据是一回事，但不可变业务逻辑则完全是另外一回事！一定要想清楚是否在智能合约中包含逻辑，这一点真的非常重要。

(Smart contracts)

技术

使用CDC方法发布低级别的事件，别的服务直接消费这些事件，会导致服务间的集成变得非常脆弱。

(Puncturing encapsulation with change data capture)

虽然完全支持关于定期发布和演示可用软件的规则，但从中长期来看，我们发现该方法存在一些严重缺陷，因为该方法会增加有关变更排序的临时耦合，而且如果团队赶着在给定时间范围内完工，质量可能会降低。

(Release train)

Wardley mapping
评估

我们在介绍图解技术时通常很谨慎，但在讨论组织内软件资产的演进时，绘制Wardley地图（Wardley mapping）是一个非常有趣的方法。简单说来，Wardley地图从客户的需求入手，逐步绘出满足这些需求的各个功能和系统，以及这些功能和系统的演进，以呈现出组织内部存在的价值链。这项技术的价值在于协作绘图的过程，而不只在于产出的图表本身。我们建议由合适的人员一起绘制Wardley地图，并将这些图表视为活跃、不断发展的事物，而非已完成的产出物。

Productionizing Jupyter Notebooks
暂缓

数据科学家越来越钟情于Jupyter Notebook，纷纷将其用于探索性数据分析、早期原型开发和知识分享等。随着Jupyter Notebook日益普及，开始出现Jupyter Notebooks生产化（Productionizing Jupyter Notebooks）的趋势。虽然我们不想阻止任何人使用自己喜好的工具，但是仍然建议不要尝试用Jupyter Notebooks构建可维护、规模化、生产化的在线系统，因为其缺乏构建规模化、生产化所需最基本的版本控制、错误处理、模块化手段和可扩展性。相比之下，我们更鼓励开发者和数据科学家共同努力，利用持续交付实践，采纳正确的框架开发相应的解决方案，帮助数据科学家构建可用于生产环境的机器学习模型。我们建议对生产化Jupyter Notebooks保持谨慎，以克服机器学习在持续交付流程和自动化测试上的先天不足。

Puncturing encapsulation with change data capture
暂缓

变更数据捕获(CDC)是一种非常强大的方法，用于从系统中拉取数据库变更，并对该数据执行一些操作。其中最受欢迎的一种方式是，使用数据库的事务日志来识别变更，然后将这些变更直接发布到可由其他服务消费的事件总线中。该方法特别适用于将单体分解为微服务这样的情况，但用作微服务之间的一级集成时，这会导致封装性变弱，并将源服务的数据层泄露到事件契约中。我们之前谈论过领域范围事件和其他方法，这些方法强调了正确的领域事件建模的重要性。我们看到一些项目在使用CDC方法，发布低级别的事件，别的服务直接消费这些事件。这样会破坏变更数据捕获的封装（Puncturing encapsulation with change data capture），可能会产生滑坡效应，导致服务间的集成变得非常脆弱。我们想要通过此技术点提醒大家关注这一问题。

Release train
暂缓

我们已亲眼见证，组织通过使用版本火车概念（Release train），从极低的发布频率成功转向更高频率。版本火车是一种用于协调跨多个团队或具有运行时依赖性组件的发布技术。无论所有预期功能是否已准备就绪，所有版本根据一个固定且可靠的时间表发布（火车不会等你，如果错过，就只能等下一趟了）。虽然我们完全支持关于定期发布和演示可用软件的纪律性，但从中长期来看，我们发现该方法存在一些严重缺陷，因为该方法会增加有关变更排序的临时耦合，而且如果团队赶着在给定时间范围内完工，质量可能会降低。我们更倾向于关注在必要的架构与组织的方法，来支持独立发布。虽然火车对于提升较慢团队的速度非常有用，但我们也看到它给快速团队带来了上限。所以我们认为，在使用该技术时，应尽量小心谨慎。

Templating in YAML
暂缓

随着基础设施的复杂度不断增加，用于定义基础设施的配置文件也越来越复杂。AWS CloudFormation、Kubernetes和Helm等工具要求使用以JSON或YAML来编写配置文件，以简化编写和处理。然而多数团队很快会碰到一些诸如在不同区域部署同一个服务这种相似但又不尽相同的配置问题。很多人都在使用YAML（或JSON）模版（Templating in YAML）时受到挫折。问题在于，JSON和YAML的语法都需要各种尴尬的折衷，才能加入条件和循环。我们建议用Python之类的通用编程语言来消费API。若没有API可用，则可以使用Jsonnet之类的专用模板语言。

平台

Contentful
采纳

无头内容管理系统(CMS)正在成为数字化平台的常用组件。Contentful是一款现代化的无头CMS，我们的团队已成功将其集成到开发工作流程中。我们尤其喜爱它的API优先和CMS即代码的实现特点。它支持强大的“内容建模原语即代码”和内容模型演化脚本等特性，因此可以对其像处理其他数据存储schema那样进行处理，并且能将演进式数据库设计的实践应用于CMS的开发中。Contentful的稳健性和一系列新功能(包括沙盒环境)，给我们团队留下了深刻的印象，成为我们在这个领域的第一选择。

AWS Fargate
试验

AWS Fargate这个在AWS上实现docker即服务的计算引擎，目前已经能在多个亚马逊云区域上使用。对于一些需要运行Docker容器的团队来说，这是一个非常好的选项，因为它比AWS Lambda更加强大，同时不用管理EC2实例或者Kubernetes集群。我们的许多团队都感受到Fargate的良好使用体验，但与此同时这种便捷的托管式服务也带来了较高的成本。

EVM beyond Ethereum
试验

以太坊虚拟机(EVM)最初是为以太坊主网络设计的。但如今，大多数团队不再想要从头开始发明区块链；相反，他们会选择“超越以太坊的EVM(EVM beyond Ethereum)”。我们看到众多区块链团队选择对以太坊进行分支(如Quorum)或实现EVM规范(如Burrow、Pantheon)，并添加他们自己的设计。这样做不仅是为了重用以太坊的设计，还可以利用其生态系统和开发人员社区。对于许多开发人员而言，“智能合约”的概念差不多等同于“以Solidity编写智能合约”。虽然以太坊本身具有一些限制，但围绕EVM生态系统的技术正在繁荣发展。

InfluxDB
试验

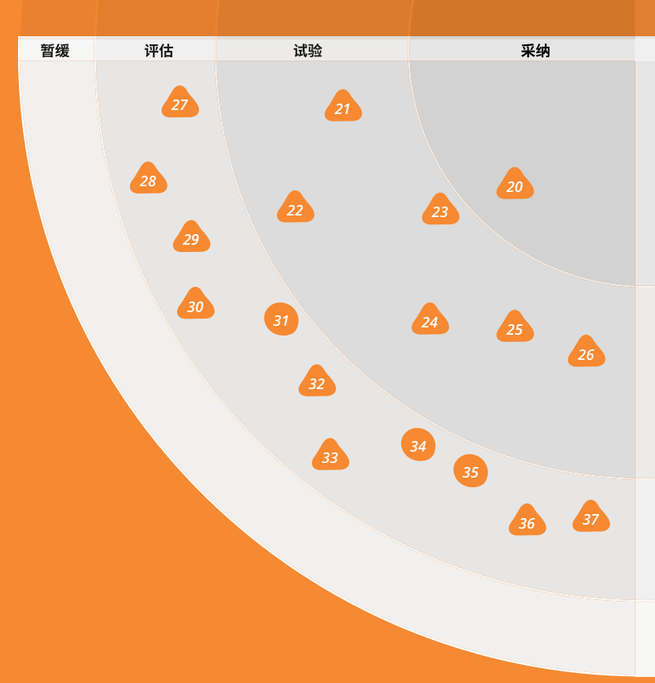
时序数据库(TSDB)已经问世一段时间了。随着符合时序模型的使用场景愈发常见，TSDB日益成为主流。InfluxDB仍然是TSDB的理想选择，主要用于监控场景。TICK Stack就是一款以InfluxDB作为核心的监控解决方案。Influx 2.0的alpha版最近引入了Flux(一种用于查询和处理时序数据的脚本语言)。虽然Flux目前仍处于早期阶段，且无法断定能比InfluxDB获得更广泛的应用，但它承诺会比InfluxQL更强大且更具表达力，且能将时序分析工作交由数据库来完成。然而，InfluxDB只有企业版才能提供集群支持，因此在项目中需要留意这个限制。

采纳
20. Contentful

试验
21. AWS Fargate
22. EVM beyond Ethereum
23. InfluxDB
24. Istio
25. Kafka Streams
26. Nomad

评估
27. CloudEvents
28. Cloudflare Workers
29. Deno
30. Hot Chocolate
31. Knative
32. MinIO
33. Prophet
34. Quorum
35. SPIFFE
36. Tendermint
37. TimescaleDB

暂缓



平台

时序数据库(TSDB)已经问世一段时间了。随着符合时序模型的使用场景愈发常见，TSDB日益成为主流。InfluxDB仍然是TSDB的理想选择。

(InfluxDB)

虽然事件是常用的函数即服务触发机制，并且所有云提供商都以某种形式提供支持，但目前的私有规范阻止了跨云互通。

(CloudEvents)

Istio
试验

Istio正逐渐成为将微服务生态系统付诸应用的标准基础设施。其开箱即用的横切关注点的实现，已经帮助我们快速实现了微服务。这些横切关注点实现包括：服务发现、服务之间和从请求到服务之间的安全性、可观测性(包括遥测和分布式跟踪)、滚动发布和弹性。Istio是我们一直使用的服务网格技术的主要实现平台。我们非常享受它的每月发布及无缝升级所带来的持续改进。我们使用Istio来启动项目，从一开始就能获得可观测性(跟踪和遥测)和服务之间的安全性。我们正密切关注其在网格内外各处服务之间身份验证方面所做的改进。此外，我们希望看到Istio为配置文件建立最佳实践，从而在为服务开发人员提供自主权和为服务网格运营商提供控制权之间达到平衡。

Kafka Streams
试验

Kafka Streams是一个轻量级库，用于构建流式应用程序。该库支持基本的流处理API，如join、filter、map和aggregate。另外对于窗口化和会话等常见用例，该库还支持本地存储。与Apache Spark和Alpakka Kafka等其他流处理平台不同，Kafka Streams非常适合不需要大规模分布和并行处理的场景。因此即使没有诸如集群调度程序等其他基础设施，它同样也可以工作。当在Kafka生态系统中进行运维工作时，Kafka Streams自然是个不错的选择。当必须严格按顺序处理数据且只能处理一次时，Kafka Streams尤其有用。Kafka Streams的一个特别用例，是构建change data capture(CDC)台。

Nomad
试验

HashiCorp陆续推出了很多有趣的软件。2017年3月那期技术雷达特意点评了HashiCorp Vault，本期雷达也在多处点评了与Terraform相关的很多工具。由于Nomad具有很好的使用体验，所以本期我们将其移至雷达的“试验”环。尽管Kubernetes持续受到关注，但我们很喜欢Nomad的普适性。它可以处理任何类型的系统，而非仅能用于容器。它原生支持Java和Golang，也支持批量任务和分布式的定时任务。我们不仅欣赏它能专注于多云和混合云的运维领域(能避免粘性云这一点变得很重要)，也欣赏它的良好的处理效果。

CloudEvents
评估

在函数代码之外，以无服务器函数编写的应用程序与托管它们的云平台耦合紧密。虽然事件是常用的函数即服务触发机制，并且所有云提供商都以某种形式提供支持，但目前的私有规范阻止了跨云互通。CloudEvents是一项正在迅速发展的新标准，并已被纳入CNCF Sandbox。该标准仍在积极开发过程中，目前已支持多种语言。微软已宣布在Azure上对CloudEvents提供最高等级的支持。我们期待其他云提供商能够效仿。

Cloudflare Workers
评估

大多数现代服务器端或无服务器代码执行平台，都是围绕容器或虚拟机来构建的。但是，Cloudflare Workers采取了不同的方法来托管无服务器计算产品。该平台使用了原先为Chrome而开发的开源JavaScript引擎V8 Isolates，用来在其大型CDN网络上实现函数即服务(FaaS)。运行于该平台的代码可以使用JavaScript或任何能够编译成WebAssembly的编程语言编写，数据可通过Cloudflare的缓存或键值存储库进行访问。对开发人员而言，其主要好处是性能——位于边缘网络的CDN接近最终用户，冷启动只需5毫秒。而对供应商而言，其好处包括其较低内存开销所带来的密集包隔离，以及通过减少进程上下文切换所带来的高性能。这绝对是一个有趣且值得关注和评估的平台。

Deno
评估

说起在服务器端使用JavaScript编程，我们的心情可谓五味杂陈，特别是当这么做仅是为了避免多语言编程时。不过，如果最终决定在服务器端使用JavaScript或TypeScript，可以考虑Deno。Node.js发明者Ryan Dahl编写了Deno，旨在避免他所意识到的Node.js中所存在的缺陷。该平台提供严格的沙盒系统和内置依赖项与包管理功能，支持开箱即用的TypeScript。Deno使用Rust和V8构建而成。

Hot Chocolate
评估

GraphQL生态系统和社区正不断发展。Hot Chocolate是用于.NET（包括新的core和原先的传统框架）的GraphQL服务器。该平台可用于构建和托管schema，并能用于处理针对这些schema的查询。Hot Chocolate的开发团队近期增添了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。虽然该功能会被以多种方式误用，但还是值得对其进行评估。

Knative
评估

无服务器架构的应用，让FaaS编程风格在开发人员之间越来越普及。该架构通过独立构建和部署的函数，帮助开发人员专注于解决核心业务问题。这些函数能响应事件、运行业务流程、在流程中生成其他事件，完成任务后随即消失，不再消耗资源。以前，AWS Lambda或Microsoft Azure Functions等专有无服务器平台已实现这种编程范式。Knative是基于Kubernetes的开源平台，用来运行FaaS工作负载。Knative有几点突出之处：开源且适用于任何供应商；实现了CNCF无服务器工作小组白皮书中所定义的无服务器工作流；通过实现符合CNCF CloudEvents规格的事件接口，来确保跨服务的互操作性；尤其重要的是，它能够解决在运维混合FaaS与长期运行的容器化架构时所遇到的常见挑战。该平台易与Istio和Kubernetes集成。例如，通过在不同版本的函数之间切换流量，开发人员可以利用Istio实施金丝雀发布策略。对于处在相同Kubernetes环境中的长期运行的容器服务和FaaS程序，开发人员都可以享受到Istio所提供的可观测能力。我们预计，Knative开源事件接口将继续支持更多底层源和目的事件的集成。

MinIO
评估

若要在云端存储非结构化数据和一些结构化数据，对象存储是一个热门选择。一方面我们不建议使用通用云，另一方面如果想要最大限度降低对象存储的云粘性风险，MinIO会非常有用。凭借与S3兼容的API层，MinIO能够在多个云提供商中实现对象存储，这些云提供商包括AWS、Azure和Google Cloud Platform(GCP)。我们已成功将其运用于具有灵活目标基础设施（从数据中心到云提供商）的产品中。

Prophet
评估

即使在深度学习的时代，统计模型仍然在商业决策支持中发挥重要作用。时间序列模型广泛用于预测库存、需求、客户流量等信息。手动制作这些模型以确保其稳健灵活，这通常是专业统计学家或大型商业软件供应商的职责。Prophet是一个商业预测软件包的开源替代方案，它可以使用R或Python进行编程。Facebook 声称已在内部使用 Prophet 进行了大规模的商业预测，并以开源包的形式将其分享出来。我们比较满意的是，Prophet消除了模型构建、维护和数据操作中一些单调乏味的工作，让人类分析师和专家能够专注于他们所擅长的领域。

平台

如果最终决定在服务器端使用JavaScript或TypeScript，可以考虑Deno。该平台提供严格的沙盒系统和内置依赖项与包管理功能，支持开箱即用的TypeScript。

(Deno)

无服务器架构的应用，让FaaS编程风格在开发人员之间越来越普及。Knative是基于Kubernetes的开源平台，用来运行FaaS工作负载。

(Knative)

平台

Tendermint是一种BFT状态机制复制引擎，可以帮助你实现自己的区块链系统。

(Tendermint)

Quorum
评估

Quorum是“企业版以太坊”，旨在提供网络许可、事务隐私以及更高的性能。我们的一个团队深入使用了Quorum；然而到目前为止，他们的体验并不算好。所遇到的挑战部分来自复杂的智能合约编程，部分来自以及Quorum本身。例如，它与负载均衡器配合欠佳，且仅支持部分数据库，这带来了严重的部署负担。同时我们遇到了一些稳定性和兼容性问题，尤其是在隐私事务方面。由于JPM摩根币的出现，Quorum近期吸引了大量关注。然而，从技术角度来说，我们建议在实施Quorum时保持谨慎，同时关注其发展。

SPIFFE
评估

在为服务之间的端到端加密和双向认证提供开箱即用的解决方案方面，服务身份的SPIFFE标准化已成为重要一环。SPIFFE标准基于OSS SPIFFE运行时环境(SPIRE)，由SPIRE自动向服务提供密码学上可验证的身份。Istio也默认使用SPIFFE。SPIFFE 支持许多使用场景，包括身份转换、OAuth客户端认证、mTLS“随处加密”和工作负载可观察性。ThoughtWorks正与Istio和SPIFFE社区积极合作，在原有服务身份提供商和基于SPIFFE的身份之间架起桥梁，让mTLS在服务之间、服务网格内部和外部变得随处可用。

Tendermint
评估

拜占庭容错(BFT)是加密货币和区块链系统的一个基本问题。在存在多个任意错误进程的情况下（其中还包括恶意欺诈），它要求整个系统针对单个数据值达成共识。Tendermint是一种BFT状态机制复制引擎，可以帮助你实现自己的区块链系统。Tendermint Core共识引擎负责处理点对点通信和共识部分，你只需实现应用程序的剩余部分（例如构建交易和验证加密电子签名）并通过ABCI与Tendermint Core通信即可。部分区块链实现已选择Tendermint作为共识引擎。

TimescaleDB
评估

在往期雷达报告中，我们讨论了适用于NoSQL的PostgreSQL。PostgreSQL的成熟性和可扩展性，促进了在基于Postgres引擎所打造的持久化存储库方面的稳步创新。其中，TimescaleDB引起了我们的关注。这个数据库支持快速写入，可实现对时间序列数据的优化查询。虽然尚未像InfluxDB一样功能齐全，但TimescaleDB的数据模型和查询功能可以作为备选方案。如果只有适中的可扩展性需求，喜欢使用SQL，青睐PostgreSQL所提供的稳定性和熟悉的管理界面，那么你应该评估TimescaleDB。

工具

Cypress
采纳

我们不断收到关于Cypress、TestCafe和Puppeteer等“后Selenium”web UI测试工具的积极反馈。运行端到端测试时经常会遇到一些棘手的问题，如运行时间过长、测试过于零碎、还需要修复无头模式下运行的测试所导致的CI失败。我们的团队借助Cypress很好地解决了性能差、响应时间长、资源加载慢等常见问题。Cypress已成为我们团队内部执行端到端测试的首选工具。

Jupyter
采纳

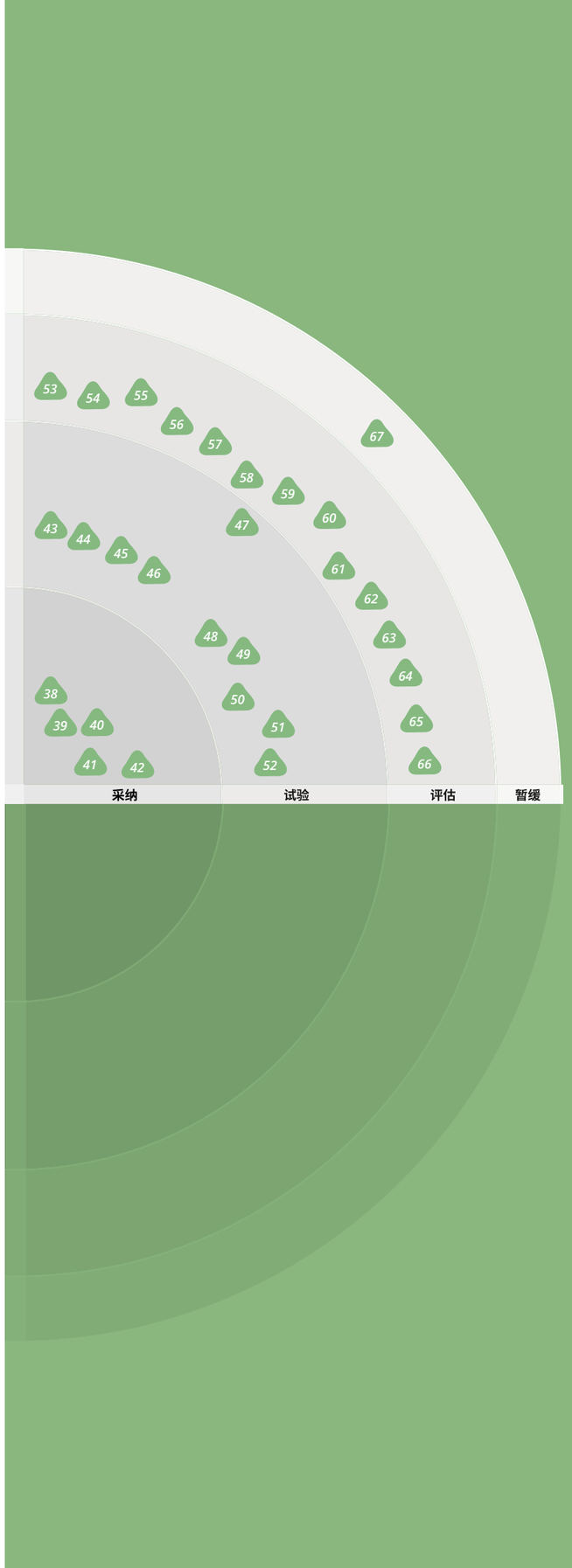
在过去几年里，我们注意到用于数据分析的notebooks越来越受欢迎。这些notebooks受到Mathematica的启发，将文本、可视化和代码结合到一个可动态交互计算的文档中。我们的团队广泛使用Jupyter notebooks进行数据分析和机器学习方面的探索和原型设计。在本期雷达报告中，我们将Jupyter移至“采纳”来体现其已成为现今默认的Python notebooks。然而，我们仍对在生产环境中使用Jupyter notebooks持谨慎态度。

LocalStack
采纳

使用云服务时面对的一个挑战是如何在本地进行开发和测试。LocalStack为AWS解决了这个问题。它提供了各种 AWS 服务的本地测试替身实现，包括S3、Kinesis、Dynamodb和Lambda等。它基于现有的最佳工具如Kinesalite、Dynalite、Moto等构建，并增加了进程隔离与错误注入的功能。LocalStack的使用很简单，可以通过其附带的一个简单的JUnit运行器和JUnit 5扩展来使用，也可以在一个docker容器中运行。它已成为许多团队在测试部署在AWS上的服务时选用的默认工具。

Terraform
采纳

Terraform正在迅速成为通过声明式定义来创建和管理云基础设施的首选工具。经由Terraform实例化的那些服务器配置信息，通常会留给诸如Puppet、Chef或Ansible这样的工具来处理。我们之所以喜欢Terraform，一方面是因为它的文件语法可读性很强，另一方面是因为它支持众多云服务提供商，但并没有企图在这些提供商之间提供一个矫揉造作的抽象构件。活跃的社区将为大多数云提供商的最新特性提供支持。两年前我们曾第一次谨慎地提及Terraform。如今，Terraform已经有了长足的进步，并且发展成为一款稳定且具有良好生态系统的产品，这一点已经在我们的项目中得到证明。状态文件管理的问题现在已经可以由Terraform的“远程状态后端”来解决。我们成功地使用AWS S3实现了这个目的。



采纳
38. Cypress
39. Jupyter
40. LocalStack
41. Terraform
42. UI dev environments

试验
43. AnyStatus
44. AVA
45. batect
46. Elasticsearch LTR
47. Helm
48. InSpec
49. Lottie
50. Stolon
51. TestCafe
52. Traefik

评估
53. Anka
54. Cage
55. Cilium
56. Detekt
57. Flagr
58. Gremlin
59. Honeycomb
60. Humio
61. Kubernetes Operators
62. OpenAPM
63. Systems
64. Taurus
65. Terraform provider GoCD
66. Terratest

暂缓
67. Handwritten CloudFormation

工具

Storybook、React Styleguidist、Compositor 和 MDX等，这些工具专注于提升用户体验设计师与开发人员之间的协作水平。

(UI dev environments)

大量的精力仍然被浪费在部署本地开发环境和排查“works on my machine”（在我的机器上可以工作）的问题上。Batect可帮助轻松搭建和共享基于Docker的构建环境，还能够启动容器来执行完全不依赖于本地配置的构建任务。

(batect)

UI dev environments
采纳

随着越来越多的团队拥抱DesignOps，该领域的实践和工具也日渐成熟。UI开发环境专注于用户体验设计师与开发人员之间的协作（UI dev environments），为UI组件的快速迭代提供了综合环境。目前在该领域可用的工具包括：Storybook、React Styleguidist、Compositor和MDX。这些工具既可以在组件库或设计系统的开发过程中单独使用，也可以将其嵌入到web应用程序中使用。通过使用这些工具，许多团队在开发准备工作中缩短了UI反馈周期并改善了UI工作的时间。于是，使用UI开发环境成为了我们合理的默认选择。

AnyStatus
试验

开发人员习惯于每天推送许多小的提交，因此我们依赖于监控器来通知我们何时构建通过。AnyStatus是一款轻量级Windows桌面应用，将不同来源的度量指标和事件汇总到一处。例如，构建结果与发布、不同服务和操作系统度量指标的健康检查。可以将该应用视为CCTray的加强版。它还可以作为Visual Studio插件被使用。

AVA
试验

AVA是Node.js的测试运行器。虽然JavaScript是单线程的，但在Node.js里，其异步的特性使得IO可以并行。AVA利用这个优点可以并发执行你的测试，这对于IO繁重的测试尤其有益。此外，测试文件作为单独进程并行运行，让每个测试文件可以获得更好的性能和独立的环境。与Jest等功能齐全的的框架相比，AVA是一款有态度的工具，且强制你编写原子测试用例。

batect
试验

大量的精力仍然被浪费在部署本地开发环境和排查“works on my machine”（在我的机器上可以工作）的问题上。多年来，我们的团队已经采纳“检查并实施”的方法，使用脚本化方法来确保本地开发环境的配置始终一致。batect是由ThoughtWorker开发的一款开源工具，可帮助轻松搭建和共享基于Docker的构建环境。Batect作为构建系统的入口点脚本，能够启动容器来执行完全不依赖于本地配置的构建任务。对构建配置和依赖项的更改仅通过源码管理即可共享，无需在本地机器或CI代理上进行任何更改或安装。在该领域的其他工具中，我们偏向于使用Cage，但我们也看到batect正在以符合我们团队需求的方向迅速发展。

Elasticsearch LTR
试验

搜索面临的一大挑战就是确保对用户来说，最相关的结果显示在列表的顶部。这就是排序学习（LTR）的用武之地。LTR是指应用机器学习来对搜索引擎检索的文档进行排序。如果你使用的是Elasticsearch，则可以通过Elasticsearch LTR插件实现搜索相关的排序。该插件使用RankLib在训练阶段生成模型。然后，当查询Elasticsearch时，可以使用该插件对排列靠前的结果进行“重打分”。我们已经将其运用于一些项目中，并且取得了令人满意的结果。另外还有面向Solr用户的等效LTR解决方案。

Helm
试验

Helm是针对Kubernetes设计的一种包管理工具。它附带了一些精选的Kubernetes应用并在官方的Charts仓库中维护。Helm包含两个组件：命令行工具“Helm”与集群组件“Tiller”。对Kubernetes集群进行安全加固是一个广泛而细致的主题，但我们仍然强烈建议在基于角色的访问控制（RBAC）环境中搭建Tiller。我们在一些客户项目中已经使用了Helm，它的依赖管理、模板和钩子(Hook)机制大大简化了Kubernetes中应用程序的生命周期管理。尽管有如此多的优势，我们还是建议谨慎行事—Helm的YAML模板目前可能有些难以理解，而且Tiller仍然存在未完善之处。预计Helm 3的新版本将会着力解决这些问题。

InSpec
试验

组织如何为交付团队赋予自主权，同时仍然确保他们部署的解决方案安全且合规?如何确保服务器在部署后仍然保持一致的配置，不会有偏差?InSpec定位于提供持续合规性与安

全性的解决方案，但你也可以将其用于一般基础设施测试。InSpec允许创建声明式基础设施测试，然后可以针对经过预配的环境（包括生产环境）持续运行这些测试。我们的团队对其资源以及适用于多平台的匹配器的可扩展性设计给予极大的好评。我们建议大家尝试将InSpec作为确保合规性与安全性问题的解决方案。

Lottie
试验

良好的UI动画可以显著提高用户体验。然而，在应用中重现设计人员的精美动画，对于开发人员来说通常是一项具有挑战性的任务。Lottie是适用于Android、iOS、web和Windows的库，用于解析通过BodyMovin导出的JSON格式的Adobe After Effects动画，并在移动端和web端本地渲染。设计人员和开发人员都可以使用他们熟悉的工具，开展流畅的协作。

Stolon
试验

搭建高可用的PostgreSQL实例可能很棘手，但这正是Patroni深受喜爱的原因--它可以帮助我们加速PostgreSQL集群的设置。Stolon是另一个我们已经成功在生产环境中使用的、利用Kubernetes运行PostgreSQL实例的高可用性（HA）集群的工具。尽管PostgreSQL支持开箱即用的流复制，但HA设置的难题在于确保客户端始终连接到当前主实例。Stolon通过主动关闭与未选中主实例的连接并将请求路由到有效的主实例，强制连接到正确的PostgreSQL主实例，这是Stolon的优势所在。

TestCafe
试验

在使用Cypress、TestCafe和Puppeteer等“后Selenium”web UI测试工具方面，我们拥有良好的体验。TestCafe支持采纳JavaScript或TypeScript来编写测试，并在浏览器中运行测试。TestCafe提供了开箱即用的并行执行、HTTP请求模拟等有用的功能。TestCafe使用异步执行模型而无需指定等待时间，有效提升了测试套件的稳定性。它的选择器API可更轻松实现PageObject模式。TestCafe最近发布了1.0.x版本，进一步提升了稳定性和功能性。

Traefik
试验

Traefik是一款开源的反向代理及负载均衡器。如果只是需要具有简单路由功能的边缘代理，而非NGINX和HAProxy这样全功能的重量级产品，就可以考虑Traefik。它提供了微服务所必不可少的免重载更新配置、度量、监控与断路器等功能。同时能够很好地与Let’s Encrypt集成以提供SSL终端，也可以和Kubernetes、Docker Swarm或Amazon ECS等基础设施组件集成，从而自动选取包含在其负载均衡中的新服务或实例。

Anka
评估

Anka是一组为iOS和macOS开发、创建、管理和分发构建，为测试macOS提供可重现虚拟环境的工具。该工具为macOS环境提供类似Docker的体验：即时启动、使用CLI管理虚拟机、注册表用于版本管理、标记用于分发的虚拟机。在向客户提议macOS私有云解决方案时，我们发现了Anka。如果要将DevOps工作流应用于iOS和macOS环境，该工具值得考虑。

Cage
评估

Cage是Docker Compose的开源封装容器，允许你将多个依赖组件作为一个大型应用程序进行配置和运行。你可以编排Docker镜像、代码库中的service源码、脚本等组件的执行，以加载数据存储和pods，它们是作为一个整体一起运行的容器。Cage使用Docker Compose v2配置文件格式。它填补了Docker Compose的一些空白，如对多环境的支持，包括用于在本地开发机器上运行分布式应用的开发环境，用于运行集成测试的测试环境，以及生产环境。

工具

InSpec定位于提供持续合规性与安全性的解决方案，但你也可以将其用于一般基础设施测试。

(InSpec)

利用Linux中的eBPF框架，Cilium通过以基于服务，pod或容器认证的方式插入安全可见性来提供具备API感知的网络和安全性。

(Cilium)

工具

Detekt是一个适用于Kotlin的静态代码分析工具。它能够发现代码中的坏味道和复杂性。你可以通过命令行运行它，也可以使用其插件集成一些热门的开发者工具。

(Detekt)

在日志管理领域，Humio是一款相对较新的工具。该工具完全从零开始构建，通过基于定制设计的时序数据库的内置查询语言，在日志提取和查询方面表现得非常快。

(Humio)

Cilium
评估

iptables等传统Linux网络安全方法根据IP地址和TCP/UDP端口进行过滤。然而，在动态微服务环境中，这些IP地址经常变动。利用Linux中的eBPF框架，Cilium通过以基于服务，pod或容器认证的方式插入安全可见性来提供具备API感知的网络和安全性，而不是IP地址识别。通过安全性与寻址分离，Cilium作为一个新的网络保护层可以发挥重要作用，我们建议你查看相关内容。

Detekt
评估

Detekt是一个适用于Kotlin的静态代码分析工具。它能够发现代码中的坏味道和复杂性。你可以通过命令行运行它，也可以使用其插件集成一些热门的开发者工具，例如Gradle（用于在项目构建时执行代码分析）、SonarQube（用于除静态代码分析外的代码覆盖率统计）和IntelliJ等。Detekt能够给Kotlin应用的构建流水线锦上添花。

Flagr
评估

特性开关是持续部署场景中的重要技术。我们已经看到了一些不错的自制解决方案，但Flagr提供的方案的确深得我们青睐：将一个完整的特性开关作为一个服务分发到Docker容器之中。Flagr为所有主流语言提供了SDK，具有简单且有据可查的REST API，并提供便捷的前端界面。

Gremlin
评估

Gremlin是一个SaaS解决方案，可供组织进行混沌试验，以助其测试系统的弹性。它提供一系列能导致系统失效（包括资源、网络和状态失效）的攻击。这些攻击既可以随机运行，也可以定时运行，且只需进行很少的设置（特别是对于可以运行Helm来安装Gremlin的Kubernetes用户）。Gremlin客户端还有基于web的友好的用户界面，便于轻松执行和管理混沌试验。

Honeycomb
评估

Honeycomb是一个可观测性工具，它从生产系统中提取出丰富的数据，并通过动态采样使其可管理。开发人员可以记录大量丰富的事件，并在之后决定如何划分和关联它们。这种交互方式对于当今大型分布式系统非常有益，因为我们现在已经不再能合理预测出我们将会需要对生产系统提出哪些问题了。

Humio
评估

在日志管理领域，Humio是一款相对较新的工具。该工具完全从零开始构建，通过基于定制设计的时序数据库的内置查询语言，在日志提取和查询方面性能非常快。从提取、可视化和报警提醒的角度来看，该工具能够与几乎所有工具相集成。日志管理领域已被Splunk和 ELK Stack主导，所以，有替代选择也是一件好事。我们将持续关注 Humio 的发展。

Kubernetes Operators
评估

我们对于Kubernetes对行业产生的影响兴奋不已，但也担心随之而来的运维复杂度。保持Kubernetes集群启动并运行、管理在该集群上部署的软件包都需要特殊技能和时间。升级、迁移、备份等运维流程经常会是一项全职工作。我们认为Kubernetes Operator会对降低复杂度起到关键作用。该框架提供了一套标准机制，为在Kubernetes集群中运行的软件包描述了自动化运维流程。虽然Operator由RedHat发起和推广，但多个社区为常用开源软件包（如Jaeger、MongoDB和Redis）开发的Operator已初露头角。

OpenAPM
评估

采纳开源方案替代流行商业软件包的挑战之一是整理复杂的项目格局，以了解你需要哪些组件、哪些组件能够很好地配合使用、以及确切地知道每个组件涵盖总体解决方案的哪个部分。这在可观测性领域尤其困难，该领域的标准实践是购买一个全面但昂贵的软件包来执行所有工作。OpenAPM简化了可观测性工具的开源选择流程。它显示目前提供的按照组件角色分类的开源软件包，因此你可以交互地选择可兼容的组件。只要确保你使用最新版该工具，它就可以帮助你浏览一系列令人困惑的可行工具。

Systems
评估

我们工作的许多流程，都很容易被认为是因果关系的线性链。在大部分时间里，我们要处理较为复杂的系统，其中正负反馈循环都会对结果产生影响。Systems是一组用于描述、执行和可视化系统图表的工具。通过使用紧凑的DSL独立运行或在Jupyter Notebook内运行，利用该工具描述相对复杂的流程和信息流都是一项相当简单的工作。这款工具虽然较为小众，但十分有趣。

Taurus
评估

Taurus是一个通过Python实现的、简便的应用程序和服务性能测试工具。它封装了很多诸如Gatling和Locust的性能测试执行器。该工具能够通过命令行运行，可以很容易地被集成到持续交付流水线中，以便在流水线的不同阶段运行性能测试。Taurus 还具备出色的报告功能，不仅能将报告输出到基于文本的控制台，还能将其集成到交互式web UI上。我们的团队发现，配置Taurus YAML文件非常简单，因为您可以使用多个文件来描述每个测试场景，并引用基础执行器的场景定义。

Terraform provider GoCD
评估

Terraform provider GoCD可以让你使用Terraform（一款在基础设施即代码领域被广泛使用的成熟工具）来构建流水线。利用它可以使用HashiCorp配置语言(HCL)来配置流水线，并支持Terraform提供的所有特性，包括工作区、模块和远程状态。这是我们在之前的流水线即代码词条中曾重点介绍过的Gomatic的出色替代方案。它使用的Golang SDK提供了适用于GoCD API的自动化回归测试，这些测试能最大限度减少升级时出现的问题。

Terratest
评估

我们广泛的使用Terraform来代码化配置云基础设施。Terratest是一个Golang库，用来简化基础设施代码的自动化测试编写。测试的运行会创建真实的基础设施组件（如服务器、防火墙或负载均衡器）、在它们之上部署应用程序并使用Terratest验证预期的行为。在测试结束后，Terratest可以取消应用的部署并清理资源。对于基础设施在真实环境中的端到端测试，该工具非常有用。

Handwritten CloudFormation
暂缓

AWS CloudFormation是专用于AWS基础设施即代码的声明式语言。手写CloudFormation文件通常是自动化构建AWS基础设施的默认方式。当启动一个小型项目时，这可能是一个合理的选择，但我们的团队和业界发现，随着基础设施变得越来越复杂，手写Cloudformation（Handwritten CloudFormation）很不易于扩展。对于大型项目而言，手写CloudFormation文件的缺点愈加凸显：比如代码可读性差、缺少命令式结构、参数定义与使用受限很大、缺少类型检查。解决这些缺点催生了丰富的开源及第三方工具生态系统。我们发现Terraform是一个明智的首选方案，它不仅能够避免CloudFormation的缺点，还拥有一个活跃的社区为AWS最新特性和缺陷修复提供支持。除了Terraform，你也可以选择其它工具和语言，比如troposphere、sceptre、Stack Deployment Tool和Pulumi。

工具

Systems是一组用于描述、执行和可视化系统图表的工具，其中正负反馈循环都会对结果产生影响。

(Systems)

Terraform provider GoCD可以让你使用Terraform（一款在基础设施即代码领域被广泛使用的成熟工具）来构建流水线。提供了适用于GoCD API的自动化回归测试，这些测试能最大限度减少升级时出现的问题。

(Terraform provider GoCD)

语言&框架

Apollo
采纳

我们团队的报告中指出，在构建使用GraphQL从后端服务中访问数据的React应用时，Apollo已经成为首选库。虽然Apollo项目还提供服务器框架和GraphQL网关，但其客户端之所以引起我们的关注，是因为它简化了将UI组件绑定到和GraphQL后端提供的数据这一问题。简言之，这意味着与使用REST后端和Redux相比，Apollo可以减少代码量。

MockK
采纳

为Kotlin应用程序编写测试时，MockK是我们的首选模拟工具。我们喜欢使用该库是因为其对协程或lambda块等Kotlin语言特性提供一流支持。作为原生库，它帮助我们的团队编写清晰简洁的Kotlin应用程序测试代码，无需使用蹩脚的的Mockito或PowerMock包装器。

TypeScript
采纳

TypeScript是一种静态类型语言，也是JavaScript的超集，现已成为我们明确的默认之选。大型项目从类型安全中获益最多。我们的开发人员喜欢它简单的配置管理、良好集成的IDE支持、安全地重构代码以及逐步采纳类型的能力。凭借其优秀的TypeScript类型定义资源库，我们可以从丰富的JavaScript库中受益，同时也获得类型安全的好处。

Apache Beam
试验

Apache Beam是一个开源的统一编程模型，用于定义和执行数据并行处理流水线的批处理与流式传输。Beam模型基于数据流模型，允许我们以优雅的方式表达逻辑，以便在批处理、窗口化批处理或流式传输之间轻松切换。大数据处理生态系统已经取得了长足发展，这可能会导致人们难以选择正确的数据处理引擎。允许我们在不同运行程序之间切换，这是选择Beam的一个关键原因。几个月前，它支持了Apache Samza，这是除Apache Spark、Apache Flink和Google Cloud Dataflow之外的又一个新的运行程序。不同运行程序具有不同能力，且提供轻便的API是一项困难的任务。Beam将这些运行程序的创新主动应用于Beam模型，并与社区合作以影响这些运行程序的路线图，从而试图达到微妙的平衡。Beam具有包括Java、Python和Golang多种语言的SDK。我们也成功使用了Scio，它为Beam提供了Scala包装器。

采纳
68. Apollo
69. MockK
70. TypeScript

试验
71. Apache Beam
72. Formik
73. HiveRunner
74. joi
75. Ktor
76. Laconia
77. Puppeteer
78. Reactor
79. Resilience4j
80. Room
81. Rust
82. WebFlux

评估
83. Aeron
84. Arrow
85. Chaos Toolkit
86. Dask
87. Embark
88. fastai
89. http4k
90. Immer
91. Karate
92. Micronaut
93. Next.js
94. Pose
95. react-testing-library
96. ReasonML
97. Taiko
98. Vapor

暂缓

语言&框架

joi是一款针对JavaScript对象结构的描述语言与验证工具，它独立于任何Web应用框架。

(joi)

响应式系统具有更好的可扩展性和伸缩性，但是调试成本更高，学习曲线也更陡峭。响应式系统具有更好的可扩展性和伸缩性，但是调试成本更高，学习曲线也更陡峭。

(Reactor)

Formik
试验

表单操作在React中一直是一件比较繁琐复杂的事情，使用Formik可以极大的简化表单操作。Formik是一个React高阶组件，它将状态管理本地化，协助完成提交，并且可以使用Yup来简化数据验证。

HiveRunner
试验

HiveRunner是适用于Apache Hadoop Hive查询的基于JUnit4的开源单元测试框架。在Hive SQL中编写重要分析或数据管道时，我们发现HiveRunner能够在一些中等复杂的SQL中很好地实现测试编写，甚至是TDD。通过HiveRunner，可以将Hive SQL编写为可释放且经过测试的工件。

joi
试验

joi是一款针对JavaScript对象结构的描述语言与验证工具。我们看中它独立于任何Web应用框架的特点，因此我们的团队可以在不同的技术栈中使用同一套结构模型来描述JavaScript对象。你也可以使用配套库，为使用joi schema来验证请求的API生成Swagger文档。

Ktor
试验

Kotlin证明了其在移动应用程序开发之外的价值。在微服务构建以及软件从生产到发布的过程中，我们的团队体验到了Ktor的诸多好处。与其它支持Kotlin的Web框架相比，Ktor本身就是用Kotlin编写的，并运用了诸如协程等语言特性来支持异步非阻塞的实现。除了具有轻量级架构外，它还能灵活的与各种不同的日志、依赖注入和模板引擎工具结合使用，这让 Ktor成为了创建RESTful服务时一个有趣的选项。

Laconia
试验

Laconia是一个用JavaScript开发AWS Lambda函数的框架。随着对无服务器技术的关注和使用，人们所构建的应用程序也越来越复杂。Laconia是一个小型轻量级框架，完善了我们经常遇到的一些不完美之处。该框架使用依赖注入将应用程序代码与较低级别的AWS API分离开来，并为应用程序可以响应的不同事件提供适配器。在部署时，该框架也能很好地与无服务器框架搭配使用。我们喜欢小巧且简单的框架，Laconia正是如此。

Puppeteer
试验

与Cypress和TestCafe一样，Puppeteer也是备受我们团队推崇的一款Web UI测试工具。Puppeteer能够对无头浏览器进行细粒度控制，生成时间轴信息，以用于性能诊断等。我们的团队发现，相较其他基于WebDriver的同类工具，Puppeteer更加稳定、快速和灵活。

Reactor
试验

在往期雷达报告中，我们谈论过Reactor。在我们的许多项目中，该工具持续受到了关注。随着Spring生态系统开始采纳Reactor，它已成为Reactive Streams的主要实现方式。响应式系统具有更好的可扩展性和伸缩性，但是调试成本更高，学习曲线也更陡峭。对于可以接受这种折中的项目，Reactor是一个不错的选择。我们的某些项目在采纳了Reactor和其他的一些响应式技术栈后，扩展性得到了明显的提高。我们还通过R2DBC，获得了对RDBMS驱动程序的响应式支持，从而解决了响应式服务的一个缺陷。

Resilience4j
试验

Resilience4j是一个轻量级的容错库，其灵感来自于Netflix Hystrix。它的轻量级和模块化结构备受欢迎，可以引入特定模块来实现特定功能，例如熔断、限流、重试和隔离等。虽然服务网格也具有一些容错功能，但容错库仍然是我们系统的关键组件，用于实现更精细的领域特定的容错行为，以及为非容器化的服务进行容错。Hystrix进入维护模式后，Resilience4j成为了Java生态系统中的默认选择。它既可以用于

同步API，也可以用于响应式API。Resilience4j还可以通过附加模块，将指标发送给Dropwizard Metrics、Prometheus等。

Room
试验

Room是一个数据持久化库，用于在Android上访问SQLite。它支持使用最小限度的样板代码进行数据库访问，同时通过编译时SQL校验使数据库访问更加稳健。令我们开发人员感到满意的是，使用LiveData后，Room能够与可观察查询完整集成。Room是Android Jetpack组件之一，旨在简化Android应用开发。

Rust
试验

Rust最近一次在技术雷达中出现是2015年，自那以来，我们看到开发者对Rust的兴趣在逐渐地提升。我们的一些客户正在使用Rust语言，尤其在围绕基础设施工具方面的使用最为常见，而在高性能的嵌入式设备中也可以见到Rust的身影。不断完善的生态系统以及语言本身的改进推动了人们的兴趣提升。语言的改进方面，包括了直接的性能增强，也包括了直观表现力的提高，例如非词法作用域的更改。大多数重大变化都包含在去年12月发布的Rust 2018标准中。

WebFlux
试验

WebFlux是Spring Framework中的Reactive Streams实现。我们看到团队越来越多地使用反应式编程模型，工作在Spring生态系统上的团队对WebFlux的使用也与日俱增。WebFlux最适用于大型微服务生态系统，其高性能的请求是我们的主要关注点。它支持异步并发请求处理，同时避免了普通多线程处理中的复杂性。WebFlux使用Reactor作为其反应式库，可以通过Reactive Streams与其他反应式库互操作。它使用Netty作为其基础的高性能通信引擎。虽然我们鼓励使用Reactive Streams，但采纳该编程模型需要在思维模式上做一些重大转变。

Aeron
评估

Aeron是一个高效且可靠的点对点消息传输工具。它通过一些媒体驱动程序（包括HTTP、UDP和TCP）提供重复的持久性消息日志。它还支持消息流的持久化存储，以供日后重放。对于许多应用程序而言，使用Aeron可能显得大材小用，因为它在非常低的级别运行（从概念上讲属于OSI第4层），但其点对点设计和低（且可预测的）延迟时间在许多用例中非常有用。我们确实发现它在某些机器学习应用程序中的用武之地，并能够在事件驱动型架构中起到一定作用。有必要指出的是，这种不需要额外服务（如Apache Kafka）就能运行的替代消息传递协议是存在的。

Arrow
评估

Arrow是适用于Kotlin的函数式编程库，是由两个现有流行库（kategory和funKTionale）合并而成。虽然Kotlin为函数式编程提供了构建模块，但Arrow为应用程序开发人员准备了随时可用的高级抽象包。它提供数据类型、类型类、作用（Effects）、Optics和其他函数式编程模式，并且可以与流行库相集成。通过Arrow，现有库得到统一，这对避免该领域社区分裂大有帮助。

Chaos Toolkit
评估

Chaos Toolkit是本期雷达提及的一款混沌工程工具。这款工具包可用来描述和运行基础架构上的可重复性试验，以了解架构在发生故障时的恢复能力。我们有很多团队在使用自创的工具来做这些事情，所以我们很高兴看到这样一款专门用于这种实践的开源项目出现。该工具包现已为AWS、Azure Service Fabric和GCE等提供驱动，而且与构建工具很好地配合，以实现试验的自动化。请注意，混沌工程是一项非常强大的技术，最适用于弹性感知的系统，即为了应对故障而构建的系统。因此，我们推荐可以先在非生产环境中使用Chaos Toolkit。

语言&框架

Room是一个数据持久化库，用于在Android上访问SQLite。它支持使用最小限度的样板代码进行数据库访问，同时通过编译时SQL校验使数据库访问更加稳健。

(Room)

Dask包括一个轻量级、高性能的调度程序，可以从一台笔记本电脑扩展到一个计算机集群，并且可与NumPy、pandas和Scikit-learn结合使用。

(Dask)

语言&框架

fastai是一个开源Python库，能够简化对快速且准确的神经网络的训练。fastai是一个开源Python库，能够简化对快速且准确的神经网络的训练。

(fastai)

http4k是用纯Kotlin编写的HTTP工具箱，用于提供和消费HTTP服务。它非常简洁和优雅，也很重视可测试性

(http4k)

Dask
评估

数据科学家和数据工程师经常使用诸如pandas之类的库来执行即席(ad hoc)数据分析。这些库虽然既富有表现力又功能强大，但却存在一种严重限制，即：只能在单个CPU上运行，并且不能为大型数据集提供横向可扩展性。然而Dask包括一个轻量级、高性能的调度程序，可以从一台笔记本电脑扩展到一个计算机集群。并且Dask可与NumPy、pandas和Scikit-learn结合使用，因此其进一步评估前景非常可观。

Embark
评估

之前，我们曾推荐使用Truffle来开发去中心化应用(dapp)。同样，Embark也可以简化你的工作。Embark提供脚手架、构建、测试和调试等功能，并与IPFS等去中心化存储集成。通过其声明式配置，可以轻松管理智能合约配置、依赖项、软件工件和部署。Embark的交互式CLI仪表盘也给人留下了深刻的印象。我们不断看到有人使用Remix编写智能合约并手动部署其应用程序，缺乏自动化测试、源码控制管理或软件工件管理。我们想要通过宣传Truffle和Embark等工具来吸引人们对dapp工程实践的关注。

fastai
评估

fastai是一个开源Python库，能够简化对快速且准确的神经网络的训练。它基于PyTorch构建，已成为备受我们数据科学家欢迎的工具。fastai可简化模型训练中的难点，如预处理、使用少量代码加载数据。该库根据深度学习最佳实践构建而成，对计算机视觉、自然语言处理(NLP)等提供开箱即用的支持。创始人的动机是为深度学习创建易于使用的库，使之成为一个改进版的Keras。GCP、AWS和Azure很快便接纳了fastai，将其包含在机器学习的镜像中。fastai的创建者意识到Python在速度和安全方面的限制，已宣布接纳Swift作为深度学习的替代语言。我们将密切关注其进展。

http4k
评估

http4k是用纯Kotlin编写的HTTP工具箱，用于提供和消费HTTP服务。http4k背后的关键理念之一是，HTTP应用通过组合两个简单的函数 — HttpHandler和Filter — 来进行建模。它的灵感来自于Twitter的“服务器即函数”论文。它拥有仅依赖于Kotlin StdLib的超轻量的核心模块。除了优雅和简洁，我们还要表扬其对可测试性的重视 — 考虑到库中的实体不可变，应用中的路由和应用本身只是函数，它们非常容易测试。但值得注意的是，http4k尚不支持非阻塞调用或协程。

Immer
评估

随着单页JavaScript应用程序的复杂性提高，能够以可预测的方式管理状态变得越来越重要。不可变性可帮助确保我们的应用程序行为一致，但遗憾的是，JavaScript本身不具有创建不可变对象的能力。Immutable.js等库填补了这一缺陷，但带来了新的问题，因为现在应用中存在两种对象和数组，库的版本和原生JavaScript版本。Immer（德语，意为始终）是一个微型库，能够以更加便捷的方式处理不可变状态。它基于写入时复制机制，提供尽可能少的API，对原生的JavaScript对象和数组进行操作。这意味着可实现无缝数据访问，同时在为现有代码库引入不可变性时不需要大量的重构工作。

Karate
评估

由于我们认为测试是唯一真正重要的API规范，因此我们始终在寻找可能提供帮助的新工具。Karate是一个API测试框架，其特色在于，直接使用Gherkin来编写测试，无需依赖常用编程语言来实现测试行为。Karate是一个领域特定语言，用来描述基于HTTP的API测试。虽然该方法很有趣，可以为简单的测试创建非常易读的规范，但用于匹配和验证负载的专用语言可能会变得语法晦涩、难以理解。从长远来看，使用此风格编写的复杂测试是否将可读且可维护，仍有待观察。

Micronaut
评估

Micronaut是一个新的JVM框架，用于使用Java、Kotlin或Groovy构建微服务。它的独特之处在于内存占用空间小、启动时间短。这是由于它避免了依赖注入和代理生成时的运行时反射（这是传统框架的常见缺点），而改用在编译时执行依赖项注入的DI/AOP容器，。如此一来，不仅是对于标准的服务器端微服务，在物联网、Android应用程序和无服务函数等上下文中，该框架都十分具有吸引力。Micronaut使用Netty，并为响应式编程提供一流支持。它还包含许多对云原生友好的功能，如服务发现和熔断。Micronaut是JVM领域的一款极有前途的全栈框架，我们对此表示密切关注。

Next.js
评估

React.js颠覆了大多数人编写单页JavaScript应用程序的方式。一般来说，我们建议在整个应用程序生命周期中使用Create React App，因而不必手动配置设置、构建和包。但是，一些开发人员更喜欢使用那些初始默认设置体现了一套完善意见的工具。Next.js就是这样的单一框架，因而吸引了前端爱好者的广泛关注。Next.js简化了路由，默认在服务器端渲染，并简化了依赖关系和构建。我们很想看看它是否符合我们自己项目的期望。

Pose
评估

Pose是一个简单的类似于CSS的动画库，适用于React.js、React Native和Vue.js框架。它是一个声明式动作系统，兼具CSS语法的简洁性与JavaScript在动画和交互上的强大和灵活性。

react-testing-library
评估

随着JavaScript框架的变革速度放缓，我们的团队有更多时间来使用特定框架，并因此获得了更加深入的见解。通过对React和其主要测试框架Enzyme的观察，我们发现单元测试的一个令人担忧的趋势。单元测试与实现细节耦合得太紧密，而不是保证功能能够正常工作(因为关注的是浅层细节)。这些单元测试导致设计难以演进，且将很多职责上移到了测试金字塔中的功能测试中。这让我们又重新回顾了皮下测试（subcutaneous testing）的理念。此外，由于其设计，Enzyme也很难跟上React的发展。所有这一切都推动我们将react-testing-library作为测试React应用程序的新框架进行评估。

ReasonML
评估

ReasonML是一个很有意思的基于C风格语法的新OCaml方言默认编译到JavaScript。它由Facebook创建，可嵌入JavaScript片段和JSX模版，还提供了很好的React集成。它旨在给JavaScript开发者提供一个既易用，又类型安全，还可利用既有生态的函数式语言。

Taiko
评估

Taiko是一个node.js库，具有清晰简洁的API，可以帮助实现Chrome或Chromium浏览器自动化。尽管Web应用程序的结构在不断发展，你依然可以利用Taiko的智能选择器编写可靠的测试。在测试脚本中无需使用id、CSS或XPath选择器，也无需显式等待XHR请求。如果你想要一边探索功能一边开发测试，交互式REPL记录器会非常方便。尽管可以单独使用Taiko，但我们将其与Gauge结合也取得了成功。

Vapor
评估

我们十分支持多语言编程，但也意识到在某些情况下，专注于单个编程语言也是很有意义的。如果我们大力投入于Swift（很有可能是因为iOS开发），想要寻找一种技术来编写服务器端服务，请考虑一下Vapor，这是一个适用于Swift的现代化web框架，最近备受推崇。

Micronaut是一个新的JVM框架，用于使用Java、Kotlin或Groovy构建微服务。它的独特之处在于内存占用空间小、启动时间短。

(Micronaut)

Taiko是一个node.js库，具有清晰简洁的API，可以帮助实现Chrome或Chromium浏览器自动化。用Taiko编写的测试旨在拥有高度可读性和可维护性。

(Taiko)

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®

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6000人，分布于全球14个国家、拥有40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。

thoughtworks.com

ThoughtWorks®

thoughtworks.com/radar

#TWTechRadar