ThoughtWorks®

TECHNOLOGY RADAR *VOL.17*

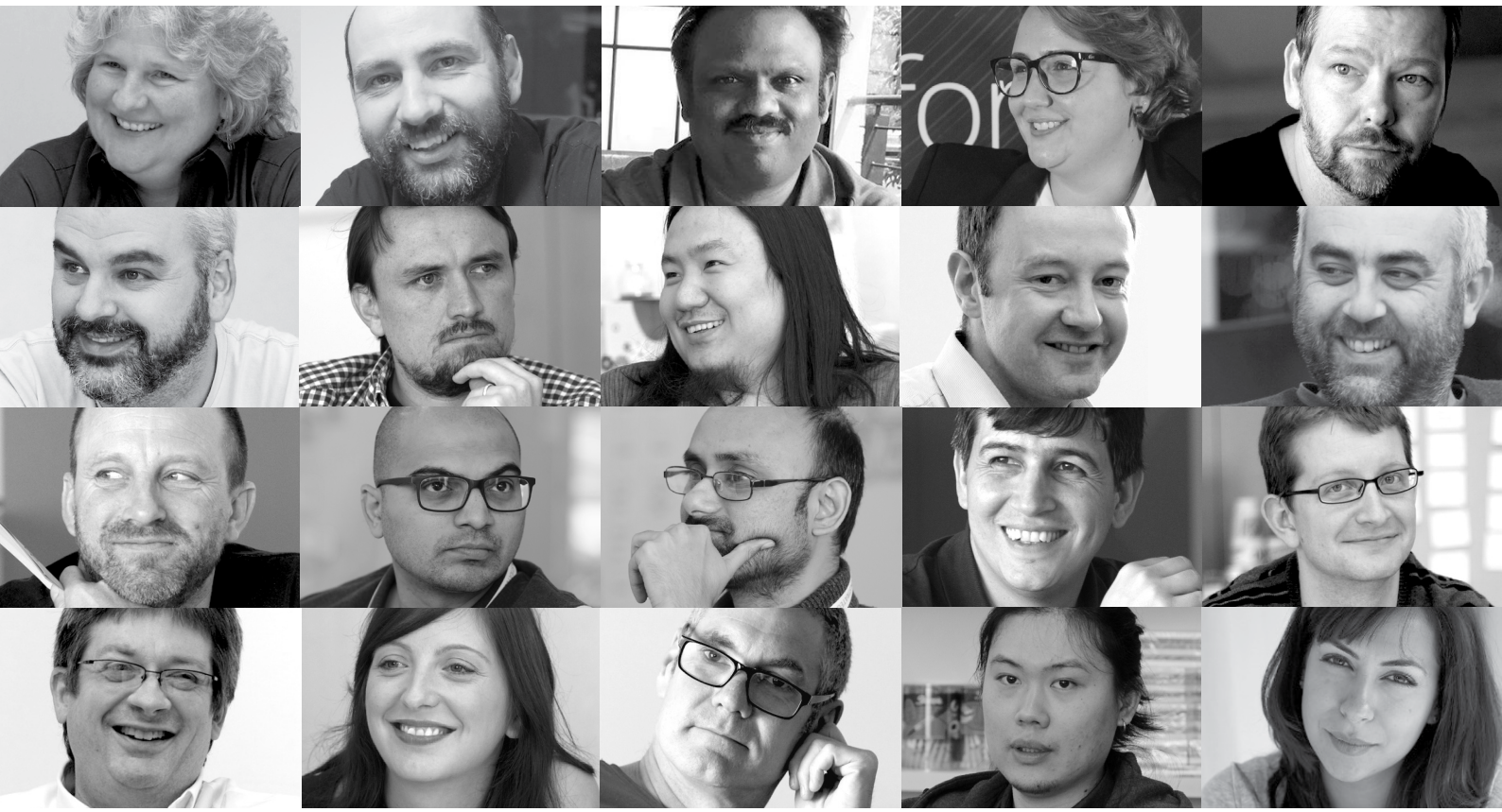
洞察构建未来的技术和趋势

thoughtworks.com/cn/radar

#TWTechRadar

贡献者

技术雷达由 ThoughtWorks 技术顾问委员会筹备，其人员组成为：



Rebecca Parsons (CTO) | Martin Fowler (首席科学家) | Bharani Subramaniam | Camilla Crispim | Erik Doernenburg
Evan Bottcher | Fausto de la Torre | 徐昊 | Ian Cartwright | James Lewis
Jonny LeRoy | Ketan Padegaonkar | Lakshminarasimhan Sudarshan | Marco Valtas | Mike Mason
Neal Ford | Rachel Laycock | Scott Shaw | 刘尚奇 | Zhamak Dehghani

技术雷达中国区技术咨询顾问组：

陈计节	曹宇鹏	陈莹莹	丁梦力	顾宇	和坚	黄博文	林帆	李好
林瀚贤	吕靖	林宁	刘先宁	蒋帆	邱俊涛	佟达	唐成	伍斌
王明涛	王晓雷	汪志成	魏喆	徐培	袁慎建	余丹妮	银大伟	杨璐
姚琪琳	鄢倩	于晓强	郑达夫	禚娴静	张凯峰	张羽辰	张渊	

最新动态

本期精彩集锦

崛起的中国开源软件市场

星星之火，已成燎原之势！在态度和政策发生转变之后，包括阿里巴巴和百度在内的众多大型中国企业正在积极发布开源框架、工具和平台。中国软件生态正伴随着经济扩张而加速成长。

从这个巨大而繁荣的软件市场向GitHub等开源网站发布的开源项目的数量必将持续增多，质量也将持续提高。中国企业为何热衷于将他们的众多资产开源出来？与硅谷等其他活跃的软件市场一样，各个企业对开发人员的争夺十分激烈。

仅仅提升薪酬水平是不够的，让聪明的开发者一起在最前沿的开源软件上共事才能够持续激励他们，这是一个放之四海而皆准的通则。我们预期主要的开源创意会继续保持README 文件中先有中文版后有英文版的趋势。

容器编排首选KUBERNETES

Kubernetes及其在许多项目中逐渐增强的主导性推动了大量雷达条目的更新，以及更多的讨论。似乎软件开发生态系统正在 Kubernetes 及其相关工具的周边稳定发展，以解决有关部署、规模化和容器操作这些常见问题。

诸如 GKE，Kops和Sonobuoy这些雷达条目提供了托管平台服务和工具，以改善采用和运行 Kubernetes 的整体体验。事实上，它具备用一个调度单元来运行多个容器的能力，可以让服务啮合（service mesh）和能够支持端点安全的sidecar得以实现。

Kubernetes已经成为容器的默认操作系统——许多云提供商已经利用其开放的模块化架构来采用和运行Kubernetes，而它的工具则可以利用其自身开放的API来访问诸如负载、集群、配置和存储等功能。

我们看到更多的产品正在把Kubernetes作为一个生态系统来使用，使其成为继微服务和容器之后的下一个抽象层次。更多迹象表明，尽管面临分布式系统固有的复杂性，开发人员仍然可以成功地驾驭现代的架构风格。

成为新常态的云技术

本期技术雷达讨论中的另一个普遍性话题，无疑是近期的“多云”天气。 随着云提供商的技术能力越来越强大，且可以提供同样好用的功能，公有云正在成为许多组织中新的默认选择。

当启动新项目时，许多公司已经不再问“为什么放在云端？”，而是问“为什么不放在云端？”。诚然，某些类型的软件仍然要在公司内部私有地部署，但随着价格的下降和功能的扩展，云原生（cloud-native）开发的可行性越来越高。

尽管主要的云解决方案提供商提供的基本功能都很相似，但它们也都提供了一些独特的产品特性，以针对特定类型的解决方案来实现差异化。因此，我们看到一些公司通过“多云”（Polycloud）策略来同时使用几个不同的云提供商，从中分别挑选最能满足其客户需求的平台专业能力。

各方对区块链的信任稳步增强

尽管加密货币市场仍然处于混沌状态，我们的许多客户已经开始尝试利用基于区块链的解决方案来构建分布式账本和智能合约。雷达中的一些条目展示了区块链相关技术运用的成熟度，它们使用各种新技术和编程语言并以一些有趣的方式来实现智能合约。

区块链解决了“分布式信任”与“共享且不可篡改的账本”这些古老的问题。如今，许多公司正致力于增强其用户对将区块链作为系统的底层实现机制的信心。许多行业存在着明显的“分布式信任”问题，我们期待区块链技术能持续找出解决这些问题的方法。

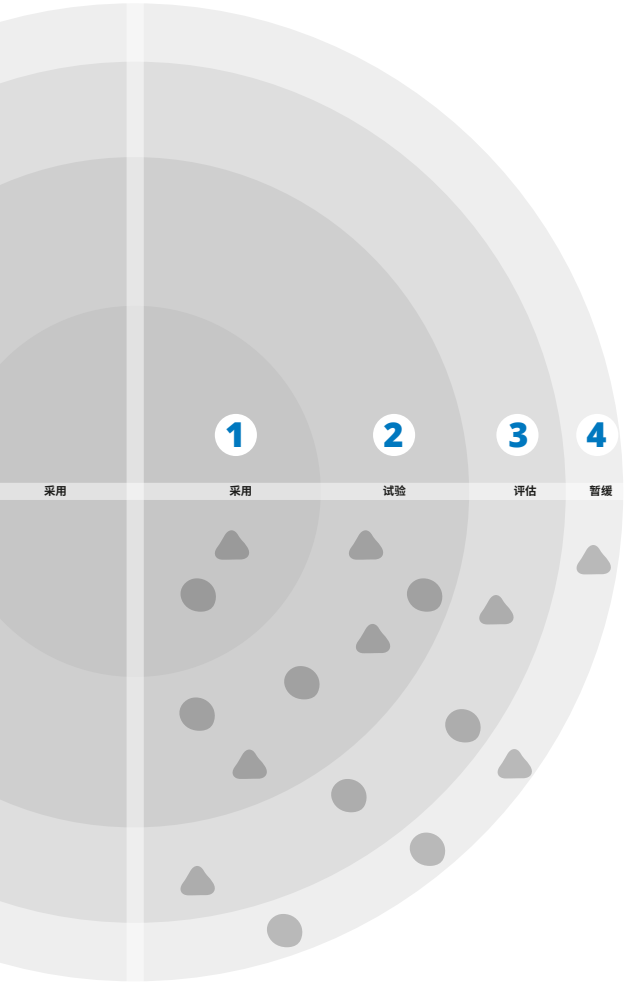
关于技术雷达

ThoughtWorks人酷爱技术。我们对技术进行构建、研究、测试、开源、记述，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(TAB)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。

这个雷达以独特的形式记录技术顾问委员会的讨论结果，为从开发人员到CTO在内的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。

这个雷达是图形性质的，把各种技术项目归类为技术、工具、平台和语言及框架，如果某个条目可以出现在多个象限，我们选择看起来最合适的象限。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。

要了解关于雷达的更多背景，请点击：https://www.thoughtworks.com/radar/faq



雷达一览

1 采用

我们强烈主张业界采用这些技术。如果适合我们的项目，我们就毫不犹豫地使用。

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

值得研究一番的技术，以确认它将对你产生何种影响。你应该投入一些精力来确定它是否会对你所在的组织产生影响。

4 暂缓

别用这项技术启动任何新项目。在已有项目上使用它没有坏处，但是想在新开发的项目上使用这个技术的话需要三思而行。

三角形图标 三角形图标表示新出现或位置发生过显著变化的条目

圆形图标 圆形表示没有变化的条目

我们感兴趣的技术实在太多，远不是如此大小的文档能合理容纳的。如果一个图标在一年内的两期技术雷达上都没有移动，我们就把它略去。以减少混乱，并为新条目腾出空间，但并不表示我们不再关心它。

THE RADAR

技术

采用
1. Lightweight Architecture Decision Records

试验
2. Applying product management to internal platforms *NEW*
3. Architectural fitness function *NEW*
4. Autonomous bubble pattern *NEW*
5. Chaos Engineering *NEW*
6. Decoupling secret management from source code
7. DesignOps *NEW*
8. Legacy in a box
9. Micro frontends
10. Pipelines for infrastructure as code *NEW*
11. Serverless architecture
12. TDD'ing containers *NEW*

评估
13. Algorithmic IT operations *NEW*
14. Ethereum for decentralized applications *NEW*
15. Event streaming as the source of truth *NEW*
16. Platform engineering product teams
17. Polycloud *NEW*
18. Service mesh *NEW*
19. Sidecars for endpoint security *NEW*
20. The three Rs of security *NEW*

暂缓
21. A single CI instance for all teams
22. CI theatre
23. Enterprise-wide integration test environments
24. Recreating ESB antipatterns with Kafka *NEW*
25. Spec-based codegen

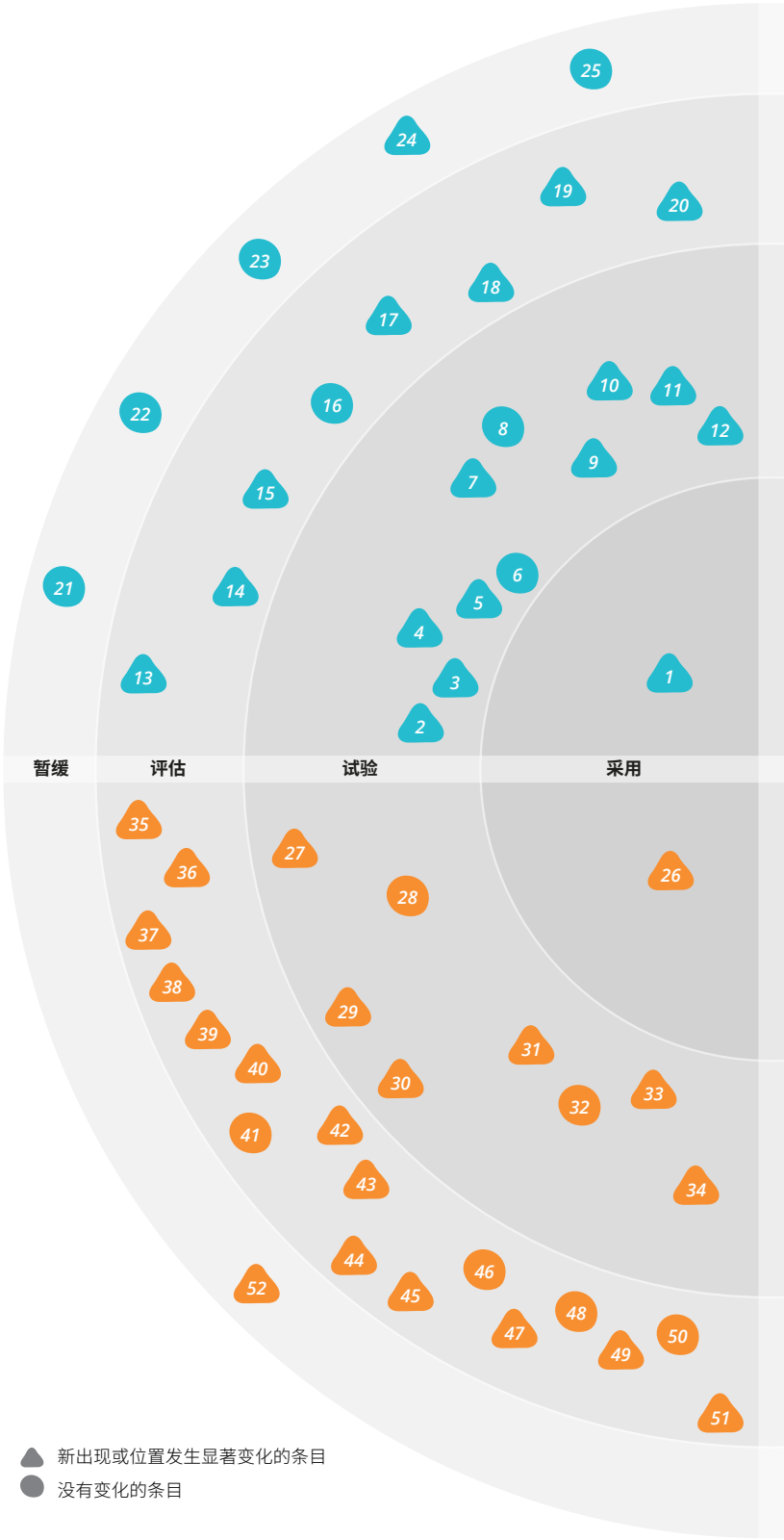
平台

采用
26. Kubernetes

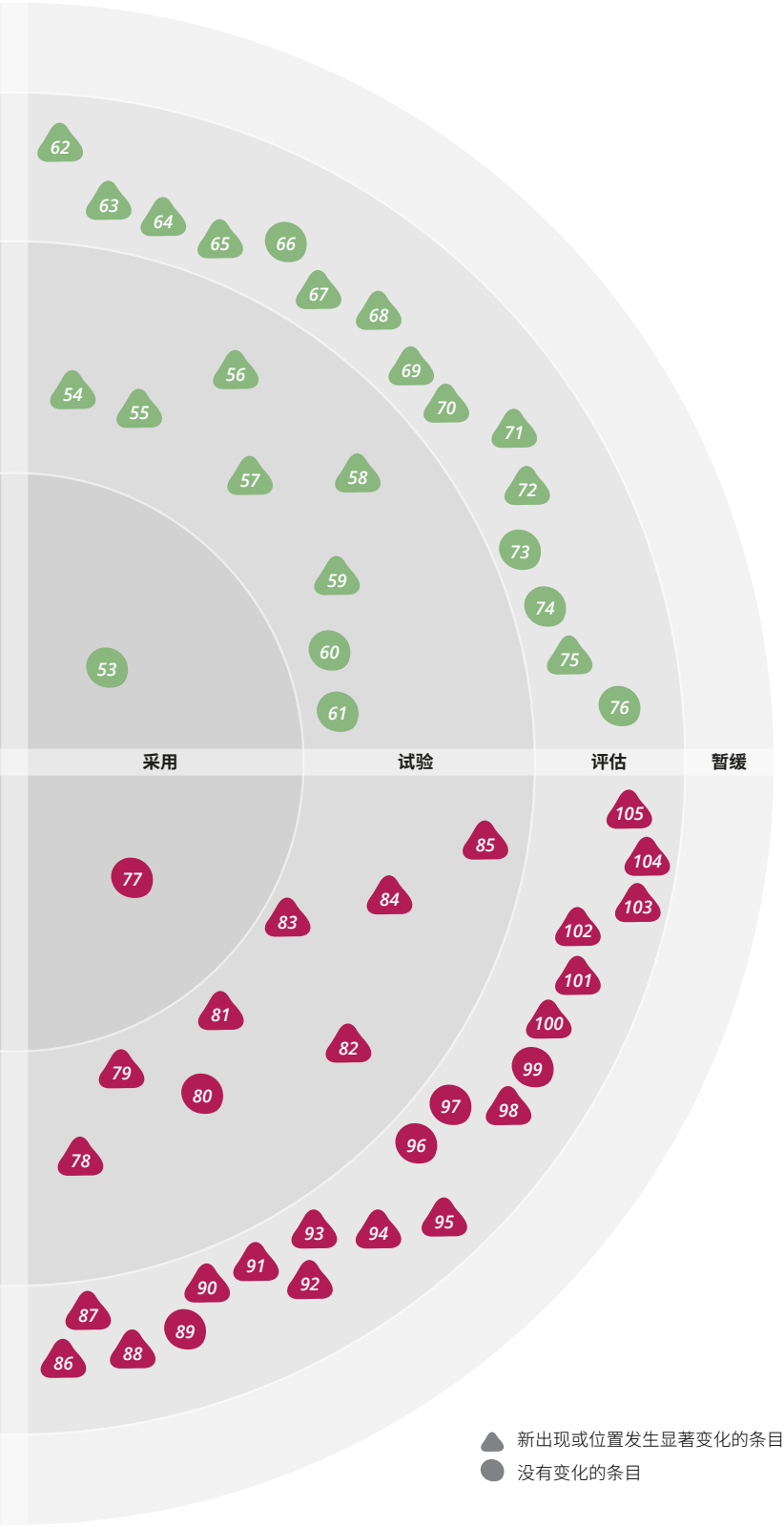
试验
27. .NET Core
28. AWS Device Farm
29. Flood IO *NEW*
30. Google Cloud Platform *NEW*
31. Keycloak
32. OpenTracing
33. Unity beyond gaming
34. WeChat *NEW*

评估
35. Azure Service Fabric *NEW*
36. Cloud Spanner *NEW*
37. Corda *NEW*
38. Cosmos DB *NEW*
39. DialogFlow
40. GKE *NEW*
41. Hyperledger
42. Kafka Streams
43. Language Server Protocol *NEW*
44. LoRaWAN *NEW*
45. MapD *NEW*
46. Mosquitto
47. Netlify *NEW*
48. PlatformIO
49. TensorFlow Serving *NEW*
50. Voice platforms
51. Windows Containers *NEW*

暂缓
52. Overambitious API gateways



THE RADAR



工具

采用
53. fastlane

试验
54. Buildkite *NEW*
55. CircleCI *NEW*
56. gopass *NEW*
57. Headless Chrome for front-end test *NEW*
58. jsoniter *NEW*
59. Prometheus
60. Scikit-learn
61. Serverless Framework

评估
62. Apex *NEW*
63. assertj-swagger *NEW*
64. Cypress *NEW*
65. Flow *NEW*
66. InSpec
67. Jupyter *NEW*
68. Kong API Gateway *NEW*
69. kops *NEW*
70. Lighthouse *NEW*
71. Rendertron *NEW*
72. Sonobuoy *NEW*
73. spaCy
74. Spinnaker
75. Spring Cloud Contract *NEW*
76. Yarn

暂缓

语言&框架

采用
77. Python 3

试验
78. Angular
79. AssertJ *NEW*
80. Avro
81. CSS Grid Layout *NEW*
82. CSS Modules *NEW*
83. Jest *NEW*
84. Kotlin
85. Spring Cloud

评估
86. Android Architecture Components *NEW*
87. ARKit/ARCore *NEW*
88. Atlas and BeeHive *NEW*
89. Caffe
90. Clara rules *NEW*
91. CSS-in-JS *NEW*
92. Digdag *NEW*
93. Druid *NEW*
94. ECharts *NEW*
95. Gobot *NEW*
96. Instana
97. Keras
98. LeakCanary *NEW*
99. PostCSS
100. PyTorch *NEW*
101. single-spa *NEW*
102. Solidity *NEW*
103. TensorFlow Mobile *NEW*
104. Truffle *NEW*
105. Weex *NEW*

暂缓

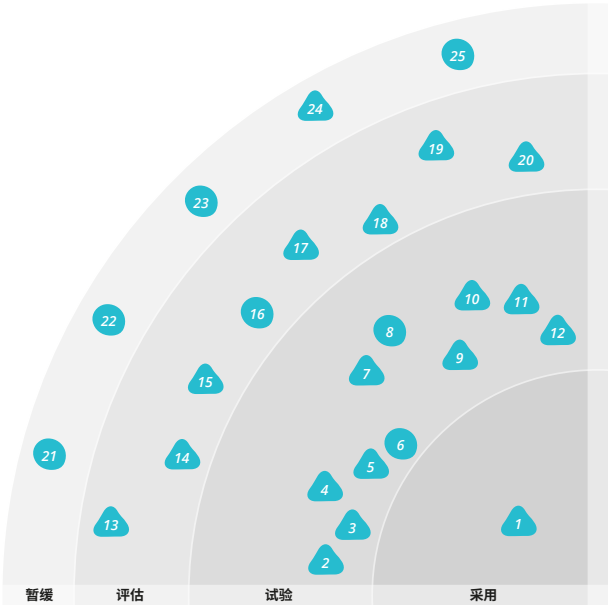
技术

采用
1. Lightweight Architecture Decision Records

试验
2. Applying product management to internal platforms *NEW*
3. Architectural fitness function *NEW*
4. Autonomous bubble pattern *NEW*
5. Chaos Engineering *NEW*
6. Decoupling secret management from source code
7. DesignOps *NEW*
8. Legacy in a box
9. Micro frontends
10. Pipelines for infrastructure as code *NEW*
11. Serverless architecture
12. TDD'ing containers *NEW*

评估
13. Algorithmic IT operations *NEW*
14. Ethereum for decentralized applications *NEW*
15. Event streaming as the source of truth *NEW*
16. Platform engineering product teams
17. Polycloud *NEW*
18. Service mesh *NEW*
19. Sidecars for endpoint security *NEW*
20. The three Rs of security *NEW*

暂缓
21. A single CI instance for all teams
22. CI theatre
23. Enterprise-wide integration test environments
24. Recreating ESB antipatterns with Kafka *NEW*
25. Spec-based codegen



很多文档都可以被高度可读的代码和测试取代。然而，对于演进式架构来说，记录某些设计决策非常重要，这不仅有利于未来的团队成员理解，也有利于外部监督。**轻量级架构决策记录**是一种用于捕获重要的架构决策及其上下文和结果的技术。我们建议将这些详细信息进行版本化，而不是wiki或网站，这样所记录的内容就可以和代码保持同步。对于大多数项目，我们没有理由不采用这种技术。

过去12个月里，我们注意到对数字化平台这个主题的关注发生了急剧的增长。希望快速有效推出新数字解决方案的公司，正在建立内部平台，为交付团队提供自助服务，从而访问那些构建和运营自己的解决方案所必需的业务API、工具、知识和支持。我们发现，当这些平台得到跟外部产品同等的重视时，它们的效率是最高的。**将产品管理思维应用于内部平台**，意味着与内部消费者（开发人员）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。一些企业甚至为内部平台创建了品牌标识，并向同事推销平台的优势。平台产品经理将负责平台的质量、收集使用指标，并持续改进平台。将平台作为产品来处理，有助于创造一个蓬勃发展的生态系统，避免构建另一个停滞不前的、未充分利用的面向服务架构。

适应度函数借鉴自进化计算，被用来衡量方案对满足目标的适合度。
（架构适应度函数）

适应度函数借鉴自进化计算，被用来衡量方案对满足目标的适合度。当定义演进式算法时，算法设计者会寻求更优解，而适应度函数则定义了在此上下文中“更优”的含义。《演进式架构》一书定义了**架构适应度函数**的概念，为衡量架构特征提供了一个客观全面的方法，包括已有的验证标准，比如单元测试、业务指标、监控等等。我们相信架构师能够验证并维持一套自动化的可持续的架构标准，这是演进式架构的关键。

CI和CD工具可以用来测试服务配置、服务镜像构建、环境准备以及环境的集成。

（为基础设施即代码使用流水线）

许多与我们合作的组织正在努力使用现代化工程方法构建新的能力和产品功能，与此同时，还必须与遗留系统引起的长尾效应共存。根据我们的经验，一个比较老旧的策略在这些场景中越来越有帮助，即Eric Evans’ s的**自治气泡模式**。这种方法为新应用程序的开发工作创建一个全新的环境，以避免遗留系统的纠缠。它并不仅仅使用了反腐层，这个新的气泡上下文还能完全控制其支持数据，然后通过异步的方式与遗留系统保持一致。要保护气泡的边界以及保持边界两侧的一致性需要花费一些工作，但是由此产生的自治性和开发摩擦力的减少是迈向现代化未来架构勇敢的第一步。

在早期的技术雷达中，我们讨论了Netflix的（Chaos Monkey）。混沌猴可以随机终止生产系统中的运行实例，并对结果进行度量，从而帮助验证系统在运行时对生产中断的应对能力。今天，人们有了一个新兴术语来描述这一技术的广泛应用：**混沌工程**。在生产环境的分布式系统中运行这些试验，可以帮助我们建立系统在动荡环境下依旧能够按预期工作的信心。如果想要更好地理解这个技术方向，请参阅混沌工程原则。

受DevOps运动的启发，**DESIGNOPS**是一种文化上的转变，同时包含了一系列的实践。DesignOps可以帮助组织不断地重新设计产品，而又无需在质量、服务连贯性和团队的自主性上妥协。DesignOps提倡创建并演进设计的基础设施，最大限度降低创造新的UI概念及其变体的工作量，并与最终用户建立快速且可靠的反馈机制。通过使用诸如Storybook这样促进紧密协作的工具，对前期分析和规范交接的需求可以被降至最低。使用DesignOps，设计正在从一种具体的实践演变成每个人工作的一部分。

我们已经从引入微服务架构中获得了明显的好处，微服务架构可以让团队裁剪出独立部署的交付物以及可维护的服务。不幸的是，我们还看到许多团队在后端服务之上创建了前端单体——一个单一，庞大和杂乱无绪的浏览器应用。我们首选的（经过验证的）方法是将基于浏览器的代码拆分成**微前端**。在这种方法中，Web应用程序被分解为多个特性，每个特性都由不同的前后端团队拥有。这确保每个特性都独立于其他特性开发，测试和部署。这样可以使用多种技术来重新组合特性——有时候是页面，有时候是组件——最终整合成一个内聚的用户体验。

使用持续交付流水线来编排软件的发布流程，已经成为了主流概念。不过，对基础设施代码进行自动化测试还没有被广泛理解。CI和CD工具可以用来测试服务配置（如Chef的cookbook，Puppet的模块，Ansible的playbook）、服务镜像构建（如Packer）、环境准备（Terraform，CloudFormation等）以及环境的集成。对**基础设施即代码使用流水线**可以让错误在进入运维环境，包括开发和测试环境之前就被发现。这些流水线还能确保基础设施工具能始终如一地运行在CI/CD的Agent上，而不是在特定的工作站上。挑战仍然存在，比如与容器和虚拟机相关的更长的反馈周期，但我们认为这是一个有价值的技术。

无服务器架构迅速得到了需要部署云端应用的组织的认可，并且有着大量可供选择的部署方式。即便是相对传统和保守的组织，也在使用一部分无服务器技术。虽然可以使用的合适的模式仍在不断涌现，但大多数时候我们的讨论都会走向函数即服务(Functions as a Service)（例如 AWS Lambda，Google Cloud Functions，Azure Functions）。部署无服务器函数毫无疑问能够减少大量传统方式特有的，涉及服务器和操作系统配置和编排的工作量。然而serverless也并不是百试不爽的万金油。当前这个阶段，因为一些特别的需求，你必须做好能回退至容器化，甚至是实例化部署的准备。与此同时，无服务器架构的其他组件，比如后端即服务（Backend as a Service），几乎成为了默认的选择。

因为测试驱动开发自身的优势，许多开发团队会在编写代码时采用这一实践。也有一些团队开始使用容器来打包和部署软件，而通过自动化脚本来构建容器已经是被广为接受的实践。但迄今为止，我们很少看到有团队能将这两种趋势结合，通过测试来驱动容器脚本的编写。借助ServerSpec和Goss这样的框架，你可以为独立的或编排的容器呈现预期的功能，并得到快速反馈。这意味着我们可以寻求用**TDD开发容器脚本**。我们在这方面得到的初步体验是十分积极的。

近年来，IT运维所收集到的数据量一直在增加。例如，微服务的迅速发展意味着更多的应用程序正在生成自己的操作数据；而像Splunk，Prometheus或ELK堆栈这样的工具让数据存储和后续处理变得更容易，从而获得运营洞见。毋庸置疑，随着机器学习工具的普及，运营人员已经开始将统计模型和经过训练的分类算法纳入其工具包中。虽然这些算法并不新颖，而且人们已经进行了各种尝试来自动化服务管理，但是在了解机器和人员如何协作以便早期识别异常和确定故障来源这个方向，我们还是先行者。尽管**基于算法的IT运维（Algorithmic IT Operations）**有过度炒作的风险，但毫无疑问，机器学习算法的稳步提升将改变未来的数据中心运营中人类所起到的作用。

在银行、数字货币、供应链透明化等多个金融科技领域，区块链技术已经被炒作成“灵丹妙药”。我们曾经在过去的雷达中推荐过带有智能合约功能的Ethereum，而最近已经看到**Ethereum在去中心化应用**的其他领域中得到更多的发展。尽管这一技术非常年轻，我们仍然鼓励你在加密货币和银行之外的领域，使用它构建去中心化应用。

> 在银行、数字货币、供应链透明化等多个金融科技领域，区块链技术已经被炒作成“灵丹妙药”。
>
> （Ethereum去中心化应用）

随着事件流（event streaming）平台（如Apache Kafka）的兴起，很多人将它们视为消息队列的高级形态，仅用来传输事件。即便按这种方式使用，事件流仍然具有传统消息队列无法比拟的优势。然而我们更感兴趣的是，人们如何通过把平台（特别是Kafka）作为主要存储，把数据保存为不可变事件，从而**将事件流作为正确数据之源**。例如，以Event Sourcing方式设计的服务，可以使用Kafka作为事件存储工具（event store），其他服务可以消费这些事件。这一技术能够减少本地持久化和集成之间的重复工作。

主要的云服务提供商(Amazon、Microsoft和Google)正陷入一场激烈的竞争中，以保持核心能力的均势，虽然他们的产品只是略有差异。这导致一些组织采用**多云（POLYCLOUD）策略**，而不是与一个提供商“全面”合作，他们正在以最佳组合的方式，将不同类型的工作负载交由不同的供应商。例如，这可能包括将标准服务放在AWS上，使用Google进行机器学习，把采用SQL Server的.NET程序部署在Azure，或者可能使用Ethereum Consortium的区块链解决方案。这并不等同于致力于供应商间可移植性的“跨云策略”，后者价格昂贵且会导致迎合大众的想法。多云策略则专注于使用每个云能提供的最好的服务。

服务啮合(service mesh)在服务发现、安全、跟踪、监控与故障处理方面提供了一致性，且不需要像API网关或ESB这样的共享资产。

（服务啮合）

现在越来越多的大型组织在向更加自组织的团队结构转型，这些团队拥有并运营自己的微服务，但他们如何在不依赖集中式托管的基础架构下，确保服务之间必要的一致性与兼容性呢？为了确保服务之间的有效协作，即使是自组织的微服务也需要与一些组织标准对齐。**服务啮合(SERVICE MESH)**在服务发现、安全、跟踪、监控与故障处理方面提供了一致性，且不需要像API网关或ESB这样的共享资产。服务啮合的一个典型实现包含轻量级反向代理进程，这些进程可能伴随每个服务进程一起被部署在单独的容器中。反向代理会和服务注册表、身份提供者和日志聚合器等进行通信。通过该代理的共享实现（而非共享的运行时实例），我们可以获得服务的互操作性和可观测性。一段时间以来，我们一直主张去中心化的微服务管理方法，也很高兴看到服务啮合这种一致性模式的出现。随着linkerd和Istio等开源项目的成熟，服务啮合的实现将更加容易。

微服务架构中，众多服务将其资产和功能都通过API暴露出来，同时也扩大了系统的被攻击面。因此一个零信任——“永不信任，始终验证”的安全架构势在必行。然而，由于服务代码复杂性的增加以及在多语言环境中缺少库和语言的支持，服务之间的安全控制往往会被忽略。为了解决这个复杂性，我们已经看到将安全性委托给进程外Sidecar的做法，Sidecar是一个独立的进程或一个容器，它与每个服务一起部署和调度，并共享相同的执行上下文、主机和身份。Sidecar实现了安全功能，如对服务间的通信作透明加密、TLS终止，以及对调用方服务或最终用户的鉴权机制。在实现自己的**用于端点安全的SIDECAR**之前，我们推荐你先研究一下Istio、linkerd或者Envoy。

传统的企业安全方法往往强调锁定事物并减慢变革的步伐。但众所周知，攻击者对系统实施攻击的时间越长，造成的损失也就越大。3Rs企业安全：轮换、修复、重建，利用基础设施自动化和持续交付来消除攻击机会。轮换凭证，一旦有可用的补丁就立即应用补丁，并且在几分钟或几小时内完成从已知的安全状态重建系统，这会使攻击者更难获得立足点。随着现代原生云架构的出现，**3RS安全**技术变得可行。当应用程序部署为容器，并通过完全自动化的流水线进行构建和测试时，安全补丁只不过是又一次通过一个点击，就可以通过流水线发布的小版本而已。当然，为了保持良好的分布式系统实践，开发人员需要设计应用以适应意外的服务器中断。这就和在环境中实施混沌猴所造成的影响相似。

Kafka已经是流行的消息解决方案，同时，Kafka Streams也站到了流式架构的最前沿。不幸的是，随着人们将Kafka作为数据和应用平台的核心，我们看到一些组织没有将Kafka的生态组件（如连接器、流处理器等）按产品和服务团队划分，而是将它们集中管制，这**用KAFKA重现了ESB反模式**。这一反模式具有很严重的问题，过多的逻辑、编排、转换被插入到集中管理的ESB中，使得系统严重依赖于一支中心化团队。我们特此提出这个问题，希望劝阻这种反模式的更多实现。

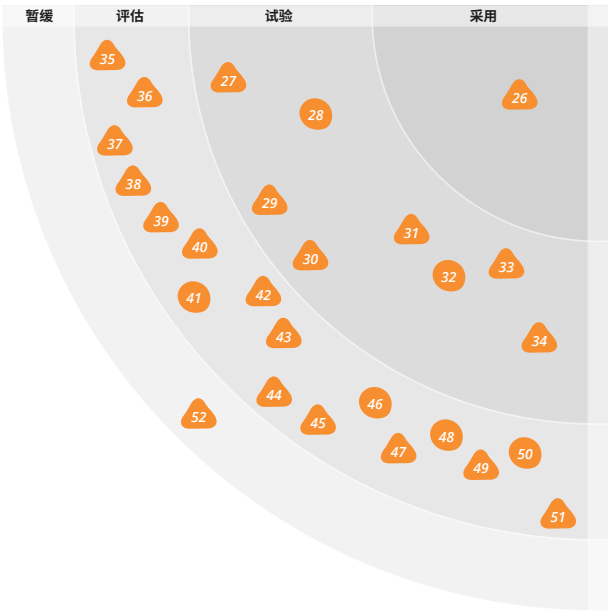
平台

采用
26. Kubernetes

试验
27. .NET Core
28. AWS Device Farm
29. Flood IO *NEW*
30. Google Cloud Platform *NEW*
31. Keycloak
32. OpenTracing
33. Unity beyond gaming
34. WeChat *NEW*

评估
35. Azure Service Fabric *NEW*
36. Cloud Spanner *NEW*
37. Corda *NEW*
38. Cosmos DB *NEW*
39. DialogFlow
40. GKE *NEW*
41. Hyperledger
42. Kafka Streams
43. Language Server Protocol *NEW*
44. LoRaWAN *NEW*
45. MapD *NEW*
46. Mosquitto
47. Netlify *NEW*
48. PlatformIO
49. TensorFlow Serving *NEW*
50. Voice platforms
51. Windows Containers *NEW*

暂缓
52. Overambitious API gateways



自我们上次在技术雷达中提到**KUBERNETES**至今，它已经成为我们大部分客户将容器部署到服务器集群的默认解决方案。而能替代它的其他产品不但没有获得如此的客户认同度，甚至在某些场景中，我们的客户会将他们的“引擎”都更换成 Kubernetes。Kubernetes已经成为主流公有云平台上的首选容器编排平台。这些主流公有云平台包括微软的 Azure 容器服务以及 Google Cloud（参见GKE）。此外市面上还有很多好用的产品，来不断丰富快速扩大的Kubernetes生态圈。与此同时，那些试图用一层抽象将Kubernetes隐藏起来的平台尚未成功地证明自己的价值。

作为一个开源的跨平台软件开发框架，**.NET CORE**被越来越多地运用到实际项目中。该框架令 .NET 应用能在 Windows、macOS 以及 Linux 系统上进行开发和部署。.NET Standard 2.0 的发布增加了跨多个 .NET 平台的标准 API 的数量，这使得往.NET Core迁移的路径变得更为清晰。有关.NET Core对其上类库的支持性问题正在逐渐减少。一流的跨平台工具已经涌现出来，用于在非 Windows 平台上进行高效的开发工作。运用Docker镜像，能让.NET Core 服务可以轻松地集成到容器环境中。其社区发展的积极方向以及来自我们实际项目的反馈，都表明.NET Core现在已经可以广泛地运用了。

随着Gatling和Locust等工具的日益成熟，压力测试变得越来越容易。与此同时，弹性云平台基础设施可以模拟大量客户端实例来进行压力测试。我们欣喜地看到像 Flood IO 这样的云平台能越来越深入地应用此类技术。**FLOOD IO**是一个基于SaaS的压力测试服务。它可以用来向数百台云端服务器分发测试脚本并在其上执行。我们的团队发现，通过重用现有的Gatling测试脚本能很简单地将性能测试迁移到 Flood IO。

随着**GOOGLE CLOUD PLATFORM (GCP)** 在可用地理区域和服务成熟度方面的扩展，全球的客户在规划云技术策略时可以认真考虑这个平台了。与其主要竞争对手Amazon Web Services相比，在某些领域， GCP 所具备的功能已经能与之相媲美。而在其他领域又不失特色——尤其是在可访问的机器学习平台、数据工程工具和可行的 “Kubernetes 即服务解决方案”（GKE）这些方面。在实践中，我们的团队对GCP工具和API良好的开发者体验也赞赏有嘉。

> 随着Google Cloud Platform在可用地理区域和服务成熟度方面的扩展，全球的客户在规划云技术策略时可以认真考虑这个平台了。
> （Google Cloud Platform）

在微服务或任何其他分布式架构中，最常见的一个需求是通过身份验证和授权功能来保护服务或 API。 这正是 Keycloak 所解决的问题。**KEYCLOAK**是一个开源的身份和访问管理解决方案，它让保障应用程序和微服务的安全变得如此简便，以至于几乎不需要编写什么代码。它提供了单点登录、社交网络登录和一些开箱即用的标准协议——如 OpenID Connect、OAuth 2.0 和 SAML。我们的团队一直在使用这个工具，并计划继续使用。不过这个平台在安装时需要做一些工作。由于在初始化和运行时需要通过 API 对其进行配置，因而必须编写脚本以确保部署是可重复的。

在之前的雷达中，我们提到由于Unity在一个成熟平台上提供了一些抽象和工具，因此它已经成为VR和AR应用程序开发的首选平台。与它的主要替代品Unreal Engine相比，Unity更容易访问。随着它最近推出的针对iOS平台的 ARKit 和针对安卓平台的 ARCore，这两个主要的移动平台都已拥有强大的原生SDK，用来构建增强现实应用。但是，我们觉得很多团队，特别是那些在构建游戏方面没有丰富经验的团队，都能从利用类似Unity这样的抽象中受益。这就是为什么我们要提出**超越游戏的UNITY**。这使得不熟悉上述开发技术的开发人员可以专注于这个SDK，来进行相关开发。它还为多设备提供了解决方案，特别是在原生SDK不支持的安卓端。

常与WhatsApp被相提并论的**微信**，在中国正在成为名副其实的商业平台。很多人可能还不知道，微信还是最流行的线上支付平台之一。借助微信内置的内容和会员管理系统，一些小型企业现已完全依赖微信开展其业务。大型组织可以通过微信的一些功能把内部系统对接给员工使用。作为覆盖七成以上中国人的平台，微信是每一个想开辟中国市场的企业都需要考虑的重要商业因素。

AZURE SERVICE FABRIC是为微服务和容器打造的分布式系统平台。它不仅可以与诸如Kubernetes之类的容器编排工具相媲美，还可以支持老式的服务。它的使用方式花样繁多，既可以支持用指定编程语言编写的简单服务，也可以支持 Docker 容器，还可以支持基于 SDK 开发的各种服务。自几年之前发布以来，它不断增加更多功能，包括提供对 Linux 容器的支持。尽管 Kubernetes 已成为容器编排工具的主角，但 Service Fabric 可以作为 .NET 应用程序的首选。我们正在 ThoughtWorks 的一些项目中使用这个平台，迄今为止感觉不错。

CLOUD SPANNER是一个完全托管的关系型数据库服务。它在提供高可用性和强大的一致性的同时又不会对延迟做出妥协。Google 在一个名为 Spanner 的全球分布式数据库上投入了大量时间之后，最近以 Cloud Spanner 的名称将这个服务对外发布。 这个平台可以将数据库实例在全球范围从单个节点规模化到数千个节点，而不必担心数据一致性问题。 Cloud Spanner 通过高可用的分布式时钟 TrueTime，为读取和快照功能提供了强大的一致性。从 Cloud Spanner 读取数据时可以使用标准 SQL，但是在做写入操作时必须使用其提供的 RPC API。 尽管并不是所有的服务都需要全球范围规模的分布式数据库，但 Cloud Spanner 的对外发布极大地改变了我们对数据库的认知。其设计正在影响像 CockroachDb 这样的开源产品。

在经过了彻底探索之后，区块链领域的重要参与者R3意识到区块链并不契合他们的目的，所以他们创造了**CORDA**。Corda是专注于金融领域的分布式账本技术（distributed ledger technology, DLT）平台。 R3具有非常明确的价值主张，并且知道他们的问题需要务实的技术方法。 这和我们的经验相符——由于采矿成本较大和运营效率低下，对于某些商业案例，目前的区块链解决方案可能不是合理的选择。尽管我们目前在Corda上的开发体验并不非常流畅，而且v1.0发布后其API并不稳定，我们还是期望看到 DLT 领域能进一步成熟。

COSMOS DB是微软于年初发布的全球分布式与多模型数据库服务。虽然大多数新型 NoSQL 数据库都提供可调节的一致性，但Cosmos DB 却将一致性作为首要特性予以支持。它提供五种不同的一致性模型。值得强调的是，它还支持多种数据模型——键值、文档、列族和图——所有这些数据模型都映射到其内部数据模型，即原子记录序列（atom-record-sequence, ARS）。Cosmos DB 有趣的一个特点是能针对其延迟、吞吐、一致性和可用性来提供服务级别协议（service level agreement, SLA）。其适用性广的特点，给其他云厂商设置了一个很高的追赶标准。

随着近来聊天机器人与语音平台的爆发，涌现出一批工具和平台——它们能够提供一些服务，从文字中挖掘意图，并管理会话流，以供人使用。Google所收购的**DIALOGFLOW**（原名为API.ai），就是一种这样的”自然语言理解即服务”的平台。它在该领域中与Facebook的wit.ai以及Amazon Lex等平台展开了竞争。

尽管以Kubernetes作为容器编排平台正成为软件开发生态的主流，但从运维角度看，运行 Kubernetes 集群仍然很复杂。**GKE**（Google Container Engine）是一个托管的Kubernetes解决方案，用来部署容器化应用程序。它能降低运行和维护Kubernets集群的运维成本。我们的团队在使用GKE获得了良好的体验。平台能完成许多繁重的工作，例如安装安全补丁，监控和自动修复节点，以及管理多集群和多区域网络。根据我们的经验，Google采用了API优先的方式来开放平台功能，并使用了诸如用OAuth进行服务授权的行业标准。这些都能够改善开发人员的体验。尽管其开发团队已经尽力隔离底层变更对使用者的影响，但要意识到GKE仍在快速开发中。在过去一段时间里我们还是会时不时地受到变更所带来的影响。我们期待随着Terraform on GKE及类似工具的出现，“基础设施即代码”这一实践的成熟度会不断提高。

Language Servers将代码补全、调用分析和重构等能力提取为一种 API，从而让任何编辑器都能与编程语言的抽象语法树交互。
(语言服务器协议)

KAFKA STREAMS是一个用于构建流式应用的轻量级库。它的设计目标在于简化流式处理，让它像为异步服务设计的主流应用编程模型一样易于访问。当需要应用流式处理模型来解决问题，又不想陷入运行集群（通常会随着功能完备的流处理框架而引入）的复杂性时，它会是一个很好的选择。 新的功能包括在Kafka集群中的“恰好一次”（exactly once）流处理。实现方式是在Kafka生产者端引入幂等性，并且使用新的事务API跨多个分区实现原子写入。

大型 IDE 的威力很大程度上源于利用源代码分析出的抽象语法树（AST）来进一步分析和操作源代码的能力，比如代码补全，调用分析和重构。语言服务器将这种能力提取到单独的进程中，从而让任意文本编辑器都可以通过 API 来使用 AST。微软从他们的 OmniSharp 和 TypeScript 服务器项目中，提炼并引领了**语言服务器协议**（Language Server Protocol, LSP）的拟定。编辑器只要使用LSP 协议就可用于任何具备 LSP 兼容服务器的编程语言的开发。这意味着我们可以继续使用自己喜爱的编辑器，同时也不必放弃各种编程语言的高级编辑功能——这对于很多 Emacs 瘾君子来说尤其利好。

LORAWAN是一种低功耗广域网，专为低功耗、远距离和低比特率的通信场景而设计。它提供了边缘设备与网关设备之间的通信能力，能够通过后者将数据转发至应用程序或者后台服务。LoRaWAN通常用于分布式传感器组或物联网这些必须具备长电池寿命和远距离通信能力特点的设备上。它解决了在使用一般的WiFi进行低功耗广域网通信时的两个关键问题：通信距离和功耗。LoRaWAN已有若干实现，其中值得注意的是一个免费的开源实现——The Things Network。

随着机器学习从试验性使用转向生产环境，需要一种可靠的方式来托管和部署这些可远程访问的模型，并能随着消费者数量的增加而进行扩展。
(TensorFlow Serving)

MAPD是一个支持SQL的运行在GPU 上的内存列式分析性数据库。我们对于数据库负载到底是I/O密集所致还是计算密集所致有过争论。不过GPU的并行能力，结合 VRAM 的充足带宽，在某些场景下非常有用。MapD可以透明地将最频繁使用的数据（比如在group-by、过滤、计算操作以及 join 条件中所涉及的列）存放在VRAM中，而将其余的数据存放在主内存中。通过这种内存管理方式，MapD无需索引即可达到相当好的查询性能。尽管还有其他GPU数据库提供商，随着MapD近期开源了其核心数据库，以及通过GPU开放分析计划（GPU Open Analytics Initiative）的推广，MapD在这个领域处于领先地位。如果你的分析任务是计算密集型的，并能够利用GPU的并行性进行加速，且能纳入主内存中，我们建议评估MapD。

我们很喜欢那些能很好地解决单个问题的简单工具。**NETLIFY**正是这样一个工具。用它可以创建静态网站内容，提交到GitHub，然后网站就能快速、轻松地上线可用了。Netlify提供命令行工具来控制流程，支持CDN（内容分发网络），可以与Grunt这样的工具协同工作。更重要的是Netlify支持HTTPS。

机器学习模型已经开始渗入到日常的商业应用中。当有足够的训练数据可用时，这些算法可以解决那些以前可能需要复杂的统计模型或试探法的问题。随着机器学习从试验性使用转向生产环境，需要一种可靠的方式来托管和部署这些可远程访问的模型，并能随着消费者数量的增加而进行扩展。**TENSORFLOW SERVING**通过将远程gRPC接口暴露给一个被导出来的模型，解决了上述部分问题。这支持以多种方式部署训练完成的模型。TensorFlow Serving也接受一系列的模型来整合持续的训练更新。其作者维护了一个Dockerfile来简化部署过程。据推测，gRPC 的选择应与 TensorFlow 执行模型保持一致。但是，我们通常都会对需要代码生成和本地绑定的协议保持警惕。

凭借**WINDOWS CONTAINERS**，微软正在容器化的道路上奋起直追。截至本期雷达，微软提供了2个可以在Docker容器中运行的Windows OS的镜像——Windows Server 2016 Server Core和Windows Server 2016 Nano Server。尽管Windows Container还有提升的空间，比如缩减镜像文件大小，增强对生态系统的支持，以及完善相关文档，我们的团队已经开始在其他容器化技术已经成功应用的场景（如构建代理）中使用它们了。

在中间件中实现业务逻辑和流程编排，特别是要在其中运用专业技能和专用工具来将中间件作为一个单点来创建，以实现规模化和控制，对此我们还是心怀顾虑的。API网关市场竞争十分激烈，那些供应商们持续地向其产品中添加新的功能，从而体现产品之间的差异化。这一点令上述趋势得以持续。但这样会产生出**过度庞大的API网关产品**。其功能在本质上就是反向代理，这助长了难以测试和部署的系统设计。API网关确实可以提供一些处理某些特定问题的实用程序——例如身份验证和速率限制——但是任何领域业务逻辑都应该仅出现在应用程序或服务中，而不是网关中。

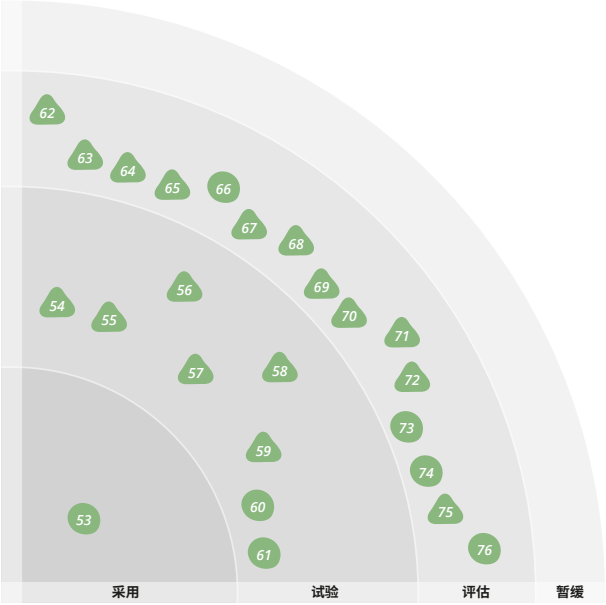
工具

采用
53. fastlane

试验
54. Buildkite *NEW*
55. CircleCI *NEW*
56. gopass *NEW*
57. Headless Chrome for front-end test *NEW*
58. jsoniter *NEW*
59. Prometheus
60. Scikit-learn
61. Serverless Framework

评估
62. Apex *NEW*
63. assertj-swagger *NEW*
64. Cypress *NEW*
65. Flow *NEW*
66. InSpec
67. Jupyter *NEW*
68. Kong API Gateway *NEW*
69. kops *NEW*
70. Lighthouse *NEW*
71. Rendertron *NEW*
72. Sonobuoy *NEW*
73. spaCy
74. Spinnaker
75. Spring Cloud Contract *NEW*
76. Yarn

暂缓



我们的团队非常喜欢托管的CI / CD工具——**BUILDKITE**，因为它既简单，搭建速度又快。只需要安装一个轻量级的代理应用程序来连接构建代理与在Buildkite上所托管的构建服务，就可以使用私有或云端的机器来执行构建。与使用托管的代理相比，能在这种级别上对构建代理的配置进行控制，在多数情况下都是一个优势。

CIRCLECI是一个持续集成引擎，它既可以在SaaS云服务上使用，又可以私有部署使用。它已经被我们很多开发团队在SaaS平台上当作常用CI工具。这些团队需要低摩擦（low-friction）和易于搭建的构建与部署流水线。CircleCI 2.0的版本支持构建任务的工作流，并具备扇入（fan-in）和扇出（fan-out）流模式和手动触发模式，且支持移动开发。它也允许开发者在本地运行流水线。另外 CircleCI 能很容易地与诸如Slack及其他通知和报警系统进行集成。就像使用任何其他承载公司资产的SaaS产品一样，我们建议用户仔细查看CircleCI的安全实践。

Gopass增加了诸如多用户密码管理、层级式密码存储、交互式查找、基于时间的一次性密码（TOTP）,以及二进制存储格式等功能。（gopass）

GOPASS是一个基于GPG和Git的团队密码管理解决方案。它的前身是pass，并在此基础上增加了诸如多用户密码管理、层级式密码存储、交互式查找、基于时间的一次性密码（TOTP）,以及二进制存储格式等功能。由于它的存储格式与pass基本兼容，因此可以直接从pass迁移过来。这意味着只需调用一次存储密钥就能将其集成到迁移的整备工作流中。

从2017年中开始，Chrome 用户有了一个在Headless模式下运行浏览器的新选择。这非常适合执行那些依赖浏览器的前端测试，而不必在屏幕上显示操作的结果。而在此之前，这属于 PhantomJS 的地盘，但Headless Chrome正在迅速取代那种用 JavaScript 驱动 WebKit 引擎的方法。测试在 Headless Chrome 浏览器中的运行速度要快得多，而且在行为上更贴近真实的浏览器，但我们的团队也发现它比 PhantomJS 要占用更多内存。 基于上述优点，针对**前端测试的HEADLESS CHROME** 很可能成为这个领域的事实标准。

如果正在寻找用Go和Java编写的高性能 JSON 编码/解码工具，那就试试开源库**JSONITER**，它与Go语言中的标准JSON编码包相兼容。

最初由Soundcloud开发的监控和时序数据库工具Prometheus，不仅在进行持续改进，而且其使用率也获得了一些提升。
（Prometheus）

最初由Soundcloud开发的监控和时序数据库工具Prometheus，不仅在进行持续改进，而且其使用率也获得了一些提升。**PROMETHEUS**主要支持基于“拉动”的HTTP模型，同时也能支持告警，这令其能够成为运维工具箱中经常得到使用的工具。在本期技术雷达编撰过程中，Prometheus 2.0正处于预发布阶段，并且还在不断演进。Prometheus的开发者们正专注于核心时序数据库以及各种可用的度量指标之上。对于Prometheus用户来说，Grafana已成为首选的仪表板可视化工具，且可以购买该工具的技术支持服务。我们的技术团队还发现，Prometheus在索引和搜索能力上能够作为Elastic Stack很好的补充。

APEX是一个能够轻松构建、部署和管理AWS Lambda 函数的工具。有了Apex，就能使用AWS尚未原生支持的包括Golang、Rust等在内的编程语言来编写函数。这一点是通过Node.js shim来实现的。它会创建一个子进程，并通过标准输入和输出来处理各种事件。Apex还有很多不错的特性，可以改善开发者的体验。我们特别欣赏它能在本地测试函数的能力，以及在将变更运用到AWS资源之前能对其进行预演的能力。

ASSERTJ-SWAGGER 是一个 AssertJ 工具库，能够用来验证API的实现是否符合其契约规格。当 API 端点的实现发生了更改但未更新其 Swagger 规格，或未能发布更新后的文档时，我们的团队就能通过使用 assertj-swagger 来捕获这些问题。

修复 CI 上失败的端到端自动化测试会是一段痛苦的经历，尤其是在 headless 模式下。而**CYPRESS**是一个很有用的工具，它能帮助开发人员轻易地构建端到端自动化测试，并且把测试的步骤录制在一个 MP4 文件里。这使得开发者可以通过查看测试视频来修复测试，而不是在headless模式下去重现问题。Cypress 不仅是一个测试框架，更是一个强大的测试平台。当前我们已经把其 CLI 集成到了我们项目的 headless CI 里。

FLOW是一个针对 Javascript 的静态类型检查工具，它可以为整个代码库逐步增加类型检查。不同于通过定义另一种语言来实现静态类型检查的 Typescript 语言，Flow 可以被逐步添加到支持 ECMAScript 第5、第6 以及 第7版的已有 Javascript 代码库中。我们建议把 Flow 添加到持续集成部署流水线中，并从最关注的代码开始做静态类型检查。使用 Flow 能使代码更清晰，重构更可靠，并在构建过程的早期就捕获到类型相关的代码缺陷。

过去几年间，我们注意到分析笔记本应用（analytics notebooks）的流行度在持续上升。这些应用都是从Mathematica应用中获得灵感，能够将文本、数据可视化和代码活灵活现地融入到一个具备计算能力的文档中。在上个版本的技术雷达中我们所提到的基于Clojure的GorillaREPL，就属于此类工具。但随着人们对机器学习的兴趣不断增加，以及该领域中的从业者们逐渐将Python作为首选编程语言，大家开始集中关注Python分析笔记本了。其中**JUPYTER**看起来在ThougthWorks团队中格外引人注目。

Kong是一个由Mashape公司搭建和支持的开源API网关。该公司也提供企业级产品，来将Kong与其专有的 API 分析和开发者门户工具相结合。它们能以各种配置进行部署，来作为边缘API网关或内部 API 代理。Kong 所基于的 OpenResty 通过其Nginx模块和用于扩展的Lua插件，为其强大和高效的功能奠定了基础。Kong 既可以使用PostgreSQL进行单一

区域部署，也可以使用 Cassandra 进行多区域配置。我们的开发人员已经享受到 Kong 的高性能、API 优先的方式（能使其配置自动化）以及易于容器化部署的种种好处。不像过度庞大的API网关那样，**KONG API 网关**只拥有更少的功能，但实现了关键的API网关功能，如流量控制、安全性、日志记录、监控和身份验证。 我们很高兴能在不久的将来以一个 sidecar 配置的方式对 Kong 进行评估。

KOPS是用于在生产环境上创建和管理高可用性Kubernetes集群的命令行工具。 最初它针对AWS，但现在已经对其他供应商提供了试验性支持。它可以快速地启动并运行。虽然某些功能（如滚动升级）尚未开发完毕，但其社区令人印象深刻。

谷歌编写的**LIGHTHOUSE**工具可用于评估Web应用程序是否遵守Progressive Web App标准。今年，新发布的 Lighthouse 2.0 向其基本工具集中新增了性能指标和可访问性检查这些功能。这些新增功能现在已经被纳入 Chrome 标准开发者工具的 audit 选项卡下。Lighthouse 2.0 也是 Chrome headless 模式的另一个受益者。 鉴于该工具可以由命令行直接执行，或作为Node.js应用程序独立运行，因此Pa11y及类似工具提供了一种能在持续集成流水线中运行可访问性检查的替代方案。

JavaScript Web 富应用的一个老问题是如何使这些页面的动态渲染部分可供搜索引擎检索。无法渲染JavaScript的爬虫机器人可以被路由到Rendertron服务器来进行渲染。
（Rendertron）

JavaScript Web 富应用的一个老问题是如何使这些页面的动态渲染部分可供搜索引擎检索。为此开发人员采用了各种各样的技巧，包括使用React.js的服务端渲染，外部服务或预渲染内容。现在谷歌 Chrome 新的 headless 模式又贡献了一个新的技巧—— **RENDERTRON**，即 Chrome 的headless 渲染解决方案。它在一个 Docker 容器中封装了一个 headless 的 Chrome 实例，可以作为独立的HTTP服务器来部署。无法渲染JavaScript的爬虫机器人可以被路由到此服务器来进行渲染。 虽然开发人员也可以部署自己的 headless Chrome代理并配置相关的路由机制，但 Rendertron 简化了配置和部署过程，并提供了令爬虫机器人进行检测和路由的中间件示例代码。

SONOBUOY是一个以非破坏性的方式在任何Kubernetes群集上运行端到端“合规性测试”的诊断工具。由两位Kubernetes项目发起者创办的Heptio公司的团队构建了这个工具，来确保各种Kubernetes发行版和配置都符合最佳实践，同时遵循开源标准化原则以实现集群互操作性。我们正在尝试使用Sonobouy作为“基础设施即代码”构建流水线的一部分，并对Kubernetes安装进行持续监控，以验证整个集群的行为和健康状况。

如果用Spring框架来实现Java服务，那么可以考虑用**SPRING CLOUD CONTRACT**来进行消费者驱动的契约测试。目前，该工具的生态系统支持根据契约来验证客户端的调用以及服务器端的实现。与Pact （一个开源的消费者驱动契约测试工具集）相比，它不支持契约代理，也不支持其它编程语言。不过它能与Spring生态系统完美集成，比如使用Spring Integration进行消息路由。

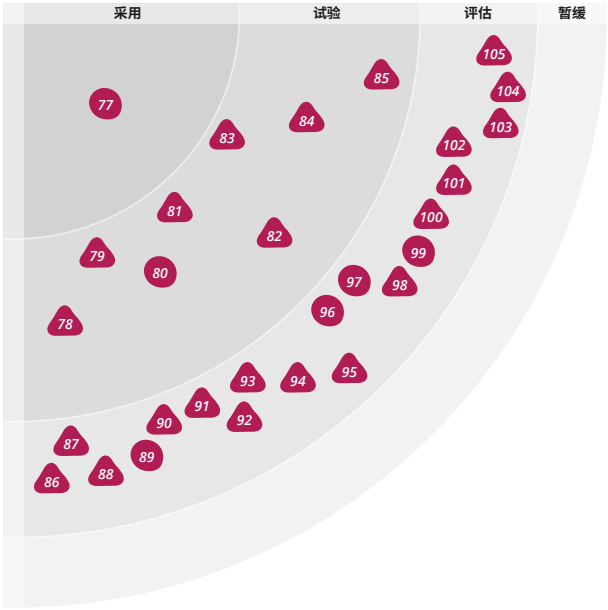
语言&框架

采用
77. Python 3

试验
78. Angular
79. AssertJ *NEW*
80. Avro
81. CSS Grid Layout *NEW*
82. CSS Modules *NEW*
83. Jest *NEW*
84. Kotlin
85. Spring Cloud

评估
86. Android Architecture Components *NEW*
87. ARKit/ARCore *NEW*
88. Atlas and BeeHive *NEW*
89. Caffe
90. Clara rules *NEW*
91. CSS-in-JS *NEW*
92. Digdag *NEW*
93. Druid *NEW*
94. ECharts *NEW*
95. Gobot *NEW*
96. Instana
97. Keras
98. LeakCanary *NEW*
99. PostCSS
100. PyTorch *NEW*
101. single-spa *NEW*
102. Solidity *NEW*
103. TensorFlow Mobile *NEW*
104. Truffle *NEW*
105. Weex *NEW*

暂缓



在往期的雷达中，我们不确定是否应该强烈推荐**ANGULAR**，因为从本质上来说，它是一个新的、整体上没那么让人兴奋的框架，仅仅是和那个我们曾经喜爱过的 AngularJS 同名而已。目前已经进化到第5个版本的 Angular，在提供向后兼容性的同时，有了稳定的改进。我们的一些团队已经在生产环境上应用 Augular，他们很满意最终的效果。基于这个原因，在这一期雷达中，我们把 Angular 移到了“试验”阶段，来表示我们的一些团队现在把它当作不二之选。然而我们的大多数团队，仍然会更倾向于选择React, Vue 或者Ember。

ASSERTJ是一个提供流式断言接口的Java库，可以很容易在测试代码中传达测试的意图。AssertJ提供了可读的错误信息、软断言以及改进过的对集合和异常支持。我们看到一些团队默认选择使用它，而不是JUnit和Hamcrest组合。

CSS网络布局（CSS Grid Layout）是一个二维（two-dimensional）的基于网格的布局系统，它所提供的机制使用了一组可预测的尺寸调整行为，将布局的可用空间划分为行和列。

（CSS网格布局）

即使CSS没有为创建布局提供明确的支持，但它依然是网页布局的首选。Flexbox提供了更简单的，一维（one-dimensional）的布局，但开发人员通常还是会采用一些库和工具包实现更复杂的布局。**CSS网格布局**（CSS Grid Layout）是一个二维（two-dimensional）的基于网格的布局系统，它所提供的机制使用了一组可预测的尺寸调整行为，将布局的可用空间划分为行和列。它不需要任何额外的库，并能与Flexbox和其他CSS元素集成得很好。然而，由于IE11仅支持其部分规范，所以目前仍然依赖 Windows 7上IE浏览器的用户需求会被忽略。

大多数大型CSS代码库都需要复杂的命名机制来避免全局命名空间中的冲突。**CSS MODULES**通过为每个CSS文件中的所有class创建局部作用域来解决这些问题。当这个文件被导入到一个JavaScript模块，其中的CSS class可以通过名称字符串来引用。然后，在构建工具（Webpack，Browserify等）中，class名称被替换为自动生成的全局唯一字符串。这是一个重大的职责转换。以前，人们不得不通过管理全局命名空间来避免class命名冲突的问题，现在这个职责移交给构建工具。我们在CSS Modules遇到了一点小麻烦：功能测试经常会超出局部作用域，因此不能通过CSS文件中定义的名称来引用class。针对这个问题，我们建议使用ID或是data属性作为替代方案。

Jest是一个“零配置”的前端测试工具，具有诸如模拟和代码覆盖之类的开箱即用特性，主要用于React和其他JavaScript框架。
(Jest)

我们团队对采用**JEST**做前端测试的结果非常满意。它提供了一种“零配置”的开发体验，并具备诸多开箱即用的功能，比如 mock 和代码覆盖率。你不仅可以将此测试框架应用于React.js应用程序，也可以应用于其他 JavaScript 框架。Jest 经常被夸大的特性之一是 UI 快照测试。快照测试可以作为测试金字塔上层一个很好的补充，但请记住，单元测试仍然是坚实的基础。

对Android的完美支持为迅速发展的**KOTLIN**语言提供了额外的推动力，我们也正在密切关注Kotlin / Native（基于LLVM，可以将Kotlin代码编译为原生可执行文件）的进展。在使用Anko库开发Android应用时，我们已经尝到了空指针安全、数据类和易于构建DSL的甜头。尽管初始编译速度慢，且只有IntelliJ才提供一流的IDE支持，但我们仍然建议尝试一下这种新颖简洁的现代语言。

SPRING CLOUD在持续演进的过程中增加了许多有趣的新特性。例如在spring-cloud-streams项目中，对Kafka Streams绑定的支持让采用Kafka和RabbitMQ通过连接器构建消息驱动的应用变得相对容易。正在使用该特性的ThoughtWorks的团队都认同它能在使用像Zookeeper这样的复杂基础设施时提供便捷性，也对构建分布式系统时需要解决的常见问题提供支持，例如使用spring-cloud-sleuth进行跟踪。目前我们已成功将它应用在多个项目中，不过大家仍然需要注意其适用场景。

一直以来，Google的Android文档实例缺乏架构和结构。随着**ANDROID架构组件**的发布，这种状况有所改善，这是一组有主见的库，它们帮助开发者用更好的架构创建 Android 应用程序。 它们解决了Android开发的长期痛点：处理生命周期，分页，SQLite数据库以及配置变更时的数据持久化。这些库无须一起使用，你可以选择最需要的集成到现有项目中。

我们在移动增强现实中看到了很多令人激动的活动，其中大部分来自**ARKIT**（Apple提供的原生AR库）/**ARCORE**（Google提供的原生AR库）的加持。 这些库将移动AR技术带入主流，让后者得到大量采用。尽管如此，各大公司在寻找真实的用户场景（而非一些花哨的Demo）和真正增强用户体验的解决方案方面还是面临挑战。

多应用策略备受争议，尤其现在越来越少的用户愿意再下载新的应用程序。不同于推出一个新的应用然后为下载量而努力，许多团队必须通过一个已经被广泛安装的应用来发布新的功能，这给应用程序的架构带来了挑战。**ATLAS**和**BEEHIVE**是分别用于Android和iOS的模块化解决方案。它们能让多个团队工作于物理隔离的不同模块，并且从一个门面应用中重新组装或者动态加载这些模块。它们都是Alibaba的开源项目，因为Alibaba也曾面临同样的下载量减少和单个应用架构挑战的问题。

“你并不需要一个规则引擎”，这常常是选择规则引擎时的首要法则，因为我们已经见到太多的人基于一些臆想的理由，将自己绑定在难以测试的黑盒的的规则引擎上——原本定制化应该是更好的解决方案。虽说如此，在一些规则引擎确实适用的场合，我们采用 **CLARA RULES** 取得了很好的成功。不同于其他的规则引擎，它使用简单的 Clojure 代码来表达和执行规则，这意味着规则可以被很好地重构、测试和版本化。比起追求“业务人员可以直接编辑业务规则”的错觉，Clara Rules 能够很好地驱动业务专家和开发人员之间的合作。

CSS IN JS是一种用JavaScript编写CSS样式的技术，通过鼓励采用一种通用模式，编写样式以及应用样式的JavaScript组件，使样式和逻辑的关注点得到统一。该领域中的新秀——诸如JSS，emotion和styled-components，依靠工具来将CSS-in-JS代码转化成独立的CSS样式表，从而适合在浏览器里运行。这是在JavaScript中编写CSS的第二代方法，与以前的方法不同，它不依赖于内联样式，这意味着它能支持所有CSS特性，使用npm生态共享CSS以及跨平台使用组件。我们的团队发现styled-components很适合像React这样基于组件的框架，并且可以使用jest-styled-components做CSS的单元测试。这是个新兴的领域且变化迅速。用该方法时，在浏览器里人工调试生成的class名称会需要费些功夫，并且可能不适用于那些前端架构不支持重用组件并需要全局样式的项目。

DIGDAG是一个在云中构建、运行、调度和监控复杂数据管道的工具。你可以使用丰富的开箱即用操作符在YAML中定义这些管道，也可以通过API构建属于自己的管道。Digdag具有数据管道解决方案中的大多数常见功能，例如依赖关系管理、易于复用的模块化工作流、安全的密码管理和多语言支持。我们最感兴趣的功能是它对多种云平台的支持，这允许你通过AWS RedShift，S3和Google BigQuery等服务来移动和连接数据。随着越来越多的云提供商提供相互竞争的数据处理解决方案，我们认为Digdag（以及类似的工具）是充分利用这些平台的最佳选择。

DRUID是一个具有丰富的监控特性的JDBC连接池。它有一个内置的SQL解析器，提供了对数据库中执行的SQL语句语义级别的监控。注入或可疑的SQL语句将被拦截，并直接在JDBC层记录下来。查询也可以基于它们的语义进行合并。这是一个阿里巴巴开源的项目，它反映了阿里巴巴从自己的数据库系统中学到的教训。

Android架构组件是一组有主见的类库，能够帮助开发者用更好的架构创建 Android 应用程序。

(Android架构组件)

ECHARTS是一个轻量级的图表库，对不同类型的图表和交互有丰富的支持。ECharts完全基于Canvas API，因此即使处理100k +数据点也具有令人难以置信的性能，并且还针对移动用户进行了优化。 凭借其扩展项目ECharts-X，它还可以支持3D绘图。ECharts 是一个百度开源项目。

Go语言能够被编译为裸片上运行的目标程序，这使得嵌入式系统开发领域对它的兴趣与日俱增 。**GOBOT**是一个用于机器人、物理计算和物联网(IoT)的框架，它基于Go语言编写，并且支持多个平台。我们在一个对实时性响应没有要求的实验性机器人项目中使用了GoBot，并且用GoBot创建了开源的软件驱动。GoBot的HTTP API使其与移动设备的集成十分容易，从而能创建更丰富的应用。

ThoughtWorks的许多移动开发团队对一款可以检测Android和Java中令人讨厌的内存泄漏工具**LEAKCANARY**感到非常兴奋。LeakCanary与App集成非常简单，同时它也提供能够清晰回溯内存泄漏原因的通知。把它加到你的工具包，它可以帮你节省在多个设备上排查内存泄漏的时间。

PYTORCH是Lua机器学习框架Torch在Python语言下的完整重写版。比起Tensorflow，它还很新不够成熟，但在程序员的眼里它却很好用。 因其面向对象的特性和原生的Python实现，模型可以表达得更加清晰简洁，并可以在执行过程中调试。尽管最近涌现出了许多这类框架，但PyTorch拥有Facebook和广泛合作伙伴的支持，包括应该会继续支持CUDA架构的NVIDIA。ThoughtWorks的团队发现，PyTorch在模型的设计开发及试验阶段拥有着明显的优势，但在大规模的生产环境中的训练及实现仍然离不开TensorFlow。

SINGLE-SPA是一个JavaScript元框架，它允许我们使用不同的框架构建微前端，而这些框架可以共存于单个应用中。 一般来说，我们不建议在单个应用中使用多个框架，但有时却不得不这么做。 例如当你在开发遗留系统时，你希望使用现有框架的新版本或完全不同的框架来开发新功能，single-spa就能派上用场了。鉴于很多JavaScript框架都昙花一现，我们需要一个解决方案来应对未来框架的变化，以及在不影响整个应用的前提下进行局部尝试。在这个方向上，single-spa是一个不错的开始。

智能合约编程需要一种比交易处理脚本更具表现力的语言。在众多为智能合约设计的新编程语言中，**SOLIDITY**是最受欢迎的。这是一种面向合约的静态类型语言，其语法类似于JavaScript。 它抽象了智能合约中自我实现的业务逻辑。围绕Solidity的工具链也在快速成长。如今，Solidity是Ethereum平台的首选编程语言。鉴于已部署智能合约的不可变性，对依赖的严格测试和审计是至关重要的。

TENSORFLOW MOBILE使开发人员可以将各种理解和分类技术融入其iOS或Android应用程序。 考虑到手机上可用的传感器数量及其可收集的数据范围，这一点尤其有用。预先训练好的TensorFlow模型可以加载到移动应用程序中，并应用于实时视频帧，文本，语音等输入的处理中。手机为实现这些计算模型提供了一个令人惊讶的合适的平台。TensorFlow模型导出和加载的文件格式都是protobuf文件，这可能会为实现者带来一些问题。 Protobuf的二进制格式让检查模型很难，并要求你将正确的protobuf库版本链接到移动应用程序。但是，本地模型的执行，提供了一个很有吸引力的针对TensorFlow Serving的替代方案，这可以节省远程执行的通信开销。

我们在适合规则引擎的场景采用 Clara Rules 取得了很好的成功。我们喜欢通过它用简单的 Clojure 代码来表达和执行规则，这意味着规则可以被很好地重构、测试和版本化。

(Clara rules)

TRUFFLE是一个开发框架。它将现代化的 Web 开发体验带到了Ethereum平台。Truffle 接管了智能合约编译、库链接和部署，以及在不同区块链网络中处理制品的工作。我们喜爱 Truffle 的原因之一就是它鼓励开发者为智能合约编写测试。这一点非常值得重视，因为智能合约的编写通常涉及到金钱。得益于其内置的测试框架以及与TestRPC 的集成，Truffle 可以允许我们使用 TDD 的方式来编写智能合约。我们期望出现更多像 Truffle 的技术能促进在区块链领域中的持续集成实践。

WEEX是一套跨平台移动应用开发方案，采用了Vue.js的组件化语法。对于那些偏爱Vue.js简洁性的开发者，Weex是一个开发原生移动应用切实可行的选择，但同时也能胜任非常复杂的应用。已经有大量的复杂的应用构建于Weex框架，其中包括中国最流行的两款移动应用程序——天猫和淘宝。Weex最初由阿里巴巴开发，目前是Apache孵化项目。

最先获悉技术雷达发布消息，
了解独家研讨会及内容。

点击订阅

thght.works/Sub-CN

ThoughtWorks®

ThoughtWorks 是由一群极富激情和内驱力的员工所组成的技术咨询公司和社区。我们帮助客户将技术作为业务核心，一同创造最具价值的卓越软件。我们致力于推动社会革新，并与众多社会组织一起，力求通过技术的力量改变人们生活。

自成立起20多年来，ThoughtWorks 已发展成为超过4500人的公司，拥有为软件开发团队研发前沿工具的产品部门。ThoughtWorks 在全球15个国家设有42间办公室：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、南非、西班牙、土耳其、英国和美国。

thoughtworks.com