ThoughtWorks®

TECHNOLOGY RADAR VOL.16

洞察构建未来的技术和趋势

thoughtworks.com/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) | Badri Janakiraman | Bharani Subramaniam | Camilla Crispim
Erik Doernenburg | Evan Bottcher | Fausto de la Torre | 徐昊 | Ian Cartwright
James Lewis | Jonny LeRoy | Marco Valtas | Mike Mason | Neal Ford
Rachel Laycock | Scott Shaw | Srihari Srinivasan | Zhamak Dehghani

技术雷达中国区技术咨询顾问组：

顾宇	刘尚奇	伍斌	杨璐
黄雨青	林帆	魏喆	禚娴静
蒋帆	佟达	王晓峰	朱傲
刘先宁	滕慧集	鄢倩	张渊

最新动态

本版精彩集锦

会话式用户界面(CONVERSATIONAL UI)和自然语言处理

观看视频 *(thght.works/ConUI)*

人机对话——这种新的应用程序交互方式——伴随着苹果 Siri、微软小娜和谷歌 Allo 这样的工具,像风暴一样席卷了整个IT生态圈。随后这股风暴继续延伸到了家用设备,例如亚马逊的 Echo 和谷歌的 Home 。虽然构建会话式自然语言用户界面会遇到许多新的挑战,但是它所带来的益处是很显著的。亚马逊 Echo 的研发团队故意在该产品上省去了屏幕,从而迫使团队成员重新思考许多人机交互的场景。

这种“会话式”的趋势不仅限于语音。随着消息应用已经增长到可以主导电话通话和工作场所,我们看到了一些在智能聊天机器人协助下所发生的多人会话。随着这些平台的不断改进,它们将逐渐学会理解会话的上下文和会话意图,从而让人机交互更加逼真和引人入胜。

市场和主流媒体对这个领域的兴趣激增,增加了开发者对这种新的个人“外部皮层”(exocortex)交互模式的兴趣。

智能即服务

观看视频 *(thght.works/IntSer)*

最近,一系列被我们称之为“智能即服务”(intelligence as a service)的平台已经爆发。这些平台都与各种强大的技术领域密切相关,从语音处理到自然语言识别、图像识别和深度学习。

几年前,要想具备这些能力还需要花费很昂贵的资源,但现在已经有开源或者基于 SaaS 平台的解决方案了。这也意味着“云计算之战”逐渐从存储和计算能力向认知能力转变。之前 Kubernetes 和 Mesos 这两个差异化工具的开源就是这场战争的见证。

在这个领域的所有大型厂商都有自己的产品,与此同时,一些利基厂商的产品也值得尝试。尽管我们对于这些服务在伦理和隐私方面的影响持保留态度,但我们相信创新地使用这些强大工具会带来很好的前景。我们的客户已经开始在新的视野上研究如何在他们的业务里把人工智能和商品的认知能力结合起来。

开发者体验成为新的差异化竞争优势

观看视频 *(thght.works/DevExp)*

多年来,用户体验设计一直是技术产品公司持续关注的关键差异化竞争优势。而现在面向开发者的工具和产品的快速崛起,加之工程人才的稀缺,迫使这些公司也开始关注开发者体验。

越来越多的组织依据所减少的“工程摩擦力”(engineering friction)来评估云产品,并将 API 视为产品来精心打磨,且专注于工程生产力来提升团队效率。在 ThoughtWorks ,我们一直执着于高效的工程实践,以及那些能让开发者们轻松工作的工具和平台。我们非常激动地看到业界开始采纳这些想法。

这些关键技术包括:将内部基础设施作为一种产品,令其具有足够的吸引力来与外部产品进行竞争;专注于自助服务系统;理解所开发的 API 的“开发者人机工程学”(developer ergonomics);对遗留系统进行封装;以及对开发者的“持续用户共情研究”(ongoing empathetic user research)的投入。

平台的崛起

观看视频 *(thght.works/RiseOTP)*

技术雷达的主题来自于审查过程中的观察和交流。在最近一次编辑技术雷达的过程中，我们注意到了进入平台象限的新条目的数量。我们认为这表明了平台在软件开发生态系统中有着更广阔的前景。

那些引人注目的硅谷公司向我们展示了构建一个合理的平台如何带来显著的效益。他们成功的一部分原因来自于找到适用于自身的封装和能力水平。从技术雷达所强调的高级功能（如自然语言处理）到基础设施平台（如亚马逊）来看，越来越多的“平台思维”出现在整个技术生态系统中。

当要通过产品化的 API 提供一些精选的功能时，企业开始考虑采用平台的方式。开发团队在集成和提升开发人员体验方面有了更多的思考。似乎业界终于走上了将“打包、便利和有用”进行合理组合这样一条道路。

我们喜欢这样来定义平台：平台应该提供一个自助服务 API，并且使之在团队环境中容易配置和创建——这很好地与“开发者体验”这样一个新兴的主题相呼应。我们期望在不久的将来，平台的定义和功能将得到进一步的完善。

盛行的PYTHON

观看视频 *(thght.works/PerPyt)*

Python 这门语言总是不断出现在有趣的地方。作为一门易用的通用编程语言，Python 在数学和科学编程领域具有坚实的基础。这使得它一直以来都为草根阶层的学术研究社区所采用。最近，围绕人工智能商品化及其应用的行业趋势，以及 Python 3 的成熟，给 Python 社区注入了新的活力。

这一卷的雷达重点介绍了一些能够促进 Python 人工智能生态圈发展的库，其中包括机器学习领域的 Scikit-learn，采用智能数据流图的 TensorFlow、Keras 和 Airflow，以及通过自然语言处理实现会话识别应用程序接口的 spaCy。我们越来越多地看到 Python 正在缩小组织内科学家和工程师之间的距离，并减弱了他们过去在最喜爱工具方面的偏见。

诸如微服务和容器的架构已经简化了 Python 在生产环境中的执行。工程师现在可以通过与语言和技术无关的 API，部署和集成由科学家们特别创建的 Python 代码。相比将特定语言（比如 R 语言）翻译到生产环境上的现有做法，这种流动性是朝构建研究人员和工程师之间一致的生态系统迈出的重要的一步。

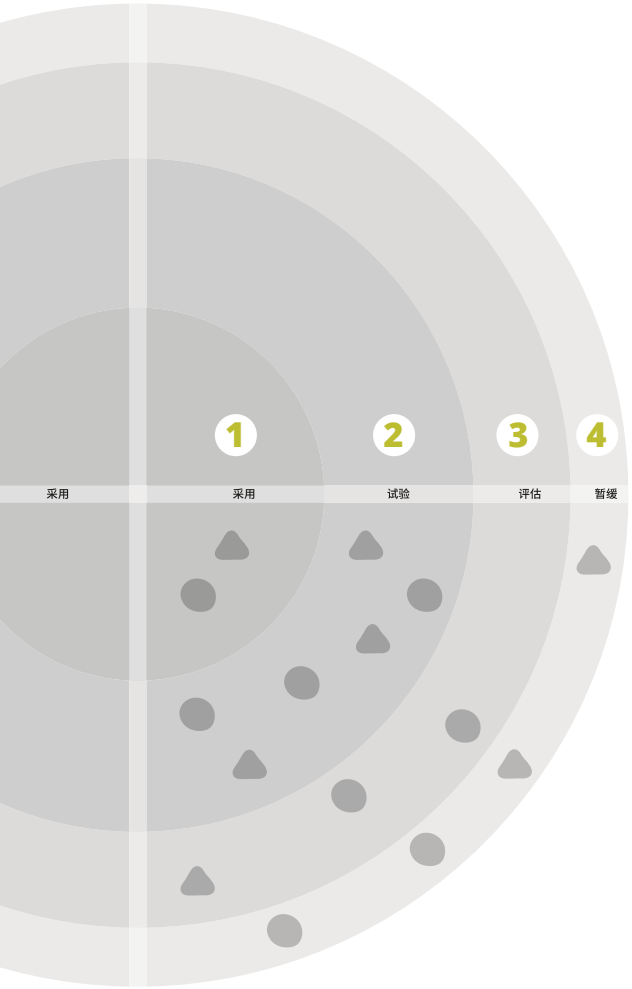
关于技术雷达

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 CIO 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，请参见 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 三角形代表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. Pipelines as code

试验
2. APIs as a product
3. Decoupling secret management from source code *NEW*
4. Hosting PII data in the EU
5. Legacy in a box *NEW*
6. Lightweight Architecture Decision Records
7. Progressive Web Applications *NEW*
8. Prototyping with InVision and Sketch *NEW*
9. Serverless architecture

评估
10. Client-directed query
11. Container security scanning
12. Conversationally aware APIs *NEW*
13. Differential privacy
14. Micro frontends
15. Platform engineering product teams *NEW*
16. Social code analysis *NEW*
17. VR beyond gaming

暂缓
18. A single CI instance for all teams
19. Anemic REST
20. Big Data envy
21. CI theatre *NEW*
22. Enterprise-wide integration test environments *NEW*
23. Spec-based codegen *NEW*

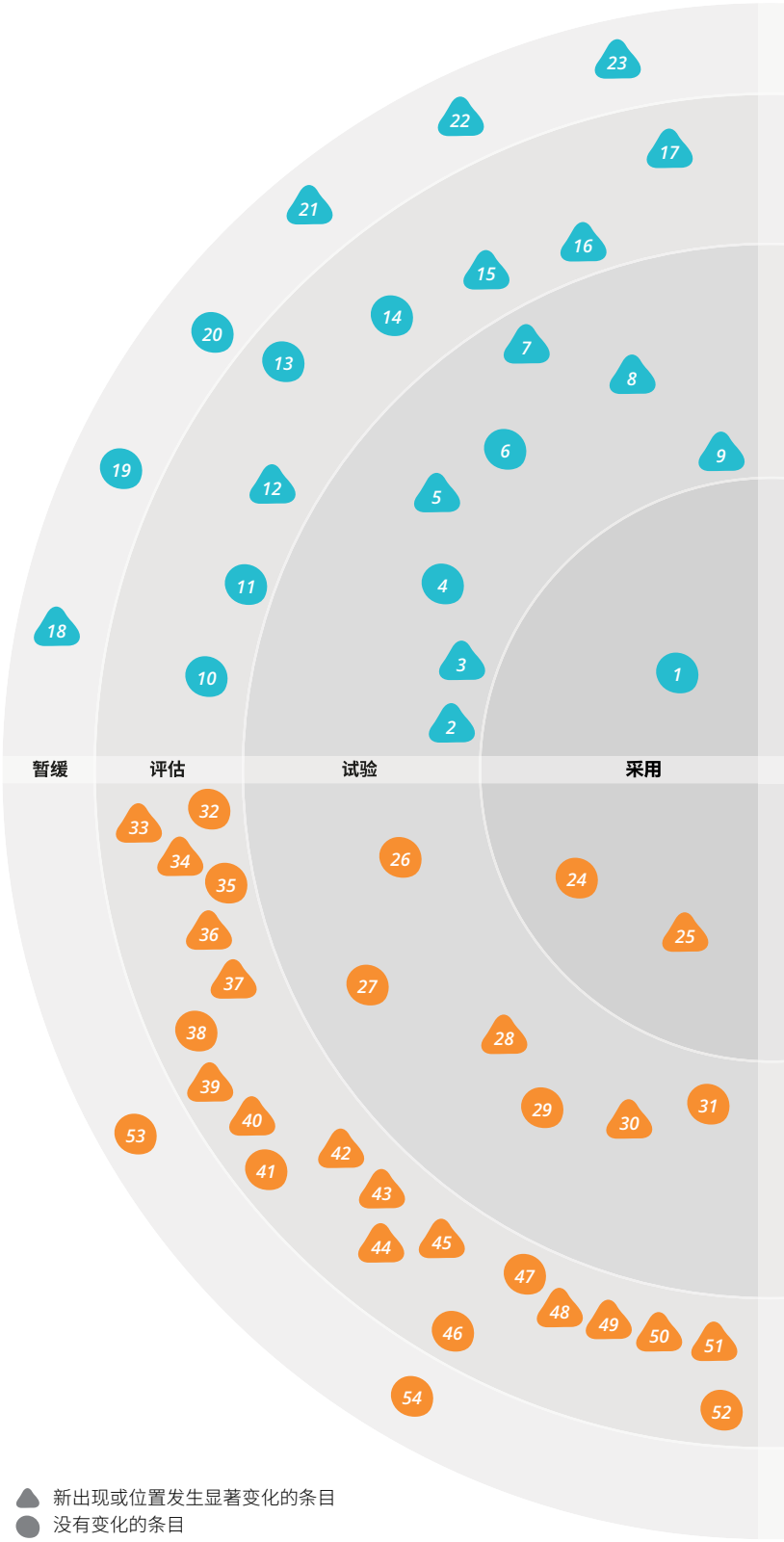
平台

采用
24. HSTS
25. Linux Security Modules

试验
26. Apache Mesos
27. Auth0
28. AWS Device Farm *NEW*
29. AWS Lambda
30. OpenTracing *NEW*
31. Unity beyond gaming

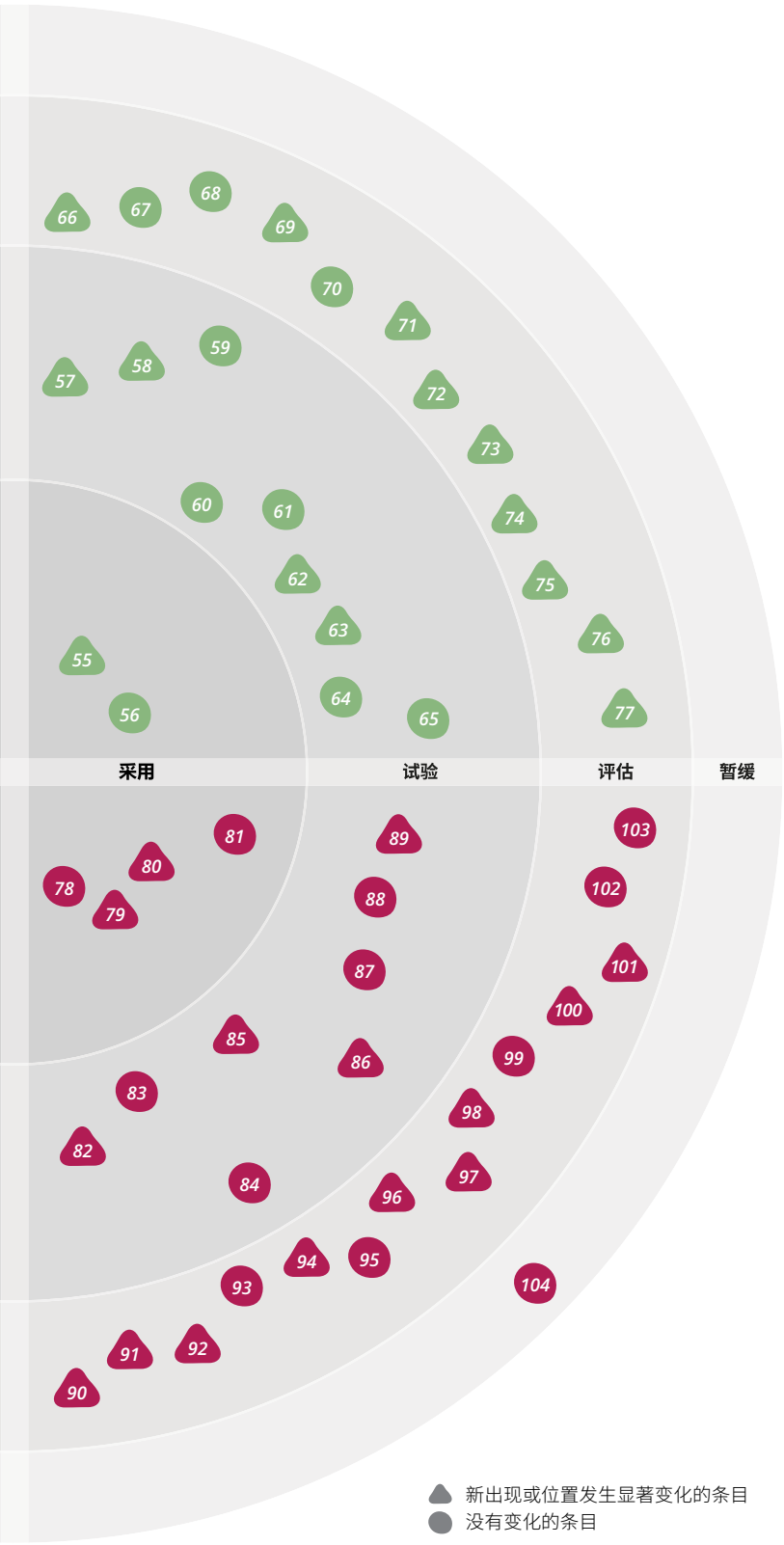
评估
32. .NET Core
33. Amazon API Gateway
34. api.ai *NEW*
35. Cassandra carefully
36. Cloud-based image comprehension *NEW*
37. DataStax Enterprise Graph *NEW*
38. Electron
39. Ethereum
40. Hyperledger *NEW*
41. IndiaStack
42. Kafka Streams *NEW*
43. Keycloak *NEW*
44. Mesosphere DCOS
45. Mosquitto *NEW*
46. Nuance Mix
47. OpenVR
48. PlatformIO *NEW*
49. Tango *NEW*
50. Voice platforms *NEW*
51. WebVR *NEW*
52. wit.ai

暂缓
53. CMS as a platform
54. Overambitious API gateways



▲ 新出现或位置发生显著变化的条目
● 没有变化的条目

THE RADAR



工具

采用
55. fastlane
56. Grafana

试验
57. Airflow *NEW*
58. Cake and Fake *NEW*
59. Galen
60. HashiCorp Vault
61. Pa11y
62. Scikit-learn
63. Serverless Framework *NEW*
64. Talisman
65. Terraform

评估
66. Amazon Rekognition *NEW*
67. Android-x86
68. Bottled Water
69. Claudia *NEW*
70. Clojure.spec
71. InSpec *NEW*
72. Molecule *NEW*
73. Spacemacs *NEW*
74. spaCy *NEW*
75. Spinnaker *NEW*
76. Testinfra *NEW*
77. Yarn *NEW*

暂缓

语言&框架

采用
78. Ember.js
79. Python 3
80. ReactiveX
81. Redux

试验
82. Avro *NEW*
83. Elixir
84. Enzyme
85. Hangfire *NEW*
86. Nightwatch *NEW*
87. Phoenix
88. Quick and Nimble
89. Vue.js

评估
90. Angular 2 *NEW*
91. Caffe *NEW*
92. DeepLearning.scala *NEW*
93. ECMAScript 2017
94. Instana *NEW*
95. JuMP
96. Keras *NEW*
97. Knet.jl *NEW*
98. Kotlin *NEW*
99. Physical Web
100. PostCSS *NEW*
101. Spring Cloud *NEW*
102. Three.js
103. WebRTC

暂缓
104. AngularJS

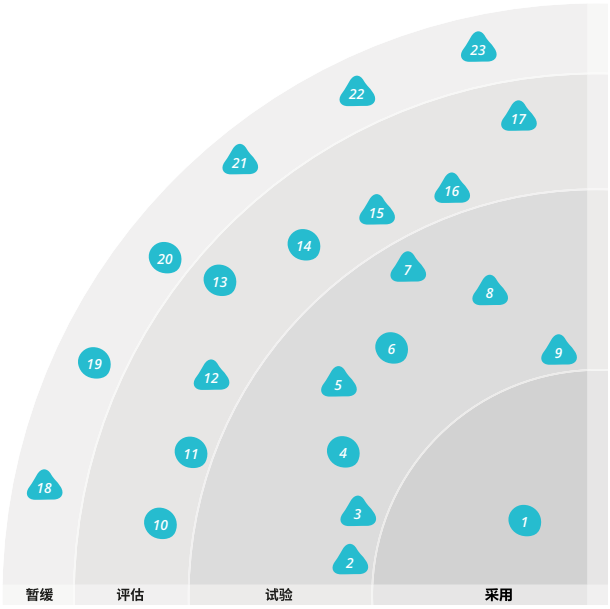
技术

采用
1. Pipelines as code

试验
2. APIs as a product
3. Decoupling secret management from source code *NEW*
4. Hosting PII data in the EU
5. Legacy in a box *NEW*
6. Lightweight Architecture Decision Records
7. Progressive Web Applications *NEW*
8. Prototyping with InVision and Sketch *NEW*
9. Serverless architecture

评估
10. Client-directed query
11. Container security scanning
12. Conversationally aware APIs *NEW*
13. Differential privacy
14. Micro frontends
15. Platform engineering product teams *NEW*
16. Social code analysis *NEW*
17. VR beyond gaming

暂缓
18. A single CI instance for all teams
19. Anemic REST
20. Big Data envy
21. CI theatre *NEW*
22. Enterprise-wide integration test environments *NEW*
23. Spec-based codegen *NEW*



企业已经全然接受通过 API 把业务能力暴露给内外部开发者。API 通过重组核心能力承诺了快速试验商业创意的能力。但 API 与普通企业集成服务有什么区别呢?其中的一个区别就是**将 API 当做产品 (APIs as a product)**，即使 API 消费者是企业内部的系统或开发人员。构建 API 的团队应该理解客户的诉求，并且让产品始终能够满足这些诉求。可用性测试 (Usability testing) 和用户体验研究有助于理解API的使用模式，并将产品思维带入到 API 中，从而得到更好的 API 设计。API 应该有一个负责任，负责关注用户并持续改进。在我们的经验中，缺乏产品导向会使普通企业集成和基于 API 平台构建的敏捷业务存在差异。

在往期的技术雷达里我们提到了诸如 git-crypt 和 Blackbox 这样的工具可以帮我们保证源代码内部的秘密信息安全。**从代码中解耦秘密信息的管理**是我们提醒技术人员存储秘密信息还有其他选项的另一种方式。例如，HashiCorp vault 持续集成服务器和配置管理工具都提供了脱离应用程序代码的秘密信息存储机制。这两种方法都切实可行，我们推荐你在项目中至少使用一种。

> 构建 API 的团队应该理解客户的诉求，并且让产品始终能够满足这些诉求。可用性测试 (Usability testing) 和用户体验研究有助于理解API的使用模式，并将产品思维带入到 API 中，从而得到更好的 API 设计。
>
> — APIs as a product

在遗留代码上工作，特别是大型的单体应用，是最糟糕的开发体验之一。尽管我们警告不要扩展和积极维护遗留单体应用，但它们仍然是各种环境中的依赖。开发人员往往低估了这些依赖开发所需的成本和时间。为了减少摩擦，开发人员采用虚拟机镜像或 Docker 容器来创建遗留系统及其配置的镜像。其目的是为了**封装遗留系统**并供开发人员在本地运行。从而消除重新构建、重新配置和共享环境时候对遗

留系统的需要。在理想情况下，团队通过流水线生成相应的遗留系统镜像。开发人员可以通过更可靠的方式在自己的沙盒环境中编排并运行这些遗留系统。虽然这种方法可以减少每个开发人员花费的总时间，但是当拥有下游依赖的团队不愿意创建遗留系统镜像供他人使用时，这种方法的成效会很有限。

渐进式 Web 应用 (PROGRESSIVE WEB APPLICATIONS)(PWAs)的增长是把用户带回 Mobile Web 以回应“移动应用疲劳”的最新一次尝试。它最初由 Google 在2015年提出，PWA 是利用了最新技术的优势来组合出最好的 Web 和原生移动应用程序的 Web 应用程序。它使用了一系列的开放标准技术，比如 service workers , app manifest 以及缓存和推送API 。我们可以通过这些技术创建出与平台无关的移动应用以及原生应用的用户体验。这平衡了 Web 应用和原生应用各自的优缺点，并帮助移动应用开发者打破了应用商店的限制去接触用户。你可以把 PWA 想作是具备原生应用功能和观感的 Web 站点。

InVision 和 Sketch 的结合使用改变了一些人开发 Web 应用程序的方式。虽然它们只是工具，但重要的是**使用 INVISION 和 SKETCH 进行原型设计**的技艺。创建丰富并可操作的原型作为实现前端和后端业务逻辑的起点，有助于加快开发并消除在实现细节上的争论。这两个工具的结合使用在打磨视觉设计的细节和捕获用户对交互式体验的早期反馈之间取得了正确的平衡。

无服务器架构这种方法通过短暂存在的计算能力来替代长期运行的虚拟机。这种计算能力会根据服务请求而存在，并在服务完成后立即消失。我们的团队非常喜欢无服务架构这种方法。这种方法工作良好，我们认为它是一种有效的架构选择。值得注意的是这种方法并不是一种“要么全部采用要么全部不用”的方法。我们的一些团队已经使用无服务器架构来部署新的系统模块，而对于其他模块则仍然使用传统的架构。虽然 AWS Lambda 几乎是无服务器的代名词，但是其他的云计算服务商都提供了相似的产品。我们也建议评估一些利基玩家，例如 webtask 。

> 诸如 Amazon Alexa，Google 语音服务和 Siri 这样的技术已经大大降低了基于语音的软件交互的门槛。然而，想要在许多现有的 API 之上构建更多的会话式输入（语音或文本）还很困难。
>
> — Conversationally aware APIs

诸如 Amazon Alexa，Google 语音服务和 Siri 这样的技术已经大大降低了基于语音的软件交互的门槛。然而，想要在许多现有的 API 之上构建更多的会话式输入（语音或文本）还很困难。特别是在涉及到有状态的交互场景，且后续的交互又需要知晓整个会话上下文时。在这种风格的交互中，如果我们想要询问从曼彻斯特到格拉斯哥的火车，就可以直接问“首班列车何时出发?”，而不必再次给出会话的上下文。通常这个上下文将出现在我们发送回浏览器的初始响应中。但在语音接口的情形下，我们需要在其他地方处理这个上下文。**会话感知 API** 是后端为前端服务模式的示例，其中前端是语音聊天平台。这种类型的 API 可以在代表语音前端呼叫底层服务时，通过管理会话的状态来处理这种交互方式的细节。

采用云计算和 DevOps，虽然能通过减少团队对集中式运维团队和基础设施的依赖来提高生产力。但与此同时，也制约了那些缺乏能力对完整应用和运维全技术栈进行自主管理的团队。有些组织开始在内部组建**平台工程产品团队**以迎接这些挑战。这些团队维护着一个内部的应用平台，使交付团队可以通过自助部署和维护应用，缩短了交付时间，同时也降低了技术栈的复杂性。这里的重点是该团队维护 API 驱动的自助服务和支持工具，而交付团队仍然需要对其在该平台上部署的应用负责。当组织考虑组建这样一个团队的时候应当非常小心，避免无意中创建了独立的 DevOps 团队，也不要把现有托管及运维设施打上平台的标签。

社会化代码分析通过将开发人员的行为与代码的结构化分析结合，丰富了我们对代码质量的理解。它利用版本控制库的数据，例如变更的频率和时间以及进行变更的人员。你可以选择自己编写脚本对此类数据进行分析，也可以使用诸如 CodeScene 之类的工具。CodeScene可以帮助你更好

的了解你的软件系统，它能识别出热点和复杂且难以维护的子系统；能通过分析分布式子系统在时间上的耦合发现子系统之间的耦合；还能帮你认识组织中的康威定律。随着分布式系统、微服务和分布式团队等技术趋势，我们相信代码的社会维度对于整体理解系统健康至关重要。

虚拟现实的想法已经存在了50多年了，随着计算技术的不断进步，许多想法都已被炒作和探索过。我们相信该领域目前已经达到了临界点。去年，市场上已经发布了价格适宜的、面向消费者的 VR 头戴式设备，再加上现代的图形显卡为这些设备提供了足够的性能以创造身临其境的体验。虽然这些头戴式设备目前主要还是针对视频游戏爱好者，但我们相信，它们在**游戏领域之外的 VR 应用**上还存在许多的可能性。但是，没有制作视频游戏经验的团队不应低估创建一个好的3D模型和纹理所需要的时间和技能。

虚拟现实的想法已经存在了50多年了，随着计算技术的不断进步，许多想法都已被炒作和探索过。我们相信该领域目前已经达到了临界点。

— VR beyond gaming

我们不得不再一次警告，不要为所有团队创建**单一持续集成实例**。尽管在理论上具有汇集和中心化特点的持续集成基础设施是一个不错的想法，但实际上在这个领域我们仍然看不到足够成熟的工具和产品可以达到期望的产出。那些必须使用中心化持续集成服务器的交付团队，常常依赖中心的团队去完成小的配置任务，或者在共享的基础设置和工具中排查问题，这给他们在进度上带来长时间的滞后。在目前的阶段，我们继续推荐组织限制对集中化管理措施的投资，而为交付团队运维自己的持续集成基础设施提供模式、指导原则和支持。

我们长期以来都是持续集成（CI）的支持者，同时也是编写持续集成服务器程序的先锋。以前，这些程序运行在独立的守护进程里，开发者可以每日提交到代码主干分支上。持续集成服务器就会构建项目并运行自动化的全面测试以保证整个软件系统正确的集成并且做好发布准备。这些充分遵照了持续交付的原则。遗憾的是，很多开发者只是简单的搭建了持续集成服务器就以为在做“持续集成”，但他们实际上会遗失持续集成的关键优点而导致失败。常见的失败模式包括：虽然在一个共享的主分支上运行持续集成，但是代码提交不频繁，所以集成并没有真正的“持续”。以及在一个测试覆盖率不足，甚至是长期状态为红的情况下进行构建；或者在功能分支上运行持续集成，这会导致持续隔离。“**不完整的持续集成**”看起来可能会让团队感觉很好，但是是无法通过任何可信的持续集成验证测试。

遗憾的是，很多开发者只是简单的搭建了持续集成服务器就以为在做“持续集成”，但他们实际上会遗失持续集成的关键优点而导致失败。“不完整的持续集成”看起来可能会让团队感觉很好，但是是无法通过任何可信的持续集成验证测试。

— CI theatre

企业级应用的整体的季度或月度的版本发布被认为是该领域的最佳实践，在部署到生产环境之前维护一个完整的环境以进行测试是非常有必要的。这些**企业集成测试环境**通常称为 SIT 或预生产环境）是当下持续交付常见的瓶颈。环境本身很脆弱而且维护成本很高，而这些环境通常存在一些需要由单独的环境管理团队手动配置的组件。在预生产环境的测试给出的反馈慢且不可靠，而且会重复测试那些在隔离的组件上已经测过的功能。我们建议企业以增量的方式为关键组件创建一条通向生产环境的独立途径。其中涉及到一些重要的技术包括契约测试，将发布与部署解耦，专注于平均恢复时间和生产环境下的 QA 。

回到 SOAP 在企业软件行业中风光的日子，从 WSDL 规范生成客户端代码的做法是一个被认可的，甚至鼓励的做法。不幸的是，这种方式产生的代码经常很复杂、不可测试、难以修改而且经常频繁出现跨平台实现不可工作的情况。随着 REST 出现，我们发现使用宽容读者模式用于演进那些需要提出和处理字段的 API 客户端会更好。最近，我们观察到一种令人不安的老习惯正在复苏，开发人员使用 Swagger 或 RAML 编写 API 规范生成程序代码，这种实践被我们称之为**基于规范的代码生成**。尽管这样的工具对于驱动 API 的设计和提取文档非常有用，但我们得警惕直接从这些规范生成客户端代码的便捷诱惑。这种生成的代码将会难以维护和测试。

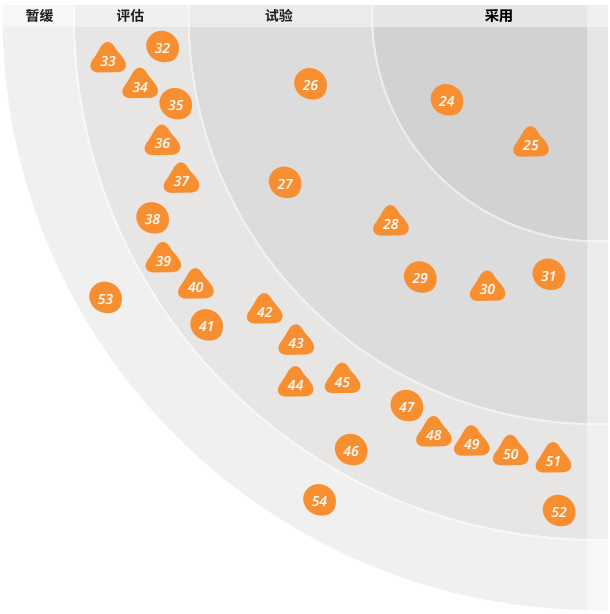
平台

采用
24. HSTS
25. Linux Security Modules

试验
26. Apache Mesos
27. Auth0
28. AWS Device Farm *NEW*
29. AWS Lambda
30. OpenTracing *NEW*
31. Unity beyond gaming

评估
32. .NET Core
33. Amazon API Gateway
34. api.ai *NEW*
35. Cassandra carefully
36. Cloud-based image comprehension *NEW*
37. DataStax Enterprise Graph *NEW*
38. Electron
39. Ethereum
40. Hyperledger *NEW*
41. IndiaStack
42. Kafka Streams *NEW*
43. Keycloak *NEW*
44. Mesosphere DCOS
45. Mosquitto *NEW*
46. Nuance Mix
47. OpenVR
48. PlatformIO *NEW*
49. Tango *NEW*
50. Voice platforms *NEW*
51. WebVR *NEW*
52. wit.ai

暂缓
53. CMS as a platform
54. Overambitious API gateways



“最小权限原则”鼓励我们限制软件只访问他们需要的资源。然而在通常情况下，Linux进程可以执行其运行的用户可以做的任何操作，包括绑定端口和执行脚本。**LINUX 安全模块 (LSM)** 框架允许将安全性扩展至内核，例如 Linux 使用该模块来实现 MAC 。SELinux 和 AppArmor 是最著名的 LSM 兼容实现，它们随 Linux 内核一起发布。我们建议团队学习使用这些安全框架（这就是为什么我们将其放置在采用），它可以帮助团队评估谁可以访问共享主机上的哪些资源（包括其中的服务）。这种保守的访问管理方法将帮助团队在其SDLC流程中建立更好的安全性。

AMAZON API GATEWAY 允许开发者把 API 服务暴露给互联网的用户。它提供了 API 网关的常见功能：流量管理，监控，认证和授权。我们的团队在把它和 AWS Lambda 集成作为无服务器架构的一部分上有很积极的评价。另一方面，我们在把它用作一个运行在 EC2 上的 HTTP/HTTPS 端点之前的更通用的前置网关时遇到了更多挑战，对我们造成了阻碍的是 VPC 的缺乏互动性和在网关上建立客户端证书验证的困难。基于这种混合的使用体验，我们建议团队结合使用 AWS API Gateway 和 Lambda 。但在更通用的配置里使用它时要评估其适用性。

海量的移动设备使得很多公司几乎不可能在所有设备上测试他们的移动应用程序。**AWS DEVICE FARM** 是一种移动应用测试服务，可以让您的 Android ， iOS 和 Web 应用同时运行在各种物理设备上，并与应用程序进行交互它会在

每次应用运行期间生成相似的日志，性能图标和屏幕截图，以及提供常规和特定设备的反馈。该服务为用户提供了很大的灵活性，允许改变每个设备的状态和配置，以便重现一些非常特定的测试场景。我们的团队正在使用 AWS Device Farm 在最大安装量的设备上运行端到端测试。

> 随着单体应用被更复杂的（微）服务生态系统所取代，跨越多个服务的请求追踪正成为常态。
>
> — OpenTracing

随着单体应用被更复杂的（微）服务生态系统所取代，跨越多个服务的请求追踪正成为常态。幸运的是 **OPENTRACING** 正在迅速成为分布式追踪的事实上的标准。它由 Uber，苹果，Yelp 和各种其他大厂商开发，支持多种分布式追踪系统，如 Zipkin，Instana 和 Jaeger。OpenTracing 标准目前提供厂商中立的六种语言实现：Go，JavaScript，Java，Python，Objective-C，以及 C++。

伴随着近期聊天机器人和语音平台的涌现，我们看到了类似 **API.AI** 等工具和平台的激增，这些工具和平台提供了从文本和可接入会话流中提取意图的服务。该公司最近被 Google 收购，这项“自然语言理解即服务”业务开始与在这一领域中的其他参与者进行竞争，比如 wit.ai 和 Amazon 的 Lex。

图片理解曾经是一项黑科技，它需要一队数据科学家在现场支持。然而近年来，我们已经接近解决像图片和脸部分类，脸部对比，脸部特征提取，以及脸部识别等问题。**云端图片理解**以服务的方式提供对机器学习能力的访问，像 Amazon Rekognition，Microsoft Computer Vision API，以及 Google Cloud Vision API，能够给增强现实（AR）应用和任何关于图片标签和分类相关的应用提供支持。

我们在使用 **DATASTAX ENTERPRISE GRAPH**（DSE Graph）处理大型图数据库方面，已经取得了一些早期的成功。建立在 Cassandra 之上的 DSE Graph目标是大型数据集，而我们一直喜欢的 Neo4j 却开始显现出一些局限。这个规模需要做出权衡；例如，访问底层 Cassandra 的表就会失去 Neo4j 的 ACID 事务以及运行时的“模式自由”（schema-free）特性，集成 Spark 用于承担分析工作，使用强大的 TinkerPop/Gremlin 查询语言使以上组合成为了一个值得考虑的选项。

对于区块链和加密货币的炒作看来已经到顶，该领域的增长正在逐渐放缓，我们预计随着时间的推移，一些投机活动将会逐渐消失。而区块链技术中，**以太坊（ETHEREUM）**取得了良好的进展，值得关注。以太坊是公有区块链，允许使用其内置的编程语言创建“智能合约”。当有区块链交易发生时，就会进行“Ether”（以太坊的加密货币）的计算并响应结果。R3CEV，一个为银行构建区块链技术的组织，已经基于以太坊实现了概念证明（POC）。以太坊还被用来建立了一个分布式的自治组织（DAO）——首批“基于算法的公司”之一——虽然最近一起涉及价值1.5亿美元的 Ether 盗窃案件表明，区块链和加密货币仍然是技术世界的混乱地带。

超级账本（HYPERLEDGER）是一种围绕区块链技术构建的平台。它由一种名为 Fabric 的区块链实现以及相关的工具组成。抛开对区块链的炒作，我们的团队发现很容易利用这些工具启动项目。事实上，这是个由 Linux 基金会（Linux Foundation）支持的开源平台，这也是我们对超级账本如此兴奋的另一个原因。

KAFKA STREAMS 是一个构建“流应用”（streaming applications）的轻量级库。它的设计目的是将流处理简化得更加易于访问，从而作为异步服务的主流编程模型。如果你想要应用程序采用流处理模型解决问题，它是一个很好的选择，且无需面对运行集群的复杂性（其它相对重量级的流处理框架会引入这样的复杂性）。

微服务或任何其他分布式架构中，最常见的需求之一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，用于确保应用程序或微服务的安全。且几乎不需要编写代码，开箱即用。它支持单点登录、社交网络登录和标准协议登录（如 OpenID Connect，OAuth2 和 SAML 等）。

MESOSPHERE DCOS 是一个基于 Mesos 构建的平台。它将底层基础设施抽象出来，适用于容器化的以及没有运行在 Docker 内的应用程序。这可能对更多“适量部署”

(modest deployments)而言是过度的，但是我们开始看到它在商业版本和开源版本中的成功。我们尤其喜欢它在不同的云计算供应商和专用硬件之间的可移植性，因此可以使你摆脱对于单一容器编排框架的依赖。虽然升级可能会比我们想要的更复杂一点，但整个技术栈正在变得更加稳定。

在我们的经验中，对于包含大量设备之间或与中央数据中心相互通信的物联网(IoT)解决方案，MQTT 连接协议已经成为事实标准。我们也推荐像 **MOSQUITTO** 这样的 MQTT 中转器。它也许并不能满足所有要求，特别是在规模可扩展性方面，但紧凑和易于安装的特性使其非常适合用于开发和测试。

PLATFORMIO 为物联网开发提供了具有跨平台构建、依赖库管理和良好的 IDE 集成的生态系统。智能的代码补全，以及内置终端和串口监视器的智能代码静态检查器大大地增强了开发人员的使用体验。它还组织并维护了数千个依赖程序库并提供一个简洁的依赖项管理器，它使用语义化的版本控制来简化 IoT 应用开发。我们已经在多个 IoT 项目中使用了 PlatformIO，并且非常喜欢它的简单性和对各种平台以及开发板的广泛支持。

由于对硬件的要求和构造虚拟世界的复杂度门槛较高，除了虚拟现实(VR)之外，替代现实(AR)和混合现实(MR)也在去年进入主流。Pokémon Go 的风靡则证明了：普通的智能手机也足以创造引人瞩目的 AR/MR 体验。**TANGO** 是一种用于手机的新型硬件传感器技术，进一步增强了在手机上实现 AR / MR 的可能性。它允许应用程序获取用户周围的详细的 3D 测量数据，以便在摄像头输入流中放置和呈现更有说服力的虚拟对象。第一批使用 Tango 技术的手机现已上市。

诸如 Amazon Alexa 和 Google Home 这样的**语音平台**目前处在技术界的风口浪尖 技术成熟度曲线(hype cycle)的炒作顶峰，甚至有人预言，未来会话式的语音接口会无处不在。我们已经有把对话式UI集成到产品中的经验，并且看到了这种新的交互方式对接口设计的影响。Alexa 则全部从头设计，他们舍弃了屏幕并将会话式用户界面视为一等公民。但现在要相信这样的炒作还为时过早，我们期待更多的大厂商进入这个领域。

我们已经有把对话式UI集成到产品中的经验，并且看到了这种新的交互方式对接口设计的影响。

— Voice platforms

WEBVR 是一组可让你通过浏览器访问 VR 设备的实验性 JavaScript API。它已经获得了技术社区的支持，并有正式版本和每日构建的版本可用。如果你想在浏览器中构造 VR 体验，那么 WebVR 将会是一个不错的开始。此项技术以及相关补充工具，例如 Three.js，A-Frame，ReactVR，Argon.js 和 Awe.js 都能够为浏览器带来 AR 体验。除了互联网委员会标准以外，该领域中的各种工具也将有助于促进 AR 和 VR 更广泛的应用。

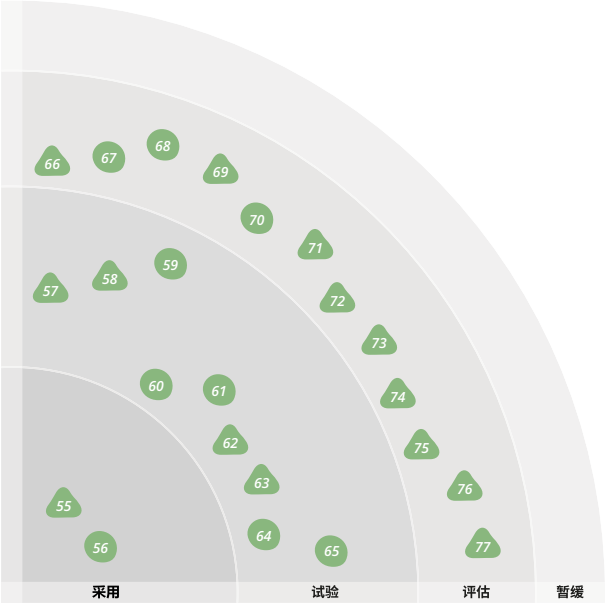
工具

采用
55. fastlane
56. Grafana

试验
57. Airflow *NEW*
58. Cake and Fake *NEW*
59. Galen
60. HashiCorp Vault
61. Pa11y
62. Scikit-learn
63. Serverless Framework *NEW*
64. Talisman
65. Terraform

评估
66. Amazon Rekognition *NEW*
67. Android-x86
68. Bottled Water
69. Claudia *NEW*
70. Clojure.spec
71. InSpec *NEW*
72. Molecule *NEW*
73. Spacemacs *NEW*
74. spaCy *NEW*
75. Spinnaker *NEW*
76. Testinfra *NEW*
77. Yarn *NEW*

暂缓



Web 应用程序开发者在简化和自动化各种应用程序的工作流程时很容易，他们可以从各种成熟的解决方案中选择最合适的方案来自动化发布流程。但是，当在开发移动应用程序时，我们需要处理两个不同的操作系统，处理两种完全不同的构建，测试，分发，生成屏幕截图，签名和发布应用程序的方式。为了解决这个痛点，我们的团队采用了 **FASTLANE** 作为自动化 iOS 和 Android 应用的发布流程的工具。通过一些简单的配置和多个发布流水线，他们就实现了移动开发的持续交付。

AIRFLOW 是一个用来通过编程创建、调度和监控数据流水线的工具。通过将有向无环图（DAG）以代码形式表现，它主张可维护、可版本化并且可测试的数据流水线。我们在项目中利用这一配置来创建动态流水线，使得数据工作流更加高效和清晰。Airflow 可以很容易的定义你自己的操作符和执行器来扩展库，以适配符合你的环境的抽象层次。

我们的团队采用了 fastlane 作为自动化 iOS 和 Android 应用的发布流程的工具。
— fastlane

MSBuild 自从2005年推出以来一直是 .NET生态系统中的主要构建系统。但是，它遇到了我们以前在 Maven 中提到的许多相同的问题。.NET社区已经开始开发MSBuild的替代品，它更容易维护且更加灵活，并能随着项目的增长更自然的演化。**CAKE** 和 **FAKE** 是其中的两个备选方案。Cake 使用一种 C# 内置的 DSL，而 Fake 使用 F# 。在过去的一年里这两个项目都取得了显著的增长，足以证明在 .NET 项目中它们是替代 MSBuild 编排常见构建任务的可行方案。

.NET 社区已经开始开发 MSBuild 的替代品，它更容易维护且更加灵活，并能随着项目的增长更自然的演化。

— Cake and Fake

SCIKIT-LEARN 并不是一个新工具（它将要迎来自己的十岁生日）；但却是一个在学术领域和主要科技公司之外新采纳的机器学习工具和技术。由于 Scikit-learn 提供了一整套健壮的模型和丰富的功能，所以它在将机器学习的概念和能力带给更广泛受众（并且通常是非专家）的过程中扮演了重要角色。

非常流行的 **SERVERLESS FRAMEWORK** 为无服务器风格架构的应用程序提供了项目脚手架和部署工具。它的大部分使用场景都是基于 AWS Lambda 以及相关 AWS 产品。Serverless Framework 为 JavaScript, Python, Java 和 C# 语言提供了项目模板，并拥有有一个活跃的社区为其贡献扩展插件。此外，它也向作为 AWS Lambda 替代品的 Apache 孵化器项目 OpenWhisk 提供支持。

AMAZON REKOGNITION 是我们在这次雷达其他地方提到的云端图像理解工具之一。我们喜欢它的地方在于 Amazon 采用了一种有点新颖的方法来让人脸在 AWS 中保持匿名（使用 GUIDS），以满足一些对于人脸识别的隐私担忧。

AWS Lambda 和 Amazon API Gateway 的组合对我们如何部署服务和 API 产生了巨大的影响。但是，即便是在这种体系的无服务器架构配置中，要将各部分串起来一同工作所需的配置也并不少。**CLAUDIA** 是一个用来自动部署用 Javascript 编写的 AWS Lambda 函数以及相应 API Gateway 配置的工具。它提供了一些合理的默认配置，我们的团队已经发现能利用它来快速地构建基于 Lambda 的微服务。

企业如何在给予交付团队自主性的同时，确保其部署的解决方案仍然安全并合规?如何确保服务器在部署后的运维生命周期中依然保持安全与合规?这些是 InSpec 尝试解决的问题。**INSPEC** 是受 Serverspec 启发而开发出来的基础架构测试工具，但是经过修改后，该工具更适合需要确保数以千计的服务器的合规的安全专家使用。各个测试可以组合成完整的安全配置文件，并支持命令行远程运行。InSpec 不仅对开发人员非常有用，还可以扩展到用于持续测试已部署的生产基础设施，从而向生产环境的QA迈进。

MOLECULE 旨在帮助开发和测试 Ansible 的 Role 。通过在虚拟机或容器上为正在运行的 Ansible Role 的测试构建脚手架，我们无需再手工创建这些测试环境。Molecule 利用 Vagrant，Docker 和 OpenStack 来管理虚拟机或容器，并支持 Serverspec、Testinfra 或 Goss 来运行测试。在 sequence facility model中的默认步骤包括：虚拟机管理，Ansible 语法静态检查，幂等性测试和收敛性测试。虽然这是一个相当年轻的项目，但我们看到了其蕴含的巨大潜力。

任何 Emacs 粉丝都会告诉你，Emacs 不仅仅是一个文本编辑器，它还是个字符映射应用程序的平台。在过去几年间，这个平台上有了新的发展，但我们认为 **SPACEMACS** 值得特别注意。Spacemacs 可以作为进入 Emacs 平台的敲门砖，它拥有新的键盘用户界面，简化的定制层以及 Emacs 软件包的分发管理。该项目的目标之一是通过将 Vim UI 与 Emacs 的内部可重新编程性结合起来，打造出这个世界上最好的编辑器。我们认为开发人员生产力工具是有效软件开发的重要组成部分。如果你暂时还没有考虑 Emacs，我们建议你看看 Spacemacs 是如何重新思考这个经典开发平台的。

SPACY 是一个 Python 编写的自然语言处理（NLP）库。它是个高性能的库，供开发者在生产环境中的使用。其 NLP 模型能够适配处理混合了表情符号和前后不一致的标点符号的文本。和其他 NLP 框架不同，spaCy 是一个可插拔的库，而不是一个平台；它的目标是产品级应用，而不是用于研究的模型训练。它能够很好的和 TensorFlow 以及其他 Python 人工智能生态环境中的工具结合。我们在企业环境中使用 spaCy 构建了一个搜索引擎，接受人类语言作为输入，并帮助用户做出业务决策。

Netflix 把旗下的 Spinnaker 开源了。它是一个微服务的持续交付平台。相比其他的 CI/CD 平台，**SPINNAKER** 将集群管理和烘培镜像部署当作头等功能来实现。它支持开箱即用的部署和多种云平台（例如 Google Cloud Platform，AWS和 Pivotal Cloud Foundry）的集群管理功能。可以把 Spinnaker 集成到 Jenkins 里来执行构建任务。我们喜欢 Spinnaker 在云端部署微服务的率性做法，可惜它的流水线只能通过用户界面而不是代码来创建。

由于基础设施工具的广泛使用，在如今项目中基础设施即代码的普及已不足为奇。伴随着这种趋势，对这些代码进行测试的需求也随之而来。无论服务器是通过手工配置的，还是 Ansible 、Puppet 或 Docker 构建的，使用 **TESTINFRA** 都可以方便地测试服务器的实际状态。Testinfra 的目标是成为 Serverspec 在Python中的等价物，并且作为 Pytest 测试引擎的插件来使用。

由于基础设施工具的广泛使用，在如今项目中基础设施即代码的普及已不足为奇。
— Testinfra

YARN 是一个新的包管理工具，它可替换现有 npm 客户端的机制，同时兼容 npm 注册表。如果使用 npm 客户端，根据依赖库的不同安装顺序，它会在 node_modules 下得到一个不同的树结构。这种非确定性的特点可能导致“在我的机器上能工作”的问题。通过将安装步骤分解为解析、获取和链接，Yarn 使用确定性算法和 lockfiles 避免了这些问题，从而确保重复安装的一致性。因为它对已经下载的包进行缓存，我们还看到在持续集成（CI）环境中的构建速度明显更快。

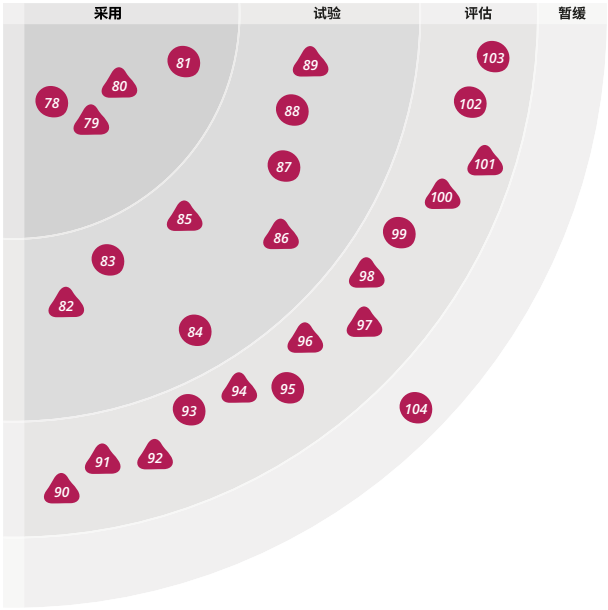
语言&框架

采用
78. Ember.js
79. Python 3
80. ReactiveX
81. Redux

试验
82. Avro *NEW*
83. Elixir
84. Enzyme
85. Hangfire *NEW*
86. Nightwatch *NEW*
87. Phoenix
88. Quick and Nimble
89. Vue.js

评估
90. Angular 2 *NEW*
91. Caffe *NEW*
92. DeepLearning.scala *NEW*
93. ECMAScript 2017
94. Instana *NEW*
95. JuMP
96. Keras *NEW*
97. Knet.jl *NEW*
98. Kotlin *NEW*
99. Physical Web
100. PostCSS *NEW*
101. Spring Cloud *NEW*
102. Three.js
103. WebRTC

暂缓
104. AngularJS



PYTHON 3 引入了很多有用的特性，这些特性和 Python 2.x 不兼容。它还移除了大量 Python 2.x 中用于向后兼容的功能，这让 Python 3 更容易学习和使用，而且和语言的其他部分更加一致。根据我们在机器学习和 web 应用开发这样的领域中使用 Python 3 的经验显示，语言本身以及大多数**支持库**都已经成熟到可被采用的程度。我们可以 fork 已有的库并为其存在的小问题打补丁，或者避免使用已经被放弃的不兼容的 Python 2.x 库。如果你在使用 Python 做开发，我们强烈鼓励你使用 Python 3。

分布式系统经常利用多线程、基于事件的通信和非阻塞 I/O 来提高整体系统效率。这些编程技术提出了诸如低级线程、同步、线程安全、并发数据结构和非阻塞 I/O 等挑战。开源的 **REACTIVEX** 库优雅地解决了这些问题，提供了所需的应用程序管道，并扩展了异步事件流之上的观察者模式。ReactiveX 还拥有一个活跃的开发者社区，支持的编程语言越来越多，最近支持的是 RxSwift 。它还实现了绑定到移动和桌面平台的功能。

> REACTIVEX 库优雅地解决了这些问题，提供了所需的应用程序扩展，并扩展了异步事件流之上的观察者模式。
>
> — ReactiveX

AVRO 是一个数据序列化的框架。它通过将 schema 与消息内容共同存放的方式来鼓励 schema 演进。生产者可以编辑字段名称，添加新字段或删除现有字段，而 Avro 确保客户端可以继续消费消息。Schema 允许每个数据被写入而没有额外开销，促成了紧凑的数据编码和更快的数据处理。尽管生产者和消费者之间无结构消息的交换形式可以很灵活，但我们已经看到团队在部署期间遇到队列中出现无法处理的不兼容消息的问题。我们在许多项目中使用了 Avro ，并且建议仅在发送非结构化消息时使用它。

> Hangfire既易用又灵活，并且拥有函数式风格。特别有趣的是它能够保存任务的状态，以便在应用程序在崩溃或关闭后重新启动时恢复。
>
> — Hangfire

应用程序开发中的一个常见问题是如何调度在主进程之外定期或在满足某些条件时运行的任务。当意外事件（如应用程序关闭）发生时，问题会变得更加复杂。我们的团队发现，**HANGFIRE** 框架可以在 .NET 环境中很好地做到这点。Hangfire 既易用又灵活，并且拥有函数式风格。特别有趣的是它能够保存任务的状态，以便在应用程序在崩溃或关闭后重新启动时恢复。

NIGHTWATCH 是一个框架，它支持用 JavaScript 创建基于浏览器的自动化验收测试、并用 Node.js 运行这些测试。Nightwatch 可以用流畅的 API 定义测试，然后在 Selenium / WebDriver 服务器上执行。在单页应用或其他重 JavaScript 页面的场景下，这可以让自动化测试和大多数代码一样在相同的语言和环境里创建并运行。

在日新月异的前端 JavaScript 框架世界里，**VUE.JS** 作为 AngularJS 的轻量级替代品占据了一席之地。它是一个非常灵活——且没有预设主张——的库。它围绕着模块化、组件和响应式数据流等概念，为构建交互式 Web 界面提供了一套工具集。它的学习门槛很低，这让初级开发者和新手很感兴趣。Vue.js 本身并不是一套大而全的框架。它仅专注在视图层，因而可以轻松地和其他库或现有项目做集成。

在上期雷达中，我们将 AngularJS 移动到暂缓环（在这个版本中仍然是暂缓）。当谈到 **ANGULAR 2** 时，我们看到了混合的信息。在过去一年中成功使用了 Angular 2 的一些 ThoughtWorks 团队认为它是一个不错的选择。然而，Angular 2 是 AngularJS 的重写而不是演进，从 AngularJS 切换到 Angular 2 与从 AngularJS 切换到另一个框架没有太大的不同。考虑到我们对 React.js，Ember.js 和 Vue.js 等其他优秀竞争者的使用经验，我们仍在犹豫是否要强烈推荐 Angular 2。但我们要强调，这不是一个坏的选择，特别是如果你采用了 TypeScript。

CAFFE 是一个用于深度学习的开源库，由伯克利视觉和学习中心开发。它主要专注于计算机视觉应用的卷积网络。对于计算机视觉相关的任务，Caffe 是一个可靠并且流行的选择，而且可以从 Caffe Model Zoo 下载很多Caffe用户创建的开箱即用的成功模型。与 Keras 一样，Caffe也是一个基于 Python 的 API。它们的不同之处在于，Keras 中的模型和组件是在 Python 代码中直接被创建的对象，而 Caffe 的模型是用 Protobuf 配置文件描述的。这两种方式各有其优缺点，并且可以相互转换。

DEEPLEARNING.SCALA 是一个 Scala 编写的开源深度学习工具箱，由我们 ThoughtWorks 的同事创建。我们对这个项目感到兴奋的地方在于，它使用了可微分函数式编程来创建和组合神经网络。开发者只需要编写具有静态类型的 Scala 代码。DeepLearning.scala 目前支持基本类型如 float、double、使用 GPU 加速的 N 维数组以及代数数据类型。我们很期待该工具箱的未来版本，据说会支持高阶函数，以及在 Spark 上进行分布式训练。

INSTANA 是拥挤的应用程序性能管理领域中的又一个新成员。事实上，INSTANA 是一个从头搭建的云原生架构，这让它在众多竞争者中别具一格。它的功能包括动态发现、分布式跟踪和服务健康，以及将你的基础设施视图回退到事件发生时的能力。至于这项产品是否能比开源项目组合（如 Consu、Prometheus 和 OpenTracing 的实现）更有吸引力，还需拭目以待。然而，如果你需要一种开箱即用的解决方案，那么 Instana 值得一看。

KERAS 是一个用 Python 编写的用于搭建神经网络的高阶接口。Keras 是由一名 Google 工程师创建的开源项目，能够运行在 TensorFlow 或者 Theano 之上。它提供了极度简洁的接口来创建强大的深度学习算法，可以在 CPU 或者 GPU 上进行训练。本着模块化、简洁性以及可扩展性的理念 Keras 得到了良好的设计。和 Caffe 这样的框架不同，Keras 支持更多通用的网络架构，如递归网络，这让其在文本分析、自然语言处理以及通用机器学习上更加有用。如果计算机视觉或其他机器学习的特定分支是主要关注点，也许 Caffe 是一个更合适的选择。然而，如果想要学习一个简单但强大的框架，Keras 应该是第一选择。

KNET.JL 是土耳其 Koc 大学基于 Julia 实现的深度学习框架，由 Deniz Yuret 及其合作者开发。不像诸如 Theano 和

TensorFlow 这些生成梯度的编译器将用户强制在一个受限的迷你语言之内，Knet 运用 Julia 的完整能力和表达力来定义和训练“机器学习模型”。Knet 使用运行时生成的动态计算图在任意 Julia 代码中实现自动求导。我们很喜欢通过 KnetArray 类型提供的 GPU 操作支持，如果你没有 GPU 服务器，Knet 团队还维护了一个预配置的 Amazon 机器镜像 (AMI)，以便于在云中使用和评估。

今年我们很多开发者把 **KOTLIN** 编程语言放到了他们的评估列表中，有些人甚至已经在生产环境中成功使用它了。Kotlin 是 JetBrains 的一个开源且基于JVM的语言。我们的Swift移动开发者喜欢它，因为它在语法上更接近 Swift，并且同样简洁。我们的 Java 开发者也喜欢它，因为它与 Java 语言和工具之间可以无缝互操作，并发现它比 Scala 更容易学习。Kotlin 支持函数式编程概念，但特性比 Scala 少。Kotlin的编译器能够辅助检测代码中的空指针缺陷，我们团队中的开发者非常喜欢这一点，因为这项特性可以帮助他们少写很多模板测试。

团队构建微服务系统时需要考虑服务协调技术 (coordination techniques)，如服务发现、负载均衡、熔断和健康检查。

— Spring Cloud

POSTCSS 是一个基于 Node.js 的 JavaScript 框架，它有繁荣的插件生态圈，能够操作基于抽象语法树的 CSS 文件。PostCSS 常常被误认为是一种预处理器 (如 SaaS 或者 Less)，然而我们发现，它的实力来源于其丰富多样的插件所提供的功能，包括语法检查 stylelint 插件、交叉编译 sugarss 插件)、命名改编以避免选择器冲突 (modules 插件)、模板 CSS 代码生成 (autoprefixer 插件)、文件压缩等等。尽管插件的成熟度各不相同，PostCSS 本身仍然是一个简洁而强大的前端开发框架，它能够像对待一个完整前端开发语言一样处理 CSS 。

团队构建微服务系统时需要考虑服务协调技术 (coordination techniques)，如服务发现、负载均衡、熔断和健康检查。这些技术有许多都需要团队来搭建工具，而这些工作往往并非易如反掌。**SPRING CLOUD** 项目为开发者提供了一系列工具，以便他们可以在所熟悉的 Spring 技术栈下使用这些服务协调技术。这些工具支持 Consul，ZooKeeper 和 Netflix 全栈开源软件技术栈。简单来说，使用这些工具集能让做正确的事情变得容易。尽管我们通常对 Spring 隐藏了太多复杂性的顾虑依然存在，但是如果你处于这个生态圈中，并且需要解决这些问题，你应该考虑 Spring Cloud 。

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4000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14个国家设有40间办公室：澳大利亚、巴西、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。

最先获悉技术雷达发布消息
了解最新独家研讨会及内容

点击订阅

thoughtworks.com/cn