ThoughtWorks®

TECHNOLOGY RADAR *NOV '16*

Our thoughts on the technology and trends that are shaping the future

微信公众号

技术雷达官网

thoughtworks.com/radar

#TWTechRadar

最新动态

本版精彩集锦

“容器即进程，PaaS即机器，微服务架构即编程模式”

由于容器化和强调松耦合，微服务风格的架构呈现了一个更抽象的开发者世界，这提供了更高层次的运行隔离。开发者可以设想容器是一个独立进程，PaaS是一个公共部署目标，微服务架构作为一致的风格。 架构上的解耦同样适用于团队，以降低协调成本。 它们对开发者和DevOps的吸引力，使之正成为许多组织事实上的新一代开发标准。

智能释放的力量

长期待在实验室的机器学习和人工智能，突然通过框架进入到实用领域：如Nuance Mix和TensorFlow。 从NLP到机器学习库，开发者都能从框架中下载。 我们高兴地看到，商业公司在这个领域频繁开源了复杂的库和工具，使得开发者能够广泛地加以应用；而在十年前获取这些知识的代价是非常昂贵甚至是受限的。 这些因素的综合演变将会促成一系列新的工具：商品计算（一种大规模、低成本、可伸缩的集群计算标准），特殊定制的硬件如GPUs，以及云端资源。 也许你将开始从屯积的大数据获得回报……

团队结构的全局影响

团队结构永远是软件行业极具影响力的话题，组织结构如何为自助式PaaS平台和微服务提供良好支撑，已成为日益关注的焦点。目前商业公司倾向于产品思维高于项目运作；科技公司正在推行“谁构建，谁运行”玩法的自治团队，我们看到这些产品思维也被应用到企业级项目。 当重组团队能产生更好的结果时，无疑再次证明，软件开发需解决的首要问题还是沟通。 全功能团队极利于改善传统组织的跨部门沟通，并减少人为产生的部门冲突。

AR/VR渐入佳境

增强现实和虚拟现实（AR/VR）正在引发企业的兴趣，过去这两项技术仅仅和游戏及新鲜感联系在一起。先是基于移动SDKs开发的夜跑引起了公众对AR的热情，随后硬件设备如Oculus Rift、HTC Vive和微软HoloLens日趋成熟，预示着技术已度过不成熟期，先行者将获得优势。虽然像OpenVR和Unity这样的软件开发平台已经非常成熟，但新的自然语言处理（NLP）工具如Nuance Mix，还有硬件提供的接近自然的交互，为AR/VR技术的采用提供了巨大的助力。我们建立了AR/VR实验室来探索下一代应用，如远程交互和零售业导购。我们的实验表明，VR在远程协作和讲述时有惊人的移情作用，这得益于它通过抽象介质向用户直接传递的沉浸式体验。然而，我们仍然觉察到挑战：创作和交付VR/AR内容应用的技能和能力，远远跟不上硬件发展的步伐，尤其是在企业应用领域。

贡献者

ThoughtWorks技术顾问委员会由以下人员组成：

Rebecca Parsons (CTO)
Martin Fowler (首席科学家)
Anne J Simmons
Badri Janakiraman
Bharani Subramaniam
Camilla Falconi Crispim
Erik Doernenburg
Evan Bottcher
Fausto de la Torre
Hao Xu
Ian Cartwright
James Lewis
Jiaxing Chen
Jonny LeRoy
Marco Valtas
Mike Mason
Neal Ford
Rachel Laycock
Scott Shaw
Srihari Srinivasan
Zhamak Dehghani

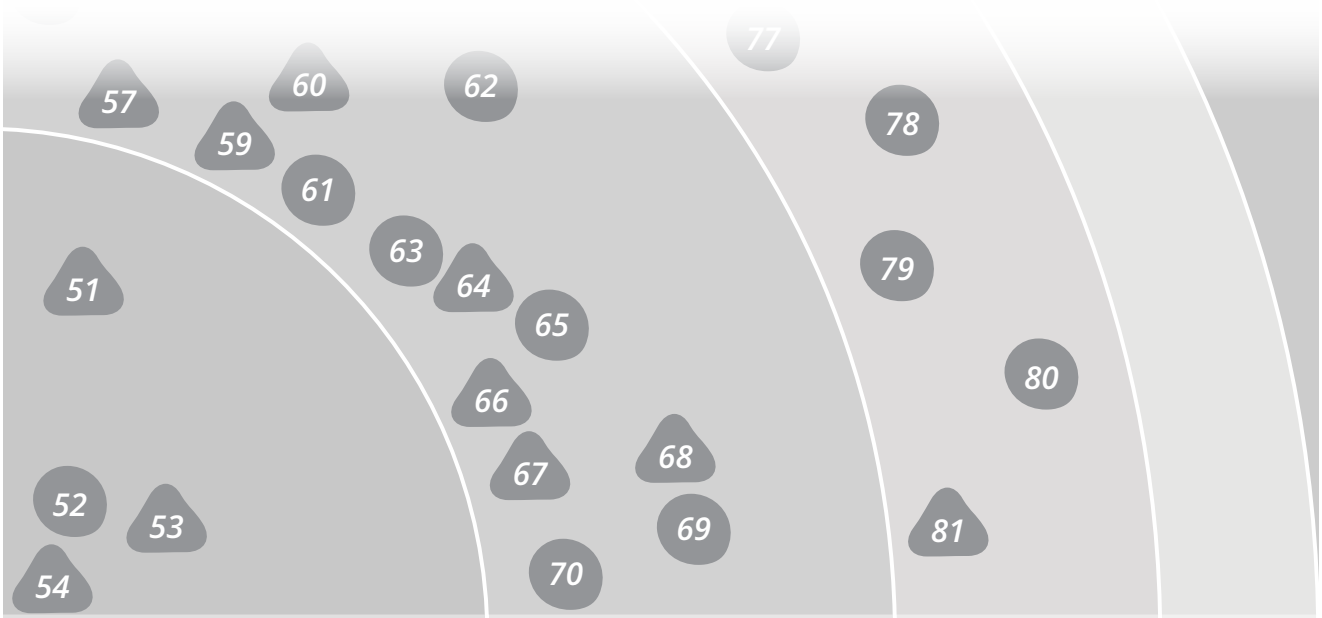
技术雷达中国区技术咨询顾问组：

陈加兴
黄雨青
林帆
刘先宁
刘尚奇
覃宇
伍斌
王威
鄢倩
禚娴静
朱傲

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、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和掀起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，从首席信息官到开发人员， 雷达为各路利益相关方提供价值。这些内容只是简要的总结，我们建议您探究这些技术以了解更多细节。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。这四个环是:



我们强烈主张业界采用这些技术。我们会在适当时候将其用于我们的项目。

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。

为了确认它将如何影响您所在的企业，值得作一番探究。

谨慎推行。

自上次雷达发表以来新出现或发生显著变化的技术以三角形表示，而没有变化的技术则以圆形表示。每个象限的详细图表显示各技术发生的移动。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，因此我们略去了上期雷达中已包含的许多技术，为新技术腾出空间，略去某项技术并不表示我们不再关心它。

要了解关于雷达的更多背景，请参见 thoughtworks.com/radar/#/faq

THE RADAR

技术

采用
1. Consumer-driven contract testing
2. Pipelines as code new
3. Threat Modeling

试验
4. APIs as a product new
5. Bug bounties
6. Data Lake
7. Hosting PII data in the EU
8. Lightweight Architecture Decision Records new
9. Reactive architectures
10. Serverless architecture

评估
11. Client-directed query new
12. Container security scanning new
13. Content Security Policies
14. Differential privacy new
15. Micro frontends new
16. OWASP ASVS
17. Unikernels
18. VR beyond gaming

暂缓
19. A single CI instance for all teams
20. Anemic REST new
21. Big Data envy
22. Cloud lift and shift

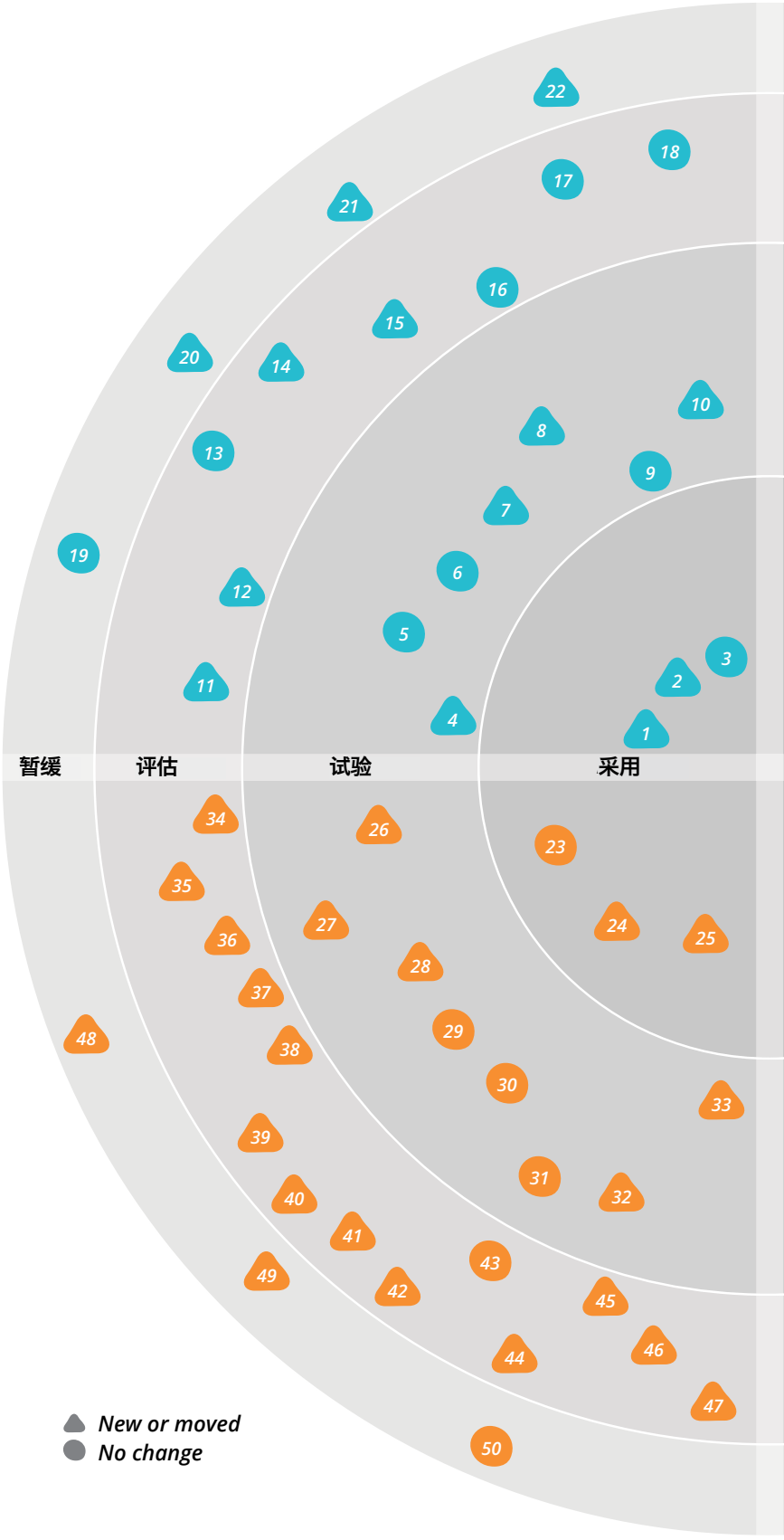
平台

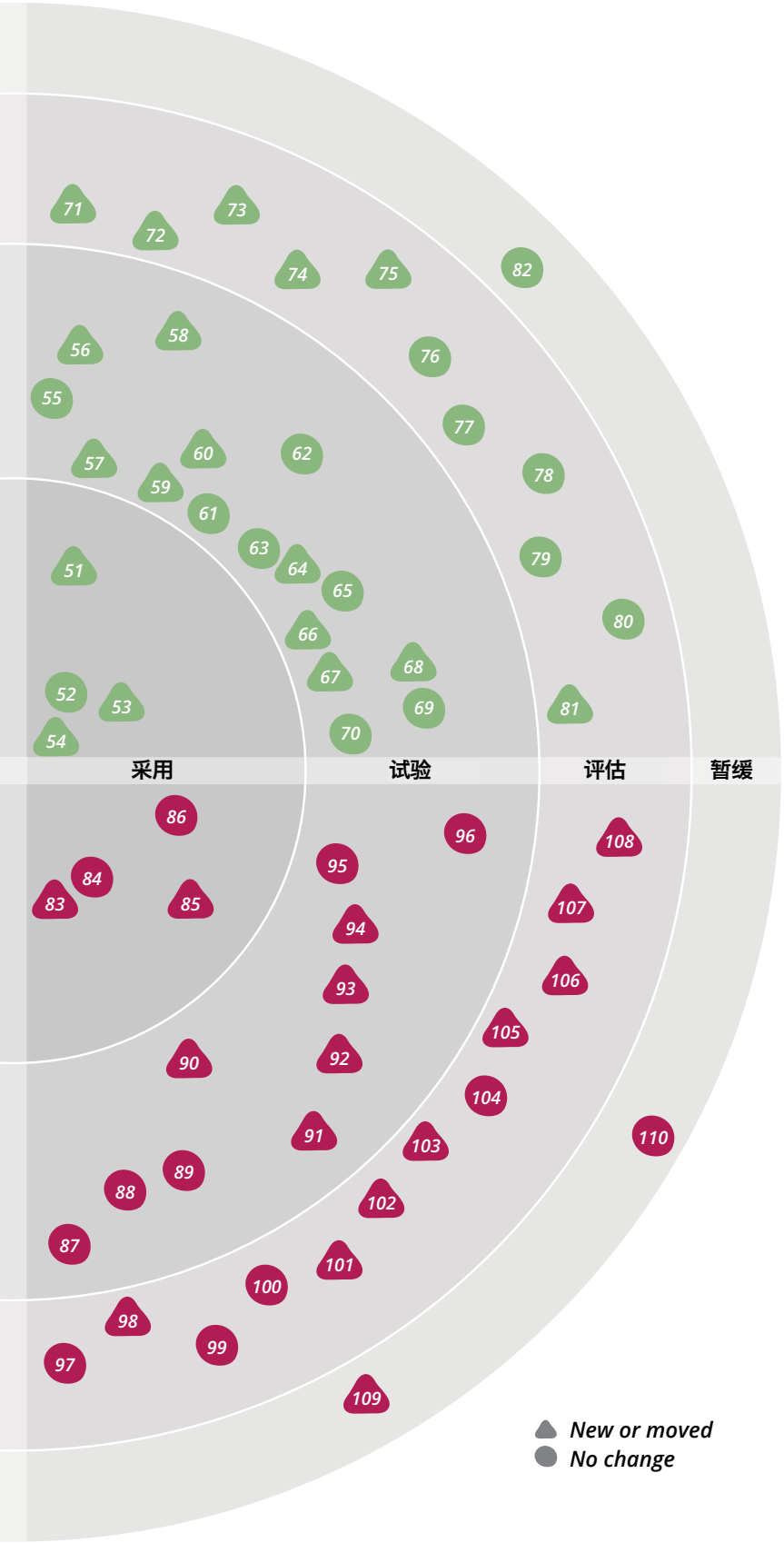
采用
23. Docker
24. HSTS
25. Linux security modules

试验
26. Apache Mesos
27. Auth0 new
28. AWS Lambda
29. Kubernetes
30. Pivotal Cloud Foundry
31. Rancher
32. Realm
33. Unity beyond gaming new

评估
34. .NET Core
35. Amazon API □□
36. Apache Flink
37. AWS Application Load Balancer new
38. Cassandra carefully new
39. Electron new
40. Ethereum new
41. HoloLens new
42. IndiaStack new
43. Nomad
44. Nuance Mix new
45. OpenVR new
46. Tarantool new
47. wit.ai new

暂缓
48. CMS as a platform
49. Overambitious API □□
50. Superficial private cloud





THE RADAR

工具

采用
51. Babel new
52. Consul
53. Grafana new
54. Packer

试验
55. Apache Kafka
56. Espresso
57. fastlane new
58. Galen new
59. HashiCorp Vault
60. JSONassert new
61. Let's Encrypt
62. Load Impact
63. OWASP Dependency-Check
64. Pa11y new
65. Serverspec
66. Talisman new
67. Terraform
68. tmate new
69. Webpack
70. Zipkin

评估
71. Android-x86 new
72. axios new
73. Bottled Water new
74. Clojure.spec new
75. FBSnapshotTestcase new
76. Grasp
77. LambdaCD
78. Pinpoint
79. Pitest
80. Repsheet
81. Scikit-learn new

暂缓
82. Jenkins as a deployment pipeline

语言和框架

采用
83. Ember.js
84. React.js
85. Redux
86. Spring Boot

试验
87. Butterknife
88. Dagger
89. Dapper
90. Elixir
91. Enzyme new
92. Immutable.js
93. Phoenix new
94. Quick and Nimble new
95. React Native
96. Robolectric

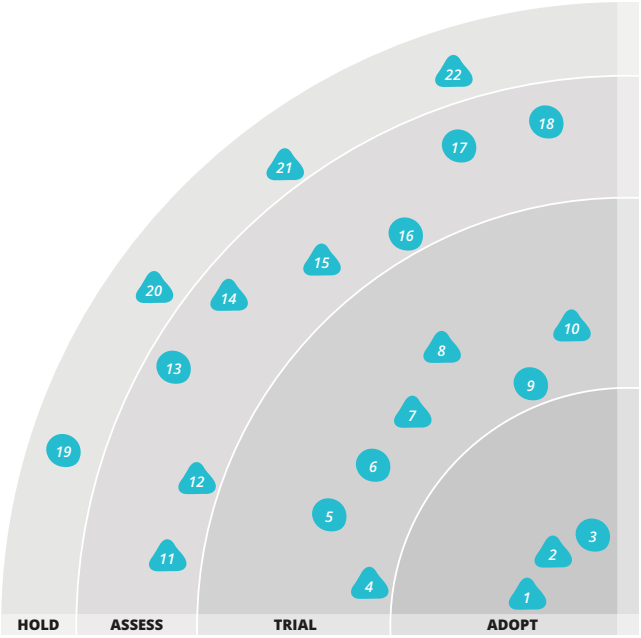
评估
97. Aurelia
98. ECMAScript 2017 new
99. Elm
100. GraphQL
101. JuMP new
102. Physical Web new
103. Rapidoid new
104. Recharts
105. ReSwift new
106. Three.js new
107. Vue.js new
108. WebRTC new

暂缓
109. AngularJS
110. JSPatch

技术

虽然**消费者驱动的契约测试**在过去几期雷达中淡出，但这一期我们决定将它拿回来。这并不是一个新概念，但随着主流市场接受微服务，我们要提醒大家，消费者驱动的契约在一个成熟的微服务测试组合中是非常重要的组成部分，利用它可实现服务的独立部署。但此外我们要强调的是，消费者驱动的契约测试本质上是一种技术和态度，并不需要特定的工具来实现。我们非常喜欢Pact这样的框架，因为他们在某些情况下让契约测试更容易实现。但是我们注意到团队过于关注框架而忽略实践本身的趋势。编写Pact测试并不能保证创建消费者驱动的契约；同样，即使没有现成的测试工具，我们也应该致力于构建良好的消费者驱动契约。

团队在推进跨环境的自动化，包括开发的基础设施。**流水线即代码**通过编码而非配置CI/CD运行工具的方式，来定义部署流水线。LambdaCD，Drone，GoCD和Concourse等都是这一技术的应用案例。此外，像 GoMatic这样的CI/CD系统自动化配置工具，可以对部署流水线即代码进行版本化和测试。



企业已经全力拥抱通过APIs将业务能力暴露给内外部开发者。基于APIs可将现有核心能力进行整合，快速验证新的业务创意。但APIs跟传统的企业集成服务有什么区别呢？一个区别在于**将APIs当作产品**，即使消费者是一个内部系统。APIs团队应该理解客户需求，为他们提供有吸引力的产品。从长期看，产品应该得到改进、维护和支持；指定专门的负责人为客户发声，并努力持续改进。产品还应该被积极地维护和支持，易于找到和易于使用。根据我们的经验，企业集成服务缺乏产品导向，是与基于APIs的敏捷业务最大的区别。

在许多国家和地区，我们看到政府机构在寻求广泛地使用个人身份和隐私信息（PII）。公有云解决方案的普遍使用，让组织更难保护其用户委托的数据，以及遵守相关法律。欧盟拥有一些最先进的隐私法律，主要的云服务提供商——亚马逊，谷歌，微软——也都在欧洲建立了多个数据中心和区域服务。因此我们建议公司，尤其是那些用户遍布全球的公司，通过**托管PII数据在欧盟**，来考察用户数据的安全度。在把这项技术列入上期雷达之后，我们推出了一个管理员工敏感信息的内部系统，并把它托管在了一个位于欧盟的数据中心上。

虽然易读的代码和测试可以代替开发过程中的大部分文档，但对于演进式架构来说，对确定的各种设计决策进行记录依然非常重要，这不仅有利于未来的团队成员理解，也有利于外部监督。**轻量级的架构决策记录**是用来记录重要的架构决策、发生的上下文和对应产生的结果的一种技术。虽然这些内容经常被保存在团队wiki以及协作工具当中，但我们通常更倾向于使用简单的markup文件来将它们保存在源代码控制工具中。

采用
ADOPT
1. Consumer-driven contract testing
2. Pipelines as code
3. Threat Modeling

试验
TRIAL
4. APIs as a product
5. Bug bounties
6. Data Lake
7. Hosting PII data in the EU
8. Lightweight Architecture Decision Records
9. Reactive architectures
10. Serverless architecture

评估
ASSESS
11. Client-directed query
12. Container security scanning
13. Content Security Policies
14. Differential privacy
15. Micro frontends
16. OWASP ASVS
17. Unikernels
18. VR beyond gaming

暂缓
HOLD
19. A single CI instance for all teams
20. Anemic REST
21. Big Data envy
22. Cloud lift and shift

技术 接上页

无服务器架构是一种架构方法，使用即时请求、用后即销毁的短暂计算能力来取代长期运行的虚拟机。从上一期技术雷达开始，我们已经有若干个团队在正式产品中应用了“无服务器架构”风格。我们的团队喜欢并适应这种方式，我们认为这是一种有效的架构选择。需要指出的是，无服务器架构并非一种绝对的架构风格：我们某些团队将系统中的一部分采用无服务器架构，而其它部分继续采用传统架构。

设计RESTful风格APIs遇到的诸多问题，都可以归咎于贫血REST这种反模式，一些场景也证明了需有更多解决方法。特别是，当组织必须支持一些长尾客户应用时（大量增长的API版本，即使已经使用了消费者驱动的契约测试——大部分的APIs都要支持无尽的活跃订阅端数目 —— 也许触及到了RESTful架构的极限。这些问题可以使用**客户直接查询**来解决。我们已经在GraphQL和Falcor中看到了成功案例：这项技术可以让客户在内容和反馈数据的粒度两个方面掌握更多的控制权。这将更多的职责推到了服务层，并且会导致数据模型的强耦合，但当设计良好的RESTful APIs不工作的时候，还是值得尝试。

由Docker掀起的容器革命，大大降低了应用在不同环境迁移的难度，但也打开了生产环境的管控漏洞。**容器安全检查**作为一种解决安全风险的技术，是不可或缺的。Docker目前已经提供了它自己的安全检查工具，CoreOS也提供了这样的工具，我们已成功使用了CIS Security Benchmarks。无论你采用那种技术，我们相信容器自动化安全检查很有价值，同时也是PaaS战略中必须考虑的一项内容。

众所周知，那些“匿名”的海量数据是可以泄露个人信息的，尤其是将多个海量数据集做交叉引用的时候。正像 increasing concern over personal privacy所指出的那样，以苹果公司、谷歌公司为代表的一些公司，开始将目光转向**差分隐私**技术来实现在针对大量用户行为进行高效分析的同时，提升对个人隐私信息的保护。差分隐私是一种可以在控制访问样本的数量的情况下，提升统计分析结果准确度的加密技术，这是通过在目标数据中加入一些少量的“噪音”而实现的。当然需要指出，这项技术目前仍然在研究阶段。苹果公司已经宣布要将差分隐私技术纳入到苹果的产品当中——我们由衷的赞赏像这样看重用户隐私的举措——但通常，苹果的保护用户隐私的举措总是令安全专家们挠头。我们将继续推荐客户使用Datensparsamkeit作为一种简单的替代方案：仅仅保存分析所需的最小量的数据，在大部分情况下将能更好的保护用户的隐私。

我们从微服务架构的引入中获益匪浅，它允许团队规模化地交付那些能够独立部署和维护的服务。然而，在前端，团队经常做了很多努力来避免创建一个大型的单体和庞大的浏览器应用程序，就如我们已经放弃的服务器端单体应用一样，这些应用程序难以维护和演化。我们看到一种方法正在浮现，我们的团队将其称之为**微前端**。在这种方法中，一个Web应用程序可以分解为页面和特性，每个特性由一个单独的团队从端到端对其负责。现存的很多种技术可以将这些应用特性组织在一起，从而提供一个统一内聚的用户体验。但是微前端的目标仍然是允许特性之间彼此独立，每个特性可以独立地开发、测试和部署。BFF - 后端服务前端的方法也可以很好地应用于这个场景下，每个团队可以开发一个BFF以支持其一系列的应用程序特性。

随着BFF - 后端服务前端模式和单向数据绑定框架，如React.js的日益普及，我们注意到很多人对REST风格架构表现出反感。批评者们指责REST导致了系统之间繁琐低效的交互，且无法适应客户端需求的变化。他们提出了一些框架，如 GraphQL或Falcor作为数据获取机制的替代方法，它们可以让客户端指定返回的数据格式。但基于我们的经验，我们认为这些问题并不是REST引起的。相反，它们源于未能将领域作为一组资源来正确地建模。通过模板化的URL、简单地暴露静态分层数据模型来开发一个服务，会导致实现出**贫血REST**。在一个建模良好的领域中，REST应该支持的不仅仅是简单重复的数据获取。在一个经过了充分演进的RESTful架构中，业务事件和抽象概念同样能被建模为资源，并且它的实现应当能有效地使用超文本、链接关系和媒体类型，从而最大程度地将各个服务解耦。贫血REST是一个反模式，它与贫血领域模型模式密切相关，根据它所设计出来的服务，在Richardson成熟度模型中，会处于成熟度较低的层次。在我们的洞见文章《REST API设计资源模型》中提供了更多关于如何有效地设计REST API的建议。

技术 接上页

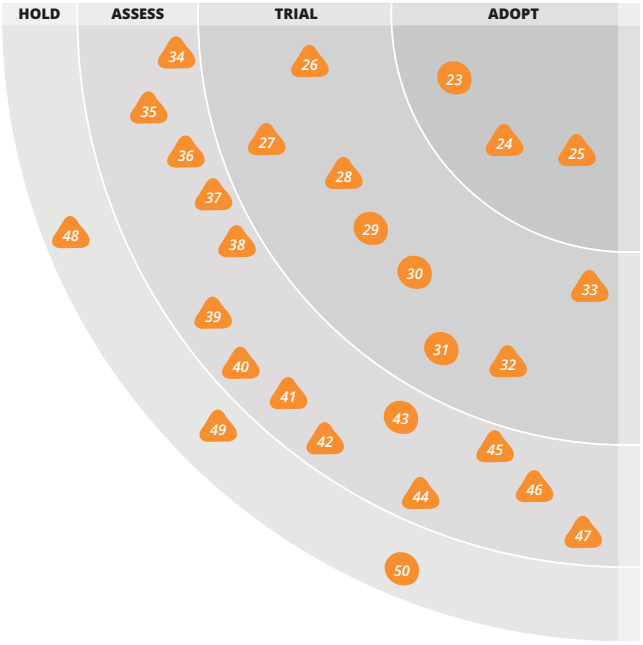
我们不断地看到很多组织喜欢追逐“酷炫”的技术，原本用更简单的方法就能做得更好的事情，偏偏要选择不必要的复杂性和风险。特别是在相对较小的数据集上，使用分布式的大数据系统。此行为提示我们再次将**大数据盲从**放在暂缓的维度，并根据我们最近的经验提供了一些额外的有关数据的观点。 Apache Cassandra数据库承诺在产品硬件上提供巨大的可扩展性，但我们看到其架构和操作复杂性使得团队不堪重负。除非数据量需要多于100个的集群节点来保存，否则不建议使用Cassandra。另外还需考虑在保持它们的正常运转所需的运维团队的投入是不是值得。在产生这个版本的雷达时，我们讨论了几种新的数据库技术。相比现有系统，许多技术能实现“10倍”的性能提升。我们总是对其持怀疑态度，直到新技术——尤其是像数据库一样关键的技术——被合理地证明。Jepsen能够在困难的情况下对数据库性能进行分析，并在各种NoSQL数据库中发现了大量 bugs。我们建议保持健康的怀疑态度，并在评估数据库技术时持续关注一些网站来获取它们的发展，如Jepsen。

随着越来越多的组织选择在云计算平台上部署应用程序，我们经常发现IT团队在这些云平台上试图去复制其现有的数据中心的管理和安全方法。这通常以防火墙、负载平衡器、网络代理、访问控制、安全应用和服务的形式，扩展到云平台，而并没有经过大量的重新思考。我们也看到组织在云提供商所提供的服务的之外，又构建了他们自己的编排API，来限制团队使用这些服务。

在绝大多数场景下，这一层的服务只会削弱云平台的能力，而丧失了大部分迁移到云计算的好处。在这一版的雷达中，我们选择重新强调**云计算平移**是应当避免的一个技术。组织应该更深入地了解他们现有的安全和运维控制的意图，然后在云平台现有的服务中，去寻找适用的控制功能，而不要自己创建不必要的限制。这些控制功能中的大部分，都已经被成熟的云提供商所提供。采用云平台的团队，可以使用云平台原生API进行自助配置和运维。

平台

HTTP严格安全传输（**HSTS**）是目前被广泛支持的策略，它使网站免于降级攻击。HTTPS降级攻击使网站的用户避开HTTPS使用HTTP，从而使如中间人攻击等进一步的攻击成为可能。通过HSTS，服务器发送头文通知浏览器必须使用HTTPS访问当前网站。目前浏览器的支持已经非常广泛，任何使用HTTPS的网站都应该考虑实施这个简单易用的功能。Mozilla推出的线上网站安全扫描工具Observatory可以帮助识别出这些有助于提升安全和隐私的头信息和配置选项。实施HSTS时，必须严格的检查所有资源都应通过HTTPS加载，因为一旦开启HSTS特性，就几乎不可能回退到原先的访问方式了。网站的子级域名也应该被加入，并且，应检查所有子级域名都已经支持HTTPS。



应用程序白名单已经被证明是减少网络入侵最有效的方法之一。其中**Linux的安全模块**是一种易于实现且被广泛推荐的实践。如今大多数Linux发行版已经内置SELinux或AppArmor这样的特性，像Grsecurity这些功能全面的工具也变得促手可及，因此我们在这个版本的雷达中将该技术移到了“采纳”的区域里。这些工具能帮助团队分析哪些用户在共享的主机上访问了哪些资源，包括运行在容器中的服务。这种保守的访问管理方式使得团队能够将安全因素纳入系统开发生命周期的流程里。

我们依然积极使用**Apache Mesos**管理分布式系统的集群资源。Mesos通过抽象出底层的CPU和存储等计算资源，从而在保持运行环境隔离的情况下提供良好的资源利用率和效率。Mesos包含了Chronos作为可容错的分布式定时任务执行器，以及Marathon来调度长时间运行在容器中的进程。

我们逐渐相信在大多数情形下，已经不再值得自行设计开发一整套鉴权功能代码。将鉴权管理的事情交给第三方服务不仅能加速交付的速度，还将减少错误的发生，并且更快速的响应新出现的安全漏洞。这方面**Auth0**提供的服务以其易于集成、广泛的协议和连接器支持，以及丰富的管理API尤其让人印象深刻。

我们的团队在继续享受**AWS Lambda**与API 网关结合来构建无服务器架构，但建议保持Lambda函数代码的简洁。因为对于庞大的Lambda函数代码，质量很难保障，托管运行的费用可

采用

ADOPT

23. Docker
24. HSTS
25. Linux security modules

试验

TRIAL

26. Apache Mesos
27. Auth0
28. AWS Lambda
29. Kubernetes
30. Pivotal Cloud Foundry
31. Rancher
32. Realm
33. Unity beyond gaming

评估

ASSESS

34. .NET Core
35. Amazon API 网关
36. Apache Flink
37. AWS Application Load Balancer
38. Cassandra carefully
39. Electron
40. Ethereum
41. HoloLens
42. IndiaStack
43. Nomad
44. Nuance Mix
45. OpenVR
46. Tarantool
47. wit.ai

暂缓

HOLD

48. CMS as a platform
49. Overambitious API 网关
50. Superficial private cloud

平台 接上页

能也不具优势。如果是高复杂性应用，我们还是推荐容器或虚拟机部署。此外，我们使用Java实现Lambda函数时遇到了严重问题，在Lambda容器启动时会不太稳定甚至延迟数秒。当然，JavaScript或Python没有这样的问题，但如果Lambbda函数代码量足够少，选择什么样的编程语言都不会有太大影响。

Realm是一款为移动设备设计的数据库，它拥有高性能的持久化引擎，定位是SQLite和Core Data的替代者。注意的是，Realm的数据迁移并没有它文档中所宣称的那么简单；但已经有越来越多的团队将Realm应用在移动产品的正式环境。

作为多年游戏开发领域的主流平台，**Unity**如今已成为虚拟现实（VR）和增强现实（AR）应用开发平台的首选。无论你正在Oculus或HTC Vive头戴设备中创造一个完全身临其境的世界，还是在新的空间受限的企业应用里使用全息技术，或为移动应用添加增强现实功能，Unity都可能已提供从原型设计到产品运营的一切资源。ThoughtWorks的大部分人认为，下一代计算平台的重大转变即在虚拟现实与增强现实，而Unity是这一波变化中最重要的一个开发工具。我们所有虚拟现实产品的原型、头戴式设备和手持式设备的增强现实功能，都是用Unity开发的。

.NET Core是一款开源模块化产品，用于创建可部署到Windows、MacOS及Linux的应用程序。这使得用户能够基于ASP.NET Core及系列工具、库和框架创建跨平台的Web应用——构建微服务架构的另一种选择。.NET Core社区正在壮大，像Visual Studio Code等新工具正在不断出现并且发展迅速。内置.NET Core的Linux和Windows （Nano Server）Docker 镜像简化了微服务开发。技术雷达曾介绍过CoreCLR和CoreFX，但几个月前微软已正式声明为.NET Core 1.0，这是该项目的第一个稳定版本。新机遇、新特性、社区活跃度，是我们好它的理由。

Amazon API网关帮助开发者在互联网上向客户发布API服务。它提供了流量管理、监控、认证和授权等常见的API网关特性。我们已有团队将它用于其他AWS服务的入口，如构建无服务器架构的AWS Lambda。我们仍在持续关注过度臃肿的API网关带来的挑战，但当前Amazon API网关服务看起来足够轻量级，避免了那些问题。

Apache Flink是新一代批处理及流处理的分布式可扩展平台。其核心是一个数据流引擎，支持表格（类SQL）、

图处理和机器学习。Apache Flink因其丰富的流处理特性脱颖而出：事件时间、丰富的流窗口操作、容错能力以及一次性（exactly-once）语义。这个项目在刚发布的1.1版本中加入了新的数据源集成和改进的流处理特性，表现十分活跃。

亚马逊最近发布了**AWS应用负载均衡器**（ALB），作为在2009推出的弹性负载均衡器（ELB）的直接替代品。ALB支持7层流量检查，为支持现代云端架构而设计。如果你正使用弹性计算云容器服务（ECS）构建微服务系统，这款新的负载均衡器能够直接处理容器托管和伸缩，即在每个虚拟机节点（EC2）的实例上具有多个容器和端口。基于内容的路由允许将请求按照目标服务器组分发，从而使这些组能够独立伸缩。负载均衡器自带的健康检查也得到了很大改进，能够捕获应用程序性能的详细指标。 我们喜欢这些功能，许多团队已经成功应用了ALB。

Apache的Cassandra数据库是一个功能强大，可扩展的大数据解决方案，用于存储和处理海量数据，它的集群经常被分散部署在全球多个位置的数百个节点里。我们喜欢这样功能强大的工具，但也发现经常有团队因为使用它而遇到麻烦，因此我们建议小心的使用Cassandra。团队经常误解Cassandra的用法，试图将其用作通用的数据存储平台，而事实上它是针对基于预定义键或索引的大型数据集的快速读取而优化的数据库，它对存储格式（schema）的依赖也使得很难随时间演进数据结构。Cassandra的操作复杂性很高且不方便，除非你绝对需要它的扩展性，其他简单方案更好。如果你并不满足Cassandra的特殊应用场景和扩展性，那么使用它也许只是在盲从大数据。小心使用Cassandra还应包括大量的自动化测试，我们很愿意推荐将CassandraUnit作为测试工具的一种选择。

Electron是使用HTML，CSS和JavaScript等Web技术构建本地桌面客户端的实用框架。 团队可以充分利用他们开发Web的能力来交付精致的跨平台桌面客户端，而无需花费时间学习另一系列技术。

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

平台 接上页

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。区块链技术中，**以太坊（Ethereum）**取得了良好的进展，值得关注。以太坊是公共区块链，允许使用其内置的编程语言创建“智能合同”。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蛮荒地带。

在**HoloLens**中，微软设计了第一个真正可用的AR头戴式设备。它不仅作为一份美丽的工业设计和一个极度舒适的穿戴设备，也清晰地表达了微软通过精湛的光学技术和深度集成Windows 10实现了对其AR产品的承诺。我们预计HoloLens将在短期内成为为客户提供实质性应用功能的第一个AR平台，并期待它的发展获得更广泛的成功。

IndiaStack是一组开放APIs，旨在将印度从过去的数据弱国转变为一个数据创新大国。该技术栈定义了一系列原子级APIs来促成分层式创新，并鼓励在APIs之上构建自定义应用程序以构建整个生态系统。Aadhaar作为基础层的服务之一，为十多亿印度公民提供了认证服务。此外，还有数字签名（eSign），统一在线支付（UPI）和电子合同层（e-KYC）的服务，以便向服务提供商安全地提供Aadhaar所存储的详细数据。我们相信Open API驱动将带来数字创新，而IndiaStack背后的设计原则可以作为其他国家和地区进行类似改变的范例。

Nuance Mix是Nuance公司推出的一个自然语言处理框架，该公司还设计了Dragon Speaking和Siri早期版本的语音识别技术。该框架支持语法生成，从而允许用户通过语音进行自由形式的交互。它的开发人员定义了一种领域特定语法能够让框架自主训练以进行理解，响应的结果会根据用户输入识别的用户意图和交互概念进行判断。起初，Mix仅能理解被训练过的短语的近似语句，但随着时间的推移，它已经开始从许多不同的短语中识别含意。虽然仍处于Beta阶段，但其早期实验版本的准确性已经引人注目，对于那些使用免提操作的应用（包括移动，物联网，AR，VR和互动空间）来说，它的最终产品非常值得关注。

OpenVR是为众多VR头戴式显示器（HMD）提供Unity支持的基础SDK，其重要性可能还将保持显著的增长。ThoughtWorks的大部分VR作品都是在OpenVR上构建的，因为相对其他SDK，它可以在任意HMD设备上运行。虽然没有开源，但OpenVR采用了免费使用许可证。Oculus SDK在其许可方面限制很多，且仅能在Oculus设备上运行。OSVR虽然开源但并没有多少使用者。如果你打算开发一个VR应用程序和并尽可能多的适配目标设备，并且不想直接使用Unity或Unreal进行开发，那么OpenVR是目前最具体实用的解决方案。

Tarantool是通过对象实体实现数据库和缓存的开源NoSQL解决方案，并提供Lua语言编写应用程序逻辑的APIs。内存和磁盘存储引擎双重支持，用户可以根据其使用场景创建多索引（HASH，TREE，RTREE，BITSET）。数据本身以MessagePack格式进行存储，并使用相同的协议在客户端和服务器之间进行通信。Tarantool支持预写日志、事务和异步主-主（master-master）复制。 我们对其采用的单一写入者策略和多任务协作处理并发连接的架构决策感到满意。

围绕人工智能领域的追捧正盛嚣尘上，但与大数据一样，实用的框架和工具仍待发掘。**wit.ai**便是其中之一，它提供了一个SaaS平台，允许开发者使用自然语言处理（NLP）创建会话接口。Wit支持文本或语音输入，帮助开发者管理会话意图，并支持JavaScript实现自定义业务逻辑。该系统可免费用于商业和非商业用途，并鼓励创建开放式应用程序。需注意的是，Wit会使用您的数据，以改进其服务，并用于内部分析，因此使用前请仔细阅读它的使用条款。该领域另一个竞争者是Microsoft Bot Framework，但当前它仅提供了有限的预览版本。我们预计Bot Framework与大多数微软产品一样会有迅猛的发展，因此值得关注。

我们看到太多组织为了交付大型和复杂的数据应用，试图将他们的**CMS作为平台**来使用，从而陷入困境。这常常是由供应商的“好意”驱动的，想要帮助业务部门绕过那些响应缓慢的IT组织，从而可以在生产环境上直接对业务通过拖拽的方式进行变更。当然我们非常支持给内容生产者提供正确的工具和工作流，但是对于拥有复杂业务逻辑的应用，我们还是倾向于把CMS只作为平台的一个组件（通常用hybrid或headless模

式），来清晰地与其服务进行合作，而不是试图在CMS上面实现所有的功能。

在中间件中实现了业务逻辑，是我们经常抱怨的一件事。这会导致那些本应负责传输功能的软件，却要野心勃勃地运行关键应用逻辑。API网关市场竞争十分激烈，那些供应商们持续地向他们的产品中添加新的功能，从而体现产品之间的差异化。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网关可以提供一些处理通用问题的功能，例如身份验证和速率限制。但诸如数据转换或规则处理这样的业务领域逻辑，应该位于应用或服务中，从而能被经常与这些业务领域打交道的产品团队所控制。

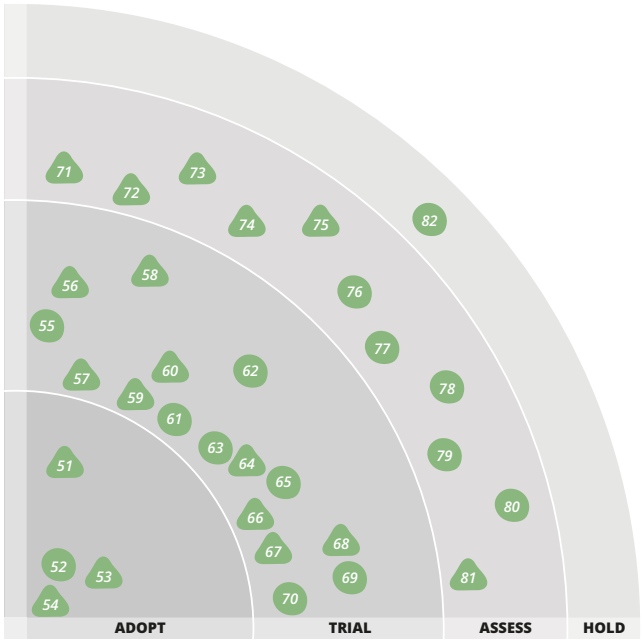
工具

Babel.js已经成为下一代JavaScript的默认解析器了，得益于它重新组织的插件系统，围绕它的生态圈已经初步形成。它允许开发者在浏览器或服务器上编写ES6（甚至ES7）时，只需少量配置，就可以保证旧版本浏览器的向后兼容。它提供了一流的构建和测试支持，能够轻松的和任意流程集成。已经成为ES6（和ES7）创新并被采用的主要驱动者。

结合流行的技术和架构风格，如微服务、DevOps、QA in production，开发团队需要越来越复杂的监视器。简单看一眼磁盘占用率和GPU利用率已经不够，很多团队会利用工具， 如Graphite和Kibana，收集应用和业务特定的指标数据。**Grafana**可以轻松地从不同数据源中创建有用且优雅的监控面板。它有一项很实用的特性，在不同图表中应用同一时间刻度，可以帮助发现数据隐藏的关联关系。模版系统的引入让我们看到了更多的期待，这可以更容易地管理相似的服务。Grafana已成为我们在监控领域的首选。

镜像已成为现代部署流水线的主要部分，也有大量的工具和技术用于创建镜像。基于一些特性优势和我们的大量经验，相比其它同类工具，我们推荐Packer，同时也不建议编写自定义脚本去做Packer已定义的任务。

针对测试金字塔塔尖的功能测试，越来越多的Android团队采用**Espresso**。它使用少量的核心API封装了大量复杂的内部实现，帮助实现简洁的测试代码，并快速和稳定地执行。依靠Espresso，你能将模拟用户行为的自动化测试运行在模拟器及真机的多个不同Android系统版本上。



fastlane是眼下常用的iOS和Android自动化构建工具，功能覆盖了移动应用从构建、测试、打包到发布整个流程。fastlane配置简单、功能完善，并且有多种工作流程可选，已是实现移动应用持续交付的一项利器。

对响应式页面在不同环境下的页面布局和样式的测试总是很耗时且通常采用手工验证。基于Selenium实现的**Galen**帮助解决了这个问题。它语法简单，支持对不同屏幕尺寸的条件验证。虽然Galen跟其它端到端测试工具一样有脆弱和速度慢的问

采用
ADOPT
51. Babel
52. Consul
53. Grafana
54. Packer

试验
TRIAL
55. Apache Kafka
56. Espresso
57. fastlane
58. Galen
59. HashiCorp Vault
60. JSONassert
61. Let's Encrypt
62. Load Impact
63. OWASP Dependency-Check
64. Pa11y
65. Serverspec
66. Talisman
67. Terraform
68. tmate
69. Webpack
70. Zipkin

评估
ASSESS
71. Android-x86
72. axios
73. Bottled Water
74. Clojure.spec
75. FBSnapshotTestcase
76. Grasp
77. LambdaCD
78. Pinpoint
79. Pitest
80. Repsheet
81. Scikit-learn

暂缓
HOLD
82. Jenkins as a deployment pipeline

工具 接上页

题，但它在快速解决反馈的设计问题。

项目中的密码管理已成为非常重要的事情。以前是将这些密码放在一个文件或者环境变量中，但这种方式越来越难于管理，特别是在多个应用和多个微服务共享环境的情况下。**HashiCorp Vault**对这个问题的解决机制是为安全访问密码提供统一的接口。它已经在我们多个项目中很好地解决了这个问题，Vault同HashiCorp服务集成的简单性也深受团队的欢迎。存储和更新密码有些繁琐，因为它依赖命令行和团队的众多纪律。

越来越多的项目将信息传递格式定义为JSON。在Java中为JSON写测试是很耗费时间的。**JSONassert**是一个很小的库，减少了处理JSON断言的代码，并提供了更友好的错误提示。

Pa11y是一个自动化辅助功能测试工具，它通过命令行运行，可以与构建流水线集成。通过创建静态版本页面，然后运行辅助功能测试用例进行验证，我们的团队已经成功的在动态网站上使用Pa11y。对于很多系统——特别是政府网站——包含辅助功能测试这一需求，Pa11y可以让这项测试简单很多。

将密码存放在代码库中是重要系统的一块软肋，在有了成熟的工具Vault之后，我们没有理由再这样做。我们之前已经提到过仓库扫描工具，如Gitrob，但我们现在要推荐更完善的工具，如（ThoughtWorks出品的）**Talisman**，它是一个Git回调函数，在代码提交前根据预定义规则检查是否包含密码。

在**Terraform**的助力下，你可以仅仅通过一些声明式的定义，来管理云基础设施。通过Terrafrom生成基础设施配置后，通常会用Puppet、Chef或者Ansible这种配置部署工具去实施构建。我们很认可Terraform，因为其语法简单、可读性强，并且无缝支持多种云服务提供商。和两年前我们首次谨慎提及Terraform相比，它已持续发展成为可靠的产品，并且在我们的项目中展现了价值。关于状态文件管理的问题也可用“后台远程状态”来解决，我们已成功地在Consul上实施这一特性。

结对编程是一种重要的编程方式，如今越来越多的团队在进行异地分布式开发，因此我们尝试过多种远程结对编程的工具。我们非常认可ScreenHero，但是对其发展前景较为担忧。对那些不使用图形化IDE的分布式团队，使用**tmate**进行结对无疑是个很棒的方式。tmate构建在tmux基础之上，较tmux而言，tmate安装简单很多。与图形化屏幕共享工具相比，tmate对于带宽等资源的需求较低，使用过程中不会被卡顿模糊的显示所困扰。团队可以自建服务器，以保证使用中的隐私安全和功能完整性。

Android-x86是开源Android在x86平台下的实现。这个项目最初是集成了社区中多个支持x86的补丁，后来创建了自己的代码库来支持不同的x86平台。在我们的CI服务器上使用Android-x86替换模拟器进行独立的UI测试后显著的缩短了时间。在针对某些分辨率运行UI测试时，它在内存消耗，带宽使用和电量消耗上都优于模拟器。

我们的团队已经成功使用了**axios**，一个基于promises的JavaScript HTTP客户端，并且认为比Fetch更好。axios在GitHub上非常活跃且广受好评，我们对其也非常认可。

随着人们对“流(streaming)数据架构”和其下游“数据湖(data lakes)”的兴趣的增长，我们已经看到他们越来越依赖于那些“改变数据捕获”的工具，来将事务性的数据存储连接到流处理系统中。在这个领域中，**瓶装水**的概念受到欢迎。例如它可以将**PostgreSQL**数据库的write-ahead日志中所发生的变化，转换成Kafka的事件。然而，这种方法的一个缺点是你会被绑定到低层次的数据库事件上，而不是高层次的业务事件上。而后者正是我们所建议使用的，因为它是面向事件架构的基础。

开发者论战中的一个永久话题，是有关编程语言的类型定义的：到底有多少种类型才是恰当的？而Clojure这种在JVM上实现的动态类型的函数式Lisp语言，其令类型的边界愈加模糊的特点，又给论战添加了新的话题。**Clojure.spec**是Clojure内置的一个新特性，它允许开发人员将数据结构用类型和其他验证条件（例如允许的取值范围）进行封装。这种数据结构一旦建立，Clojure就能利用这种规格来为程序员提供大量的便利：自动生成的测试代码、合法性验证、析构数据结构等等。Clojure.spec提供方法很有前景，它可以让开发者在需要的时候，就能从类型和取值范围中获益。

iOS应用程序的用户界面视觉效果测试是一件充满痛苦的事情，它进展缓慢，而且容易出错。所以这次我们很高兴地将**FBSnapshotTestcase**纳入到我们的工具箱

工具 接上页

中。FBSnapshotTestCase能够自动化地获取用户界面组件的快照图片，将其存储并进行对比，从而可以让用户界面的完美程度精确到像素级别。因为它（在模拟器内）以单元测试的方式运行，所以相比功能测试的方法，其运行速度更快，且更可靠。

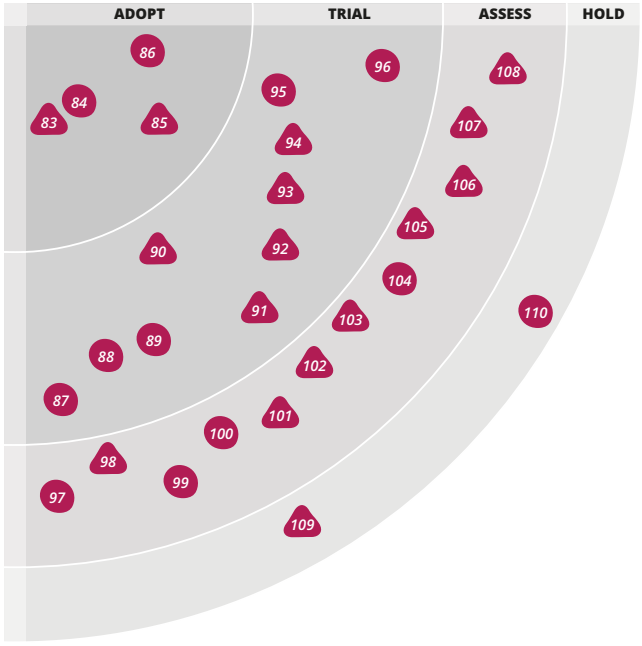
Scikit-learn是一款日益流行的用Python语言编写的机器学习库软件。它提供了一组功能强大的机器学习模型，如聚类、分类、回归和降维，以及类似模型选择、模型评估和数据准备等丰富的相关功能。由于它设计得简单易用，在不同的上下文中能够复用，且具备良好的文档，我们认为即使是非专业人员，也能借用这款工具来探索机器学习的领域。

语言和框架

如果你正尝试为单页面应用程序（SPA）选择一个框架，**Ember.js**是上上之选。Ember因其高效的开发者体验和远低于其它诸如AngularJS等框架带来的“意外惊喜”受到我们团队的推崇。在Javascript构建工具的疾风暴雨中，Ember CLI就像是一个天堂。而且，Ember的核心团队和社区十分活跃、响应十分迅捷。

随着JavaScript单页面应用程序越来越复杂，我们已经看到了更为迫切的，使客户端状态管理可预测的需求。在ThoughtWorks交付的一些项目中，**Redux**及其对更新状态的三个限制原则证明了自己在这个领域的价值。无论你是新手还是经验丰富的开发者，都可以通过Getting Started with Redux和idiomatic Redux教程入门。 Redux本身体小精悍的设计衍生出丰富的工具集，使用Redux前，我们推荐您查看redux-ecosystem-links项目中的示例，中间件和实用工具库。此外，我们还特别推崇该架构的可测试性：分派动作，状态转换和界面渲染都可以以最少的mocking实现彼此独立地做单元测试。

开发者对于**Elixir**这门编程语言的兴趣正持续升温。我们越来越多地看到它被运用在正式的项目中，并且听到来自开发者的下述反馈，即其Actor模型既健壮又快速。Elixir是一款建立在Erlang虚拟机之上的动态编程语言，它在构建高并发和高容错的系统方面，正展现出不错的前景。Elixir具有很多与众不同的特性：例如管道操作符，它允许开发人员构建一个类似于UNIX shell命令的函数管道。Elixir通过共享字节代码与Erlang互连，在利用现有库的同时，也支持诸如Mix构建工具、IEx互动shell和**ExUnit**单元测试框架这样的工具。



我们非常享受Enzyme为React.js应用提供的快速组件级UI测试功能。与许多其他基于快照的测试框架不同，Enzyme允许开发者在不进行设备渲染的情况下做测试，从而实现速度更快，粒度更小的测试。在开发React应用时，我们经常需要做大量的功能测试，而Enzyme可以在大规模地减少功能测试数量上做出贡献。

函数式编程范式经常强调不可变性，并且大多数的编程语言能够创建不可变对象——对象一旦被创建就不能修改。**Immutable.js**是一个的JavaScript工具库，来提供很多

采用
ADOPT
83. Ember.js
84. React.js
85. Redux
86. Spring Boot

试验
TRIAL
87. Butterknife
88. Dagger
89. Dapper
90. Elixir
91. Enzyme
92. Immutable.js
93. Phoenix
94. Quick and Nimble
95. React Native
96. Robolectric

评估
ASSESS
97. Aurelia
98. ECMAScript 2017
99. Elm
100. GraphQL
101. JuMP
102. Physical Web
103. Rapidoid
104. Recharts
105. ReSwift
106. Three.js
107. Vue.js
108. WebRTC

暂缓
HOLD
109. AngularJS
110. JSPatch

语言和框架 接上页

持久不可变的数据结构，令其在现代JavaScript虚拟机中可以非常高效的运行。然而，Immutable.js对象并不是正常的JavaScript对象，所以需要避免在immutable对象中直接引用JavaScript对象。现在，越来越多的团队用它来追踪变化，并维护线上应用的状态。我们建议开发者研究此类库，尤其当它和其他Facebook技术栈一起使用时。

我们ThoughtWorks的一些团队在使用**Phoenix**时都获得了一种良好的体验。Phoenix是一个用Elixir写就的服务端web MVC框架，除了简单易用以外，它还借助Elixir来令其变得及其快速。对于某些开发者来说，Phoniex唤起了他们第一次接触Ruby和Rails时所经历的欣喜之情。尽管对于某些更成熟的框架而言，Phoniex的类库生态还不够丰富，不过它应该能从Elixir的持续进步和支持增长当中获益。

目前，ThoughtWorks的大多数iOS开发团队都在使用**Quick**和**Nimble**组合做单元测试，作为RSpec家族的behavior-driven development (BDD)测试工具，它们通过describe代码块为Swift和 Objective-C提供颇具可读性的测试。此外，它们还为异步测试提供了很好的支持。

ECMAScript 2017——注意不要和ES7（又名 ECMAScript 2016）混淆——为JavaScript这门语言带来了几个值得注意的改进。到2017年夏天，各个浏览器有望全部实现这一标准。不过JavaScript编译器Babel现如今已经支持了该标准的大量特性。对于广泛使用JavaScript且代码库正处于活跃的开发阶段的团队，我们建议把Babel添加到构建流水线中，来开始使用这些已被支持的特性。

JuMP是使用Julia编程语言来进行数学优化的领域特定语言。JuMP定义了一个称为MathProgBase的通用API，允许用户在Julia中编写求解器(solver)无关的代码。 目前支持的求解器包括Artelys Knitro，Bonmin，Cbc，Clp，Couenne，CPLEX，ECOS，FICO Xpress，GLPK，Gurobi，Ipopt，MOSEK，NLopt和SCS。除此之外，另一个好处是在反向模式中实现自动微分技术来计算导数，因此用户不限于使用像sin，cos，log和sqrt这样的标准运算符，也可以在Julia中实现自己的自定义目标函数。

我们对Google提出的**Physical Web**标准颇感兴趣。Physical Web的创意非常简单——使用信号灯(beacon)来广播URL——但却带来了无限可能。 基本上，这是一种标注(annotate)物理世界的方法，来把现实中的物体和位置绑定到数字领域。 它目前的传输机制是基于低功耗蓝牙的Eddystone-URLs，一些演示用的客户端可供使用。虽然访问随机发现的链接有明显的安全问题，但我们最感兴趣的用例，是可以按照需要在定制客户端中对URL进行过滤或代理。

Rapidoid是一组web框架模块的集合，包括了一个基于Java NIO来全新实现的快速底层HTTP服务器。Rapidoid巧妙地使用了堆外输入输出缓冲区、对象池和线程局部数据结构，令其比像Netty那样基于NIO的其他服务器更胜一筹。作为一个非常年轻的项目，Rapidoid尚未实现如内置缓存和SSL支持等一些特性。我们建议关注该项目的路线图来获取最新信息。

我们非常兴奋的看到，Redux范式以**ReSwift**的形式迈向了Swift领域。我们发现， 一旦应用状态及其改变被集中管理且标以通用名称，就会对代码的简单性和可读性带来很多真切的好处。另外，这也有助于构建“离线优先”的应用程序。

尽管各种新型头戴设备的泛滥表明市场的狂热，但我们相信在浏览器中，特别是在移动平台上，会存在许多合理的VR和AR的应用场景。在这种趋势下，我们发现强大的JavaScript可视化与渲染框架**Three.js**被使用得越来越多。它基于WebGL。而目前对于WebGL支持的增长，又促使更多人采用Three.js，并形成了一个充满活力的社区来支持这个开源项目。

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不很死板的库。它围绕着如模块化、组件和反应式数据流的概念，为构建交互式Web界面提供了一套工具。它所拥有的较低的学习门槛，会令初级开发者和新手感兴趣。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已有项目集成。

随着将AR/VR作为协作和沟通的媒介得到广泛采用，市场需要一个现代且易于获得的视频流平台。**WebRTC**是一个浏览器之间进行实时通信的新兴标准，它使用常用的Web技术来支持视频流。支持该标准的浏览器正不断增加，但微软和苹果对于

在它们专有的浏览器上采用WebRTC行动缓慢。如果保持这样的发展势头，WebRTC将会形成未来在Web上进行AR/VR协作的基础。

AngularJS为单页JavaScript应用世界带来过革命性的变化，而且过去数年内我们使用它成功地交付了许多项目，然而，我们不再推荐团队在启动全新的项目时使用Angular 1，除非团队已经有了经验或者已经对其进行了投入。我们更青睐使用Ember和React（尤其是结合Redux使用）所带来的开发速度提升和更加可维护的代码。

ThoughtWorks是一家全球软件设计与定制领袖企业，1993年成立于美国芝加哥。2016年，ThoughtWorks在十四个国家：澳大利亚、巴西、中国、智利、厄瓜多尔、德国、印度、意大利、新加坡、西班牙、南非、土耳其、英国和美国成立分公司，截止目前，她已经成为超过4000位行业精英的实践乐园。ThoughtWorks进入中国十年，分别在北京，上海，深圳，西安，成都和武汉成立了分公司，有接近1000名员工通过全球资源共享和人才交流，服务全球各行业的顶尖客户。在这里，我们与全球顶尖企业合作，用颠覆性思维与客户一起并肩推动商业变革。这里，我们与世界共同进步，我们是一个全球化的社区，崇尚人道主义、多元文化和团队协作。在这里，我们将创意付诸实践，我们把对技术的热情和创意变为现实，以引导软件创新、设计和交付的革命为己任，助推全球社会变革。

ThoughtWorks®