ThoughtWorks®

TECHNOLOGY RADAR *NOV '15*

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

thoughtworks.com/radar

最新动态

本版精彩集锦

DOCKER引爆容器生态系统

以Docker为典范的容器技术在越来越多的组织中广受欢迎。在组织间以及组织内部对容器技术的兴趣点差异很大，我们所推荐范围也相应的从**评估**到**采用**。容器技术的生态系统（工具、平台及技术）正在逐步成长和成熟，更进一步加速了对该技术的关注度。敏锐的读者会注意到在我们的技术雷达上有多篇相关的主题，范围从把Docker作为开发工具来管理众多依赖，到大型的云平台把容器作为基本的“伸缩单元”，比如Mesos和AWS ECS。

微服务及相关工具受到追捧

业界对微服务这种架构风格的兴趣一直持续并未减弱，这同时也增强了对其支撑工具和技术的关注：比如像容器化技术这样的DevOps实践，比如从尝试在CI/CD工具中编程的学习到的惨痛教训，比如服务发现工具的成熟，等等。我们期望在不远的将来在这一领域看到越来越多技术及工具的成长和成熟。

JAVASCRIPT工具正在趋于平稳

我们在过往技术雷达期刊中强调过Javascript相关工具的急速增长，而现在整个社区正逐步从这种状态中平静下来并总结出一些通用的实践。团队正在探索构建工具和包管理之间的最佳组合（包括没有），同时在团队之间对有效实践的认同也基本上达成了一致。

安全是每一个人的问题

在软件开发生命周期中，安全是一个独一无二的会影响到所有角色的问题。我们在上一期雷达中着重指出了安全领域的一些改进，很高兴在很多团队中已经开始把安全实践结合在软件开发的生命周期中。这一期我们同样突出了这方面的创新，比方说 bug bounties，安全建模，HSTS，TOTP和Let's Encript。我们希望这种提升的势头能够继续。

贡献者

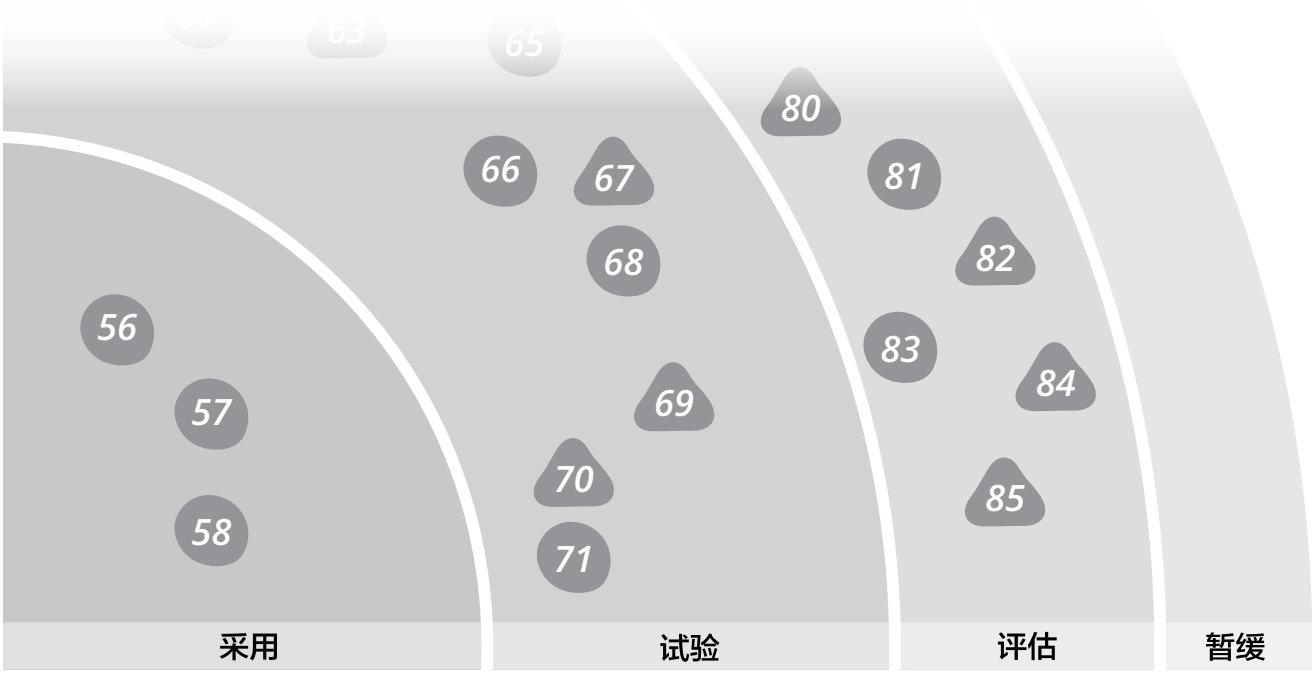
ThoughtWorks 技术顾问委员会由以下人员组成：

Rebecca Parsons (首席技术官)
Martin Fowler(首席科学家)
Anne J Simmons
Badri Janakiraman
Brain Leke
Claudia Melo
Dave Elliman
Erik Doernenburg
Evan Bottcher
徐昊
Ian Cartwright
James Lewis
Jonny LeRoy
Mike Mason
Neal Ford
Rachel Laycock
Sam Newman
Scott Shaw
Srihari Srinivasan
Thiyagu Palanisamy

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、描写，并持之以恒地推动技术的发展——以求造福大众。我们的使命是追求卓越软件并掀起 IT 革命。我们创建并分享 ThoughtWorks 技术雷达，正是为了达成这一使命。ThoughtWorks技术顾问委员会由ThoughtWorks 一群资深的技术领导者组成，他们定期召集会议讨论 ThoughtWorks 的全球技术战略，分析对行业产生重大影响的技术趋势，从而创建了技术雷达。

雷达以独特的形式记录技术顾问委员会的讨论结果，从CIO到开发人员， 雷达为各方利益相关者提供价值。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。雷达的本质是采用图形化方式将各种技术归类为技术、工具、平台和语言及框架四个象限。倘若雷达中的某种技术可以被归到多个象限中，我们会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中，由此反映我们目前对它们持有的态度。这四个环分别为：



我们强烈主张业界采用这些技术。如果适合我们的项目，我们会采用它们。

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该项技术。

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. Decoupling deployment from release new
3. Generated infrastructure diagrams
4. NoPSD
5. Products over projects
6. Threat Modelling

试验
7. BEM new
8. BFF - Backend for frontends new
9. Docker for builds new
10. Event Storming new
11. Flux
12. Idempotency filter new
13. iFrames for sandboxing new
14. NPM for all the things new
15. Offline first web applications
16. Phoenix Environments
17. QA in production new

评估
18. Accumulate-only data
19. Bug bounties new
20. Data Lake
21. Hosted IDE's new
22. Monitoring of invariants new
23. Reactive Architectures

暂缓
24. Gitflow new
25. High performance envy/web scale envy new
26. Microservice envy
27. Pace-layered Application Strategy
28. Programming in your CI/CD tool
29. SAFe™
30. Separate DevOps team

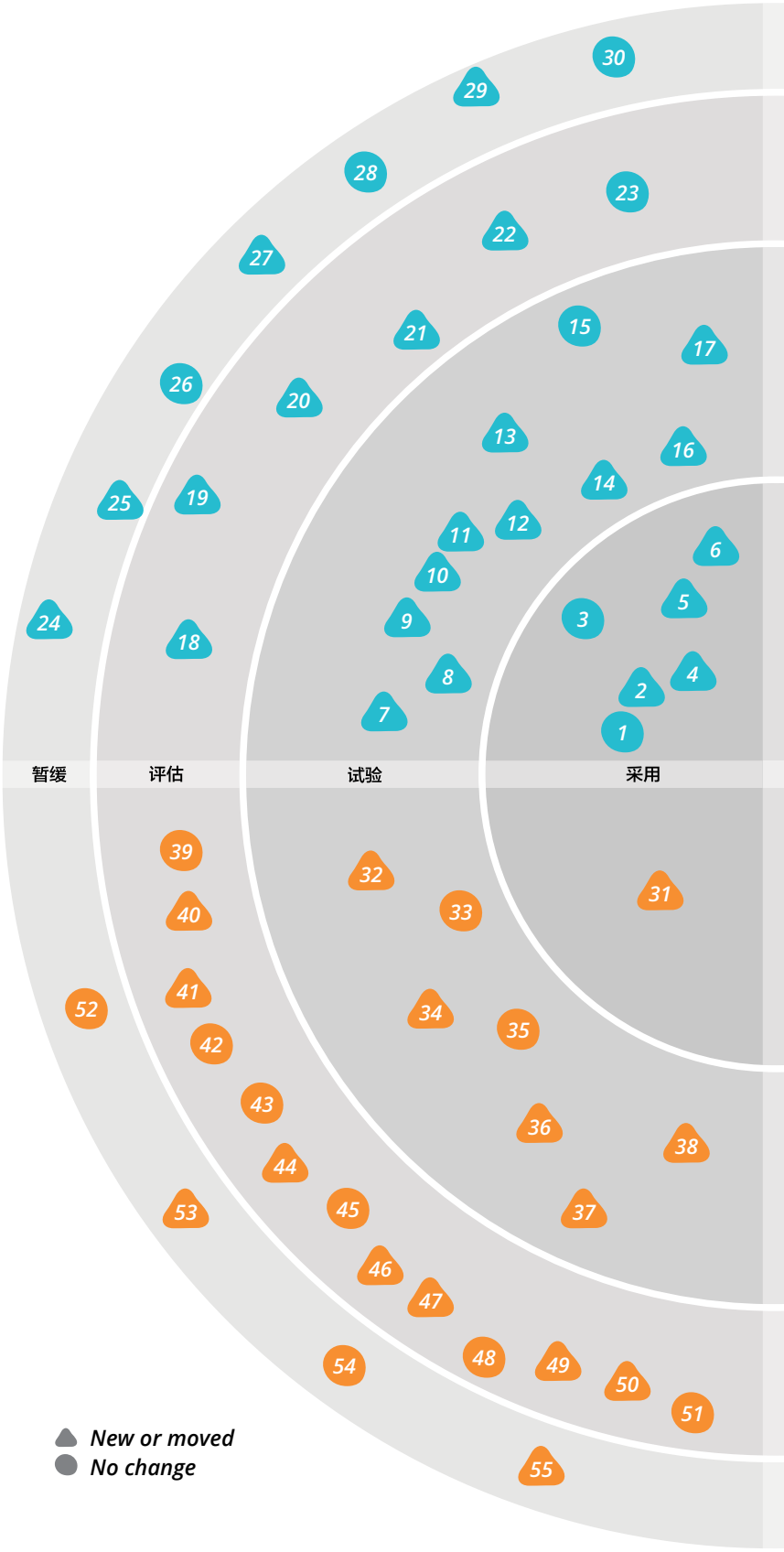
平台

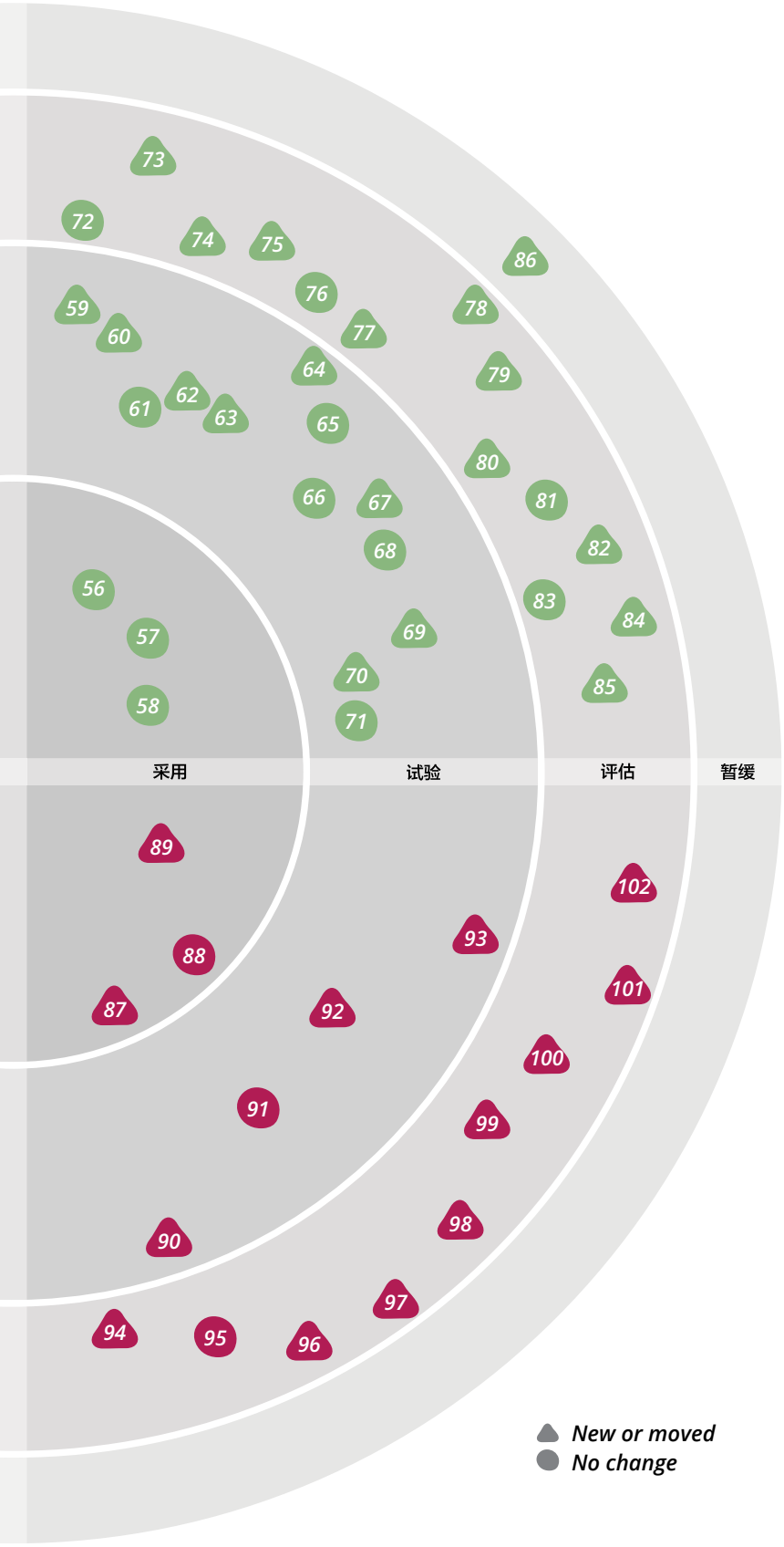
采用
31. TOTP Two-Factor Authentication

试验
32. Apache Mesos
33. Apache Spark
34. AWS Lambda new
35. Cloudera Impala
36. Fastly new
37. H2O
38. HSTS new

评估
39. Apache Kylin
40. AWS ECS new
41. Ceph new
42. CoreCLR and CoreFX
43. Deis
44. Kubernetes new
45. Linux security modules
46. Mesosphere DCOS new
47. Microsoft Nano Server new
48. Particle Photon/Particle Electron
49. Presto new
50. Rancher new
51. Time series databases

暂缓
52. Application Servers
53. Over-ambitious API Gateways new
54. SPDY
55. Superficial private cloud new





THE RADAR

工具

采用

56. Composer
57. Mountebank
58. Postman

试验

59. Browsersync new
60. Carthage new
61. Consul
62. Docker Toolbox new
63. Gitrob new
64. GitUp new
65. Hamms
66. IndexedDB
67. Polly
68. REST-assured
69. Sensu
70. SysDig new
71. ZAP

评估

72. Apache Kafka
73. Concourse CI new
74. Espresso new
75. Gauge new
76. Gor
77. ievms new
78. Let's Encrypt new
79. Pageify new
80. Prometheus
81. Quick
82. RAML new
83. Security Monkey
84. Sleepy Puppy new
85. Visual Studio Code new

暂缓

86. Citrix for development

语言和框架

采用

87. ECMAScript 6 new
88. Nancy
89. Swift

试验

90. Enlive new
91. React.js
92. SignalR new
93. Spring Boot

评估

94. Axon new
95. Ember.js
96. Frege new
97. HyperResource new
98. Material UI new
99. OkHttp new
100. React Native new
101. TLA+ new
102. Traveling Ruby new

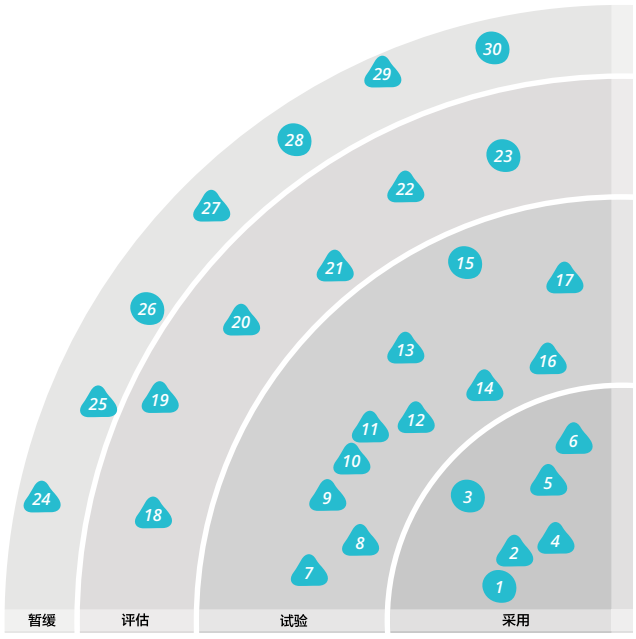
暂缓

技术

实现持续交付对很多组织而言依然很有挑战，强调例如**解耦部署和发布**等实用技巧的使用尤为重要。我们推荐严格区分术语“部署”和“发布”的使用。应用组件或基本设施的代码或配置变更在产品环境生效称为“部署”，而具有业务影响的功能变化对最终用户可见称为“发布”。使用特性开关或灰度发布等技巧可以使我们更加频繁地部署变更到产品环境但并不发布功能。频繁部署可以有效降低变更带来的风险，同时业务负责人仍然能保持何时向最终用户发布功能的控制。

NoPSD运动想要抓住的，是“即时设计”这个在视觉设计中重要和实用的概念。你不需要预先设计好整个应用程序或者每一个界面元素，而是使用尽可能轻量的工具按需设计。我们已经看到简单易学的工具被更多的使用，比方说**Sketch**，纸和笔也同样有更多回归的趋势（尤其是与一个现有强大的数字风格指南配合使用的时候）。由于为屏幕设计时平面模型的限制，使用类似于**Invision**、**FramerJS**和**Origam**这样的工具，或者是简单的HTML/CSS加上一点Javascript来创造不同的保真效果的原型，也已经成为沟通设计意图时逐渐常见的很有用的方式。

我们长期坚信将软件开发看做一个项目（即在有限的时间和预算内交付一些东西）的观点并不适合现代的商业模式了。软件开发的主要努力应该用在使软件成为一个支撑和反思业务并持续演进的产品。在其所支撑的业务失效以前，这样的努力都不应停止。根据我们在自己以及外部项目中的观察，我们确信这种**“产品优于项目”**的方法对除特殊情况外的所有情况都有效。



鉴于这几个月以来备受瞩目的安全漏洞频发，软件开发团队已经不需要被说服他们必须重点关注编写安全的软件，并且要以更负责任的方式对待用户数据。然而他们面临着陡峭的学习曲线，从有组织的犯罪和政府监控、到青少年“为了好玩”而攻击系统，这些大量的潜在威胁让人难以招架。**威胁建模**提供了一系列的技术，主要从防御的角度出发，帮助对潜在的威胁进行理解和分类。通过转化为“恶意用户故事”，威胁模型给予团队一种可以管理的、有效的方式来打造更加安全的软件系统。

调试CSS问题是非常痛苦的，有多少次你不得不从成千上万的样式覆盖中找出问题所在。为减少这种痛楚，很多团队引入了各种各样的代码规范(guidelines)，比如避免级联与覆盖，使用真正需要的(opt-in)样式，并强调有意义的命名。**BEM**(代表Block, Element,Modifier)是一种简明的CSS命名规范，它能把你的CSS变得更语义化和结构化。通过使用BEM，可以更容易的理解CSS规则是怎么影响一个元素的，并且更重要的是理解CSS规

采用
1. Consumer-driven contract testing
2. Decoupling deployment from release
3. Generated infrastructure diagrams
4. NoPSD
5. Products over projects
6. Threat Modelling

试验
7. BEM
8. BFF - Backend for frontends
9. Docker for builds
10. Event Storming
11. Flux
12. Idempotency filter
13. iFrames for sandboxing
14. NPM for all the things
15. Offline first web applications
16. Phoenix Environments
17. QA in production

评估
18. Accumulate-only data
19. Bug bounties
20. Data Lake
21. Hosted IDE's
22. Monitoring of invariants
23. Reactive Architectures

暂缓
24. Gitflow
25. High performance envy/web scale envy
26. Microservice envy
27. Pace-layered Application Strategy
28. Programming in your CI/CD tool
29. SAFe™
30. Separate DevOps team

技术 接上页

则本身。这个方法可以被看做是将面向对象中”组合优于继承”的经验搬到了CSS的世界中一样。

有价值的服务支持多种客户端，比如移动设备，网站以及各种形式的网络终端。能够设计一个可复用并且支持各个客户端的API是具有很大的诱惑力的。但是客户端需求总是不同的，比如移动设备对带宽的限制，而在网络速度快的情况下又对更多的数据有需求。因此通常情况下，**为不同的客户端设计不同的后台服务**是最好的方式。这些后台服务应该由与前台服务一致的团队开发从而保证后台服务能够满足其前端需求。

管理构建时依赖是**Docker**在我们的项目上诸多创新型用法之一。在过去，通常的做法是在操作系统上运行具有构建目标所需依赖的构建代理。Docker使得在一个具有所需依赖并完全隔离的环境里运行编译而不污染构建代理。这种使用**Docker来进行构建**的技术的有效性已经在Golang的编译场景中被部分证明，而golang-builder容器的存在正是为了此目的。

事件风暴是快速领域建模的一种方式，它由外及内，以领域中的关键事件而不是传统的静态数据模型作为起点。它以工作坊的形式展开，主要目的是识别领域关键事件，将它们以时间轴的方式组织，确定它们的触发条件，并探索它们之间的关系。这种方式对于采用CQRS或者Event Sourcing的开发者来说尤为有效。它成功的第一要素是找到合适的人。这组人需要融合业务和技术人员，能够提出问题并一起发现答案。第二，要确认你有足够的白板空间来建模。它可以通过可视化的方式，帮助我们一起发现全景，共同了解目标领域以及它的复杂度，确保在讨论解决方案之前对于问题的一致理解。

Flux是Facebook 推出的应用程序架构，通常与**React.js**被同时提及。它基于一个通过渲染流水线向上的单项数据流。Flux 拥抱基于客户端 JavaScript 的现代网络应用风格，避免了古老陈旧的 MV* 架构。ThoughtWorks 团队开始在这方面获得了一些经验，我们发现它与面向服务结合的很好，并且解决了双向数据绑定所带来的一些问题。

现在有很多服务，尤其是传统的服务，在编写时都假设请求仅会发生一次。做为网络本身而言，这很难保证。**幂等过滤器**是个简单的组件，他通过检查重复请求并保证发给服务器的请求只有一次。过滤器应当做为装饰器叠加在现有的服务调用上，并且它应该仅做这一件事情。

现代的网页往往含有过多的来自第三方的JavaScript部件和代码片段，这对安全和性能都会产生负面影响。虽然我们仍在等待基于Web组件的JavaScript完全隔离方案，我们的团队已经从使用HTML5 **iFrame来沙盒化**不可信的JavaScript中受益匪浅。

JavaScript的世界中有太多的依赖和包管理工具了，它们全部都基于NPM(Node Package Manager)之上。许多团队慢慢开始发现这些额外的工具有点冗余，他们推荐在条件允许的前提下只用NPM来管理包和依赖。只使用**NPM的简化方案**有利于减少JavaScript工具世界的噪音污染。

在很多软件开发项目中，准备和更新环境所花费的时间依然是重大的瓶颈。通过扩展凤凰服务这个概念到整个环境上，凤凰环境可以有效的降低这种延迟。我们觉得这是一种有很价值的节省时间的技术，你也应该考虑采用这种方式。通过自动化，我们可以创建包括网络设置、负载均衡和防火墙端口在内的整个环境（比方说使用 AWS 上的 **CloudFormation**）。之后，可以通过定期销毁、重建这个环境来验证这个过程是工作的。**凤凰环境**能够支持测试、开发、UAT和灾难恢复所需的新环境准备。仅仅使用凤凰服务器不总是可行，并且还需要仔细的考虑状态和依赖这些因素。当需要对环境进行重新配置时，把整个环境当做一次蓝绿部署也是一种可行的方式。

传统方式下，QA的角色主要专注于保证软件产品在类产品环境下的质量。随着持续交付的出现，QA的角色逐渐转变到需要分析软件产品在产品环境下的质量。这需要引入产品系统的监控，制定检测紧急错误的警报条件，持续质量问题的确定以及找出在产品环境下使用的度量以保证这种方式可行。这有可能会导致有些组织走的太远而忽视产品上线前的质量保证。我们的经验表明，**产品环境下的QA**只对那些已经执行并有一定程度持续交付实践的组织有价值。

不可变的数据结构正在变得越来越受欢迎，函数式语言，如 Clojure 和 Scala 默认就提供不变性。不变性使得代码更容易编写、阅读和理解。在数据库层，采用**“只积累”的数据存储**也给予了一些同样的好处，并且使审计和历史查询更为简单。实现

技术 接上页

选择各不相同，从使用 Datomic 这样特定的积累数据存储，到在传统数据库中采用简单的**“只追加，不更新”**的手段。“只积累”是一种设计策略，数据删除通过撤回而不是更新，而**“只增加”**只是一种具体的实现技术。

越来越多的组织开始通过**bug bounties** 鼓励记录常见的安全相关的bugs，帮助提高软件质量。为了支持这些方案，诸如HackerOne和BugCrowd公司的出现可以更容易地帮助企业管理这个过程。我们在这方面经验有限，但是我们喜欢这种鼓励人们以很开放，很透明的形式来提出常见的漏洞的想法。值得一提的是，在鼓励用户发现你们软件中的漏洞的前要先确认清楚是否会引发法律问题。

作为数据分析的来源，数据湖泊是一种不可变的数据存储，其中存有大量未加工的“原始”数据。而数据仓库在存储之前，先要过滤、处理数据，不同的是，湖泊仅捕获原始数据，把剩下的留给用户根据他们所需进行特定的分析。这样的例子包括 HDFS，或者基于 Hadoop，Spark 或 Storm 处理框架之内的 HBase。通常只有一小部分数据科学家针对原始数据进行工作，他们制定处理过的数据流入数据集市，为大多数用户提供查询。 数据湖泊应该只用于分析和报表。对于业务系统之间的协作，我们更倾向于使用专门为其设计的服务。

很多组织希望能充分利用分布式或海外交付，但对于其控制外的代码和知识产权的安全性深表担忧。 其结果往往是采用能维持安全控制，但牺牲开发人员生产力的高延迟的远程桌面进行开发的解决方案。另一种方案是使用通过VPN接入的浏览器访问**托管IDE**。IDE，代码和构建环境托管在组织内的私有云上既消除了安全忧虑，又大幅改善了开发者的体验。来自Eclipse基金会的Orion和Che，以及Cloud9和Code Envy都是这个领域的佼佼者。

在监控中，常见的方法是设计错误场景，设置相应告警。但是，这种方式往往难以穷举软件系统中的无数失败模式。**不变量监控**是对预期正常范围监控的补充，一般通过检查系统历史行为，并在系统行为超出正常范围时发出告警。

我们坚定地相信长期存在于版本控制中的分支会破坏像持续集成这种有价值的工程实践，这也正是我们不喜欢**Gitflow**的原因。我们喜欢Git带来的灵活性，却又痛恨Gitflow引入不好的工程实践。生命周期越短的分支集成时越容易。我们发现大多数使用Gitflow的团队受其滥用分支的工作流影响，推迟了集成点，最终阻碍了真正的持续集成。

我们看到很多团队因为觉得“系统可能会需要扩展”而选择复杂的工具、框架和架构遭遇问题。公司如Twitter和Netflix之所以需要这样的架构是因为他们需要支持极端负载的情况，并且他们拥有非常熟练的开发团队能够处理所带来的复杂性。在大多数情况下，这样的工程技艺是不必要的，团队应该克制他们的**互联网级规模的情结**而倾向于简单可行的解决方案。

Gartner **提出的节奏分层的应用策略**似乎在软件架构的层次方面创造出了一个无益的聚焦点。我们发现考虑业务能力（可能由几个架构层次构成）的不同变化节奏是一个更有用的概念。仅仅聚焦于层次的危险之处在于，很多类型的变化是在多个层次上纵切的。比方说，在网站上添加新的股票类型，不仅仅是要有一个易于变化的 CMS，你还需要更新数据库、集成点、仓库系统等等。识别出一个架构中的某些部分比起其他部分需要更强的机动性是很有用的。然而，聚焦于层次被证明是没什么用的。

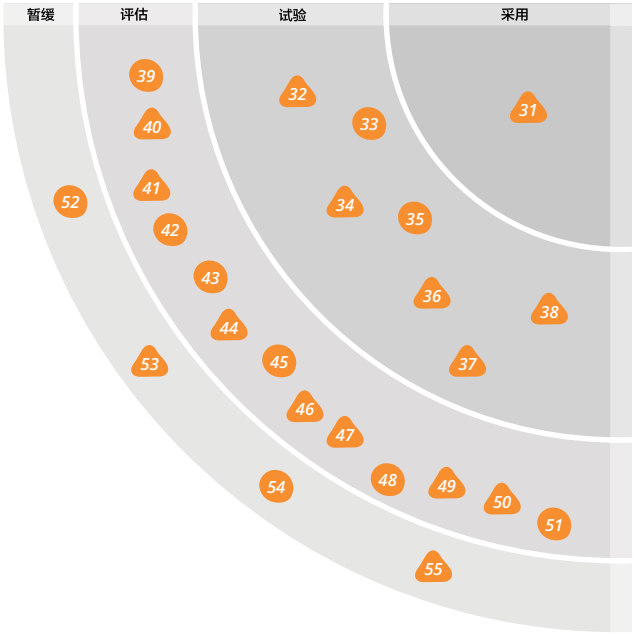
规模化敏捷框架 **Scaled Agile Framework®**（又名 **SAFe™**）继续获得很多大型组织的心理份额。此外，工具和认证正在成为是否采用了SAFe™的一个重要的判断依据。我们依然担心的是，在这个过程中容易过度标准化，并且趋向于大型发布，结果在实际上阻碍了敏捷的实行。在这方面，我们继续推荐精益方法，包括实验和持续改进实践的整合（如改进道场Improvement Katas），从而为企业提供更好的规模化敏捷模型。

Scaled Agile Framework® and **SAFe™** are trademarks of Scaled Agile, Inc.

平台

英国政府提倡使用技术控制让用户记住简单的密码，爱德华・斯诺登的密码建议被称作只达到了“及格线的安全”，密码的安全性仍然是一个激烈争论的话题。密码在安全链中是最薄弱的环节之一，所以我们建议采用可显著提高安全性的**双因素认证**。基于时间的一次性密码（**TOTP**）是这个领域的标准算法，服务器端的实现简单直接，在谷歌和微软的智能手机中也免费提供这样的认证应用。

Mesos 平台通过抽象出底层的计算资源，使得建立大规模可扩展的分布式系统变得更加容易。它可以用来为 Docker 提供一个调度层，或者充 当 AWS 之类云计算平台的一层抽象。Twitter 已经用它来助力扩展其基础设施，取得了很好的效果。基于 Mesos 之上的工具也开始出现，例如 Chronos，它是一种 分布式的，能容错的 cron 的替代工具。显著的成功案例已经出现，Apple 的 Siri 重新架构于 Mesos 之上。



AWS几乎在每个月都会发布大量的新特性，因此有时候，一项新的服务很难从这些特性里面脱颖而出。但是Lambda明显成功地吸引了大家的注意力。最开始它只支持JavaScript，但是现在已经加入了对JVM语言的相关支持（毫无怀疑还会有更多支持）。**Lambda**让你可以快速创建短生命周期的进程，来响应一个事件或者从API网关进来的访问请求。对于无状态服务，这意味着你不需要长时间保持机器一直运行，因而可以减少开销并且增强安全性。在AWS进军PaaS领域的尝试中，Lambda可能是最正确的努力。

Fastly，作为CDN市场中的一员，在ThoughtWorks的项目中的使用日益增加，同时也受到许多如GitHub和Twitter等家喻户晓的网站所亲睐。当你在寻找一个优越的缓存解决方案时，**Fastly**的功能，速度和价格都使它极具吸引力。我们也在从其他CDN解决方案向Fastly迁移的项目中看到了显著的成本节省。如果你正在寻找CDN解决方案，不如仔细研究一下Fastly。

预测分析在越来越多的产品中被使用，而且通常出现在面向最终用户的功能上。**H2O** 是一套非常有意思的开源工具包（其背后是一家创业公司），使预测分析可以为开发团队所用，他提供了简单和丰富的分析方式，具有极好的性能并且易于与 JVM 平台进行集成。同时它还集成了数据科学家最喜欢的工具，R、Python、Hadoop 和 Spark。

超文本传输协议严格传输安全策略(**HSTS**)是目前广泛被支持的策略，它能够使网站免于降级攻击。超文本传输安全协议中(HTTPS)降级攻击能够使网站的用户避开超文本传输安全协议

采用
31. TOTP Two-Factor Authentication

试验
32. Apache Mesos
33. Apache Spark
34. AWS Lambda
35. Cloudera Impala
36. Fastly
37. H2O
38. HSTS

评估
39. Apache Kylin
40. AWS ECS
41. Ceph
42. CoreCLR and CoreFX
43. Deis
44. Kubernetes
45. Linux security modules
46. Mesosphere DCOS
47. Microsoft Nano Server
48. Particle Photon/Particle Electron
49. Presto
50. Rancher
51. Time series databases

暂缓
52. Application Servers
53. Over-ambitious API Gateways
54. SPDY
55. Superficial private cloud

平台 接上页

(HTTPS)而使用超文本传输协议(HTTP)，从而使如中间人攻击等进一步的攻击成为可能。通过使用服务器头，你可以通知浏览器必须使用超文本传输安全协议(HTTPS)访问你的网站而且忽略通过超文本传输协议(HTTP)访问网站的降级攻击。浏览器对此策略的支持非常广泛，任何使用超文本传输安全协议的网站都应该考虑实行这个容易实现的功能。

弹性容器服务(**ECS**)是AWS进军多主机Docker容器市场的切入点。虽然在这个领域内存在诸多竞争，但目前为止还未出现太多像ECS这样基于云的解决方案。ECS看上去是个不错的开端，但我们担心它目前有些过于复杂，并且缺乏一个良好的抽象层。不过，如果你想在AWS上运行Docker，该工具毋容置疑是你应当首先考虑的。只是不要期望它很容易上手。

Ceph是一款可以运行在普通服务器集群上的用作对象存储，块存储和文件系统的存储平台。我们想在技术雷达中强调其为私有云的重要构件的作用。另外，Ceph还提供了一个非常有吸引力的组件RADOS Gateway，将对象存储中的数据通过与Amazon S3和OpenStack Swift APIs相兼容的RESTful接口的方式暴露出去。

如今容器在分布式集群环境下容器部署的需求场景正变得越来越多，**Kubernetes** 正是Google为解决这类容器部署问题而推出的容器集群管理框架。实际上Kubernetes并不是一个被Google在内部使用的解决方案，而是一个由Google发起并与外部贡献者一起维护的开源项目。Kubernetes支持Docker和Rocket作为容器格式，还提供了包括健康管理， 容器复制，和服务发现等在内的服务。 **Rancher**是另一个类似的开源解决方案，也提供了容器集群的部署功能。Rancher还提供了容器的生命周期管理、监控、健康度检查和服务发现等功能， 并且还包含了一个基于Docker的完整的容器化的操作系统。Rancher的优势在于其对容器化的各个方面的关注并且更加轻量级。

Mesosphere DCOS(数据中心操作系统)是建立在Mesos内核之上的统一资源调度平台。它在虚拟机集群的基础上提供了存储和计算资源池的抽象，以便能够在数据中心上运行极大规模的服务集群（已支持包括Hadoop、Spark和Cassandra在内的许多框架）。目前看来，这个平台对于那些负载不高的任务来说有些大材小用（这些场景下原始的Mesos也许更加合适），然而依然值得观察Mesosphere是否会在将来使得DCOS(数据中心操作系统)发展成为一个通用功能的系统。

与现代的基于Linux的云和容器解决方案相比，即使是Windows Server Core也显得大而笨重。微软刚推出了第一版预览版的**Nano Server**作为对策。Nano Server是一款只有400MB大小，移除了图形用户界面，32位支持，本地登录及远程桌面功能的裁剪版Windows Server。虽然早期预览版很难使用，而最终版也将被限制只使用CoreCLR，但对有兴趣运行.NET应用的公司来说，现阶段的Nano Server极值得考虑。

Presto 是一个开源的分布式SQL查询引擎，这个工具是为了运行交互式分析负载进行设计和优化的。Presto的大规模并行运算架构，结合了先进的代码生成技术和基于内存的数据处理流水线，使其具备极高的可扩展性。它支持ANSI SQL的一个大子集，包括复杂查询，联结，聚合以及窗口函数。同时Presto支持包括**Hive，Cassandra，MySQL**和**PostgreSQL**在内的各种数据源，从而可以针对一个组织内各种数据存储提供统一的交互式分析接口。应用程序可以使用JDBC接口来连接Presto。

Rancher是在容器集群部署领域与Kubernates类似的开源解决方案。Rancher提供了容器的生命周期管理、监控、健康度检查和服务发现等功能， 并且还包含了一个基于Docker的完整的容器化的操作系统。Rancher相对于其他解决方案的优势在于关注了容器化的各个方面并且更加轻量级。

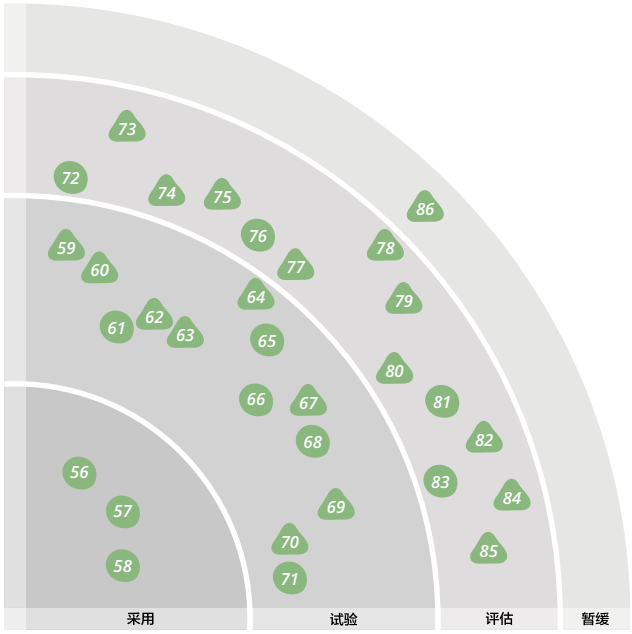
我们一直以来抱怨把业务逻辑放到中间件里的做法导致了应用服务器和企业总线“很有野心”去运行关键的应用程序逻辑。这样做的话需要在一些不大合适的环境中进行复杂的编码。我们最近看到了一些令人担忧的、**“雄心勃勃”的API网关**产品。API网关可以为我们提供一些通用的工具用来处理常见的问题，比如认证和限速的处理，领域中商业智能，比如数据的转换，领域规则，还是应该由依托于应用或者服务来处理，这样他们才能够被更接近于支持业务领域的产品团队来控制。

平台 接上页

不可否认，将应用程序或是服务部署到成熟的云平台带来了软件研发生产率的提升。这得益于团队自组织地部署和维护各自提供的软件服务的能力。然而，我们发现一些组织正在使用**伪私有云**，它给一些基本的虚拟化技术贴上了“云“的标签。团队通过虚拟化技术构建有限的服务，在统一控制的”企业蓝图“下很难提供定制化服务。这种服务的质量很难保障。由于网络，防火墙，存储之类的基础设施都需要手工搭建，部署节奏自然会受到限制。不同的组织应该慎重考虑使用这种不完整的私有云的代价。

工具

我们从ThoughtWorks一些使用**Browsersync**的团队中收到一些热烈赞扬的反馈。随着我们网站应用所支持设备的增多，花在跨设备测试上的代价也在不断增大。Browsersync是一个免费的开源工具，它能够通过同步多个移动设备或桌面浏览器上的手工浏览器测试来极大的降低跨浏览器测试的代价。通过提供命令行工具以及UI界面，Browsersync对CI构建非常友好，并且能够自动化像填写表单这样的重复任务。



过去，iOS以及OS X项目的依赖管理主要有两种方式：完全手工管理或使用CocoaPods实现全自动化管理。通过使用**Carthage**，一个新的中间地带成为了可能。Cathage管理依赖的方式是：它仅负责下载，构建并更新框架（frameworks），而把集成框架的工作留给项目本身。这和CocoaPods形成了鲜明的对比，CocoaPods基本上会接管项目的结构以及构建的组织。需要注意的是Carthage仅仅能处理动态框架（dynamic frameworks），而iOS7及以下版本的系统并不具备这个特性。

以前，我们建议boot2docker作为一种很容易在本地Windows或OS X机器上运行Docker的方式。现在，**Docker Toolbox**取代了boot2docker，同时它还加入一些其他工具。现在增加的工具包括用于容器管理的Kitematic，以及用于多Docker设置管理的Docker Compose（仅限Mac）。它可以安全的替代boot2docker，甚至他还会处理软件的升级。

越来越多的工具支持在代码库中安全地保存密码和访问令牌，上期雷达中我们就曾提到git-crypt和Blackbox。尽管已经有工具可用，但是人们依然不加防范地存储秘密。这种做法非常普遍，以至于有攻击者用搜寻软件寻找AWS的账号,然后用EC2实例挖比特币,攻击者得到比特币而AWS账号主人买单。Gitrob采用类似的做法，它扫描公司的GitHub代码库，找出不应出现在代码库中的敏感信息，然后标记包含敏感信息的文件。很明显，这是一种滞后做法，**Gitrob**提醒团队时已经（几乎）太晚了。因此，Gitrob只能作为一种损失最小化的附加工具。

Git可能会令人困惑，相当的令人困惑。就算是在一个简单的基于主干分支开发的过程中，它的用法也有足够的细微差别令开发人员时不时的纠结。问题发生时，是否理解Git背后的运作原理就显得格外重要了，Mac上**GitUp**就可以帮助你。

采用
56. Composer
57. Mountebank
58. Postman

试验
59. Browsersync
60. Carthage
61. Consul
62. Docker Toolbox
63. Gitrob
64. GitUp
65. Hamms
66. IndexedDB
67. Polly
68. REST-assured
69. Sensu
70. SysDig
71. ZAP

评估
72. Apache Kafka
73. Concourse CI
74. Espresso
75. Gauge
76. Gor
77. ievms
78. Let's Encrypt
79. Pageify
80. Prometheus
81. Quick
82. RAML
83. Security Monkey
84. Sleepy Puppy
85. Visual Studio Code

暂缓
86. Citrix for development

工具 接上页

你在命令行中输入的Git命令的时候，GitUp提供一个实时图形化界面显示正在发生写什么，这样你不仅可以掌握更多的Git命令，同时也更理解每个命令的功能。不管是否有Git经验，GitUp都是一款好用的工具。

我们的一些 .NET 项目的团队推荐使用 **Polly** 来帮助我们构建微服务系统。它鼓励使用基于流畅表达式的透明错误处理机制，还包含了多种断路模式（Circuit Breaker Pattern），如重试、不断重试、稍后重试。在其他语言中已经存在了类似的程序库，比如Java中的Hystrix，而 Polly 是 .NET 社区中一个很好的补充。**Brighter** 与 Polly 可以很好的集成，Brighter 是另一个 .NET 平台上的开源小类库，他提供了实现命令调用的脚手架。结合这两个类库，提供了熔断模式的功能，是在端口－适配器模式和 CQRS 的上下文中尤其有用。我们的团队发现他们能够在一起工作的很好，虽然是可以被单独使用的。

许多监控工具是建立在机器或实例这样的概念之上的。凤凰服务的模式、Docker这样的工具被越来越多的使用，使得这样的监控方式在现代的体系结构下越来越无助：实例变的短暂而服务长存。**Sensu** 允许一个实例把自己注册为一个特定的角色，然后在这个基础上监控它。一段时间之后，来来去去的不同实例都会扮演这个角色。鉴于这些因素和该工具的日益成熟，我们认为是时候把 **Sensu** 放回雷达了。

尽管**SysDig**已经不是技术雷达上的新宠，我们仍然惊讶地发现很多人还是没有听说过它。作为一款开源的Linux系统诊断命令行工具，SysDig具有一些非常强大的功能。生成问题机器上的系统记录，使你之后能进行查询检测来找出原因是我们非常喜欢的功能之一。SysDig对容器的支持使本已十分有用的工具如虎添翼。

许多开发团队正在从简单的持续集成迁移到持续交付流水线，往往通过多个环境的验证，最终部署到产品环境。 为了成功实现这类构建管道，并且使之持续适应交付的变化，势必需要一个将构建管道和构建产物视为一等公民的CI/CD工具, 然而不幸的是我们的选择并不多。在我们的团队尝试使用**Concourse CI**之后，这个CI/CD领域的后起之秀以其便捷的设置获得了整个团队的青睐，如构建在容器中执行，干净实用的用户界面，以及限制**雪花构建服务器**。

Espresso 是一款安卓系统功能测试工具。它微小的内核API隐藏了复杂的实现细节，并帮助我们写出更简洁、快速、可靠的测试。

Gauge是一个轻量级的跨平台的测试自动化工具。技术规格都由形式自由的Markdown写成，因此，测试用例可以以商业语言写成并且能够结合到任何现有的文件格式。

不同编程语言是以实现某个核心实现插件的方式进行支持，这确保跨语言实现的一致性。这个工具，由ThoughtWorks开源，同时支持对所有支持平台的并行执行。

尽管IE浏览器的使用量日益萎缩，但对很多产品而言IE浏览器的用户群依然不可忽视，浏览器兼容性仍然需要测试。这对于喜欢使用基于Unix的操作系统进行开发的人来说还是件麻烦事。为了帮助解决这个难题，**ievms**提供了实用的脚本来自动设置不同的Windows虚拟机镜像来测试从IE6到Microsoft Edge的各种版本浏览器。

尽管用HTTPS保护用户并提高web完整性的站点与日俱增，但仍有很多站点没用HTTPS。此外，我们看到越来越多的人在企业内部用HTTPS来提供附加的安全保障。在更大范围采用HTTPS的最主要阻碍是申请证书的流程。且不论费用高昂，流程本身也华而不实。用**Let's Encrypt**吧，这是一个新的证书权威机构，旨在解决上述问题。首先，他免费提供证书。其次，更重要的是他还提供极其易用的命令行API，很容易完全自动化证书发放、升级以及安装的流程。我们认为，Let's Encrypt现在还处于beta阶段，未来将具有革命性意义，他能帮助更多站点使用HTTPS，还是安全方面一个好的、自动化工具的标杆。

Pageify是一个Ruby库，用于为针对UI的自动化测试构建页面描述对象。它关注于更快的执行测试以及代码的可读性。它提供简单的API用于动态的构建，维护及断言页面描述对象。使处理复杂层级的DOM的处理代码更加可读，并可以很好的配合**WebDriver**或是**Capybara**使用。

SoundCloud 已于近日开源了监测和报警工具包，**Prometheus**。为了应对其产品系统中使用 Graphite 时的一些困难，Prometheus 主要支持基于拉的HTTP模式（虽然更像 Graphite 的推模式也有支持）。它通过进一步支持报

工具 接上页

警，使之成为你的运营工具集中的一部分。写这篇文章的时候，Prometheus 刚刚发布了 0.15.1 版本，但它发展很迅速。我们很高兴地看到，最近的一些产品专注于核心时间序列数据库和多维索引能力，同时允许导出到更广泛的各种前端图形工具中。

随着越来越多的服务提供了RESTful API，如何文档化这些API变得犹为重要。在之前的技术雷达里，我们提到了Swagger。在这期的技术雷达里，我们想强调一下RESTful API建模语言(**RAML**)。与Swagger相比，我们的团队认为更为轻量级的RAML将关注点由如何文档化现有的API移到了如何设计API上。

Sleepy Puppy是Netflix公司近期开源的一款盲打XSS收集框架。当攻击者试图入侵第二层系统时,这个框架可用于测试目标程序的XSS漏洞。XSS是OWASP的Top10的安全威胁，Sleepy Puppy可以用来同时为几个应用完成自动安全扫描。它可以自定义盲打方式，简化了捕获，管理和跟踪XSS漏洞的过程。Sleepy puppy还提供了API供ZAP之类的漏洞扫描工具集成，从而支持自动化安全扫描。

Visual Studio Code是微软出品的一款免费的IDE编辑器，并支持跨平台。我们发现它提供的Git集成功能对提升持续集成能力有很大的帮助。Visual Studio Code同样提供了很好的外部工具集成功能，例如通过自动感知grunt/gulp任务功能可以在IDE中非常容易的直接运行grunt/gulp任务，避免了与命令行终端的切换。随着Docker生态系统的快速发展，这款IDE同样提供了对于dockerfile的snippets以验证的支持。

很多组织依然强迫分布的或者离岸开发团队使用**Citrix远程桌面来开发**。虽然这种方式提供了一种简单的安全模型－资产理应从不会离开该组织的服务器－使用远程桌面开发绝对会戕害开发者的生产力。如果你要强行把分布式和远程桌面的负担加诸于开发人员，便宜一些的每小时收费是完全没有意义的，我们希望更多的离岸外包商能够在客户面前承认这些缺点。更好的做法是，要么使用一个“净室”在保证环境的安全性的同时进行本地开发，要么使用托管的 IDE（比如ievms）。

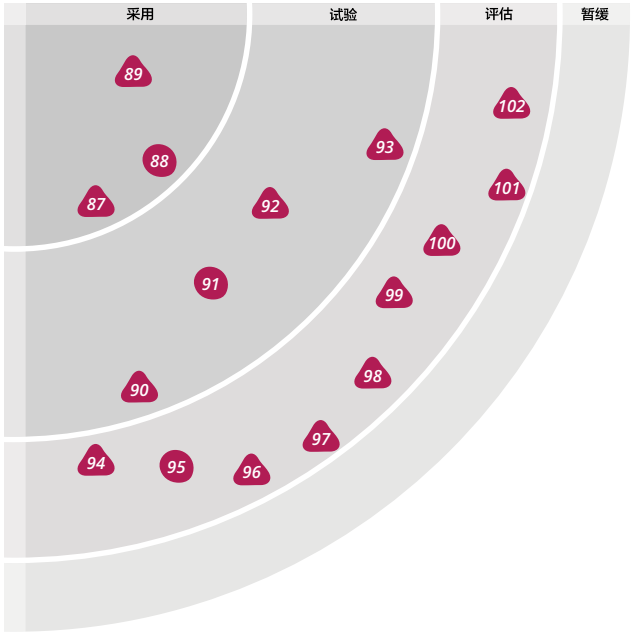
语言和框架

在过去这些年，JavaScript不断成长，现在几乎成为这个世界上最被广泛使用的编程语言。然而，这个语言他自身存在的一些问题导致很多人通过使用库，甚至在JS之上实现自己的语言（就像我们之前提到的CoffeeScript和ClojureScript）来解决JavaScript自身的问题。ECMAScript 6，JavaScript的新版本解决了很多现在使用的老版本的问题。尽管浏览器的支持还不足，但是像Babel这样的编译工具的支持使得我们可以编写**ECMAScript 6**并且让它支持老的浏览器。对于新的项目，我们强烈建议从项目一开始就是用ECMAScript 6.

在首次公开亮相一年之后，现在**Swift**是我们在开发苹果生态系统时的默认选择。随着最近发布的Swift 2，语言层面达到了一定的成熟度，并且其稳定性和性能能够满足大多数的项目的需求。Swift 仍然有一些问题，特别是在工具的支持、重构和测试方面。然而，我们觉得这些都不足以让我们避免使用 Swift。同时，移植大的、现存的 Objective-C 代码库不大像是个划算的选择。宣布 Swift 将要开源是另一个很好的信号。我们希望这不仅仅是另一个只是把内部的代码丢到公共仓库上的例子，因为苹果已经明确表示，社区贡献是被鼓励和被接受的。

Mustache或FreeMarker等模板框架都把代码和标记混杂在同一个文件中以实现复杂的动态内容。**Enlive**是基于Clojure的模板框架，她彻底分离程序代码和HTML标记，用CSS选择器查找并替换文档片段。Enlive展示了函数式编程的威力，用一组简单的、可组合的函数作用于通用的抽象结构来实现复杂行为。我们使用Clojure的团队认为Enlive是个既实用又直接的工具。

我们对于HTML5的WebSockets技术仍然有很多疑虑。无状态，请求/应答模型是支持现今互联网的基础，但是WebSockets允许服务器端来初始化浏览器端行为，这种方式已经和这个模型背道而驰。安全性是WebSockets另一个



隐患，例如在WebSockets标准中，对于跨域没有任何的限制策略。不管怎样，我们认为在一些监控、报警应用程序中，WebSockets可以大显身手。如果你需要构建一个基于.Net的WebSockets服务器，**SignalR**实现了构建健壮、实时应用的大部分代码。这包括一些推荐的安全实践，例如对于连接令牌（Connection Token）的验证、按需激活SSL加密等。尽管ThoughtWorks团队对于SignalR非常满意，但是我们建议在你选择之前还是需要慎重考虑WebSockets所带来的根本性问题。

使用**Spring Boot**可以轻松设架设独立的基于 Spring 的应用，它非常适合搭建新的微服务并且易于部署。通过更少的样板代码，它降低了数据访问时 Hibernate 映射所带来的痛苦。Spring Boot 简化了基于 Spring 的服务，这是我们所喜

采用
87. ECMAScript 6
88. Nancy
89. Swift

试验
90. Enlive
91. React.js
92. SignalR
93. Spring Boot

评估
94. Axon
95. Ember.js
96. Frege
97. HyperResource
98. Material UI
99. OkHttp
100. React Native
101. TLA+
102. Traveling Ruby

暂缓

语言和框架 接上页

欢的，然而同时，我们也知道一定要对依赖保持谨慎。毕竟，潜伏在表面之下的依然是 Spring。如果你在使用 Java 写微服务，你也可以考虑使用 **DropWizard** 或者像是 Spark 这样的微框架，一方面能够得到 Spring Boot 的好处，另一方面又不会像 Spring 那么重型。

把**CQRS**作为一个一般模式我们仍然有些顾虑，尽管如此，这个方案在特定的情况下非常适用。但是在这些情况下要正确地使用CQRS，开发者仍然有很多工作需要去做。构建于JVM平台的**Axon**框架可以很大程度地解决这个问题，借助于它，我们已经在一些项目上取得了成功。尽管目前看来，Axon框架并不是一个非常完美的技术方案，但是它正在持续演进。相比于从头开始构建所有东西，有时使用Axon会显得更有意义。

和许多其他编程语言一样，极客们最喜爱的编程语言之一——Haskell，现在也以**Frege**的形式出现在JVM平台。它把纯粹的函数式编程语言引入该平台，并支持和其他JVM语言和库的简单互操作性。

HyperResource是一个构建RESTful API客户端的Ruby框架，它接受**HAL**格式的JSON响应，动态地生成包含超链接（hypermedia links）的对象模型， 我们推荐它的原因在于它在Richardson REST成熟度模型中处于第三层，具有更好的服务发现性（discoverability）和协议的自描述性（self-documenting）。

Material UI 为实现了谷歌的Material Design 语言的 React 应用程序提供了可复用的组件。它填补了一个类似于 Twitter Bootstrap 的一个空白区域，它既能让你的程序快速的跑起来，又不会随着应用程序的增长暴露出同样的缺点。 除此之外，Elemental UI 也是值得研究的替代选择。

OkHttp是来自Square的一个Java HTTP连接库，他支持更快的HTTP/2协议并且提供流畅的连接创建接口。即便是用HTTP/1.1，OkHTTP也能通过连接池和透明gzip压缩提高性能。支持同步阻塞调用以及异步非阻塞调用。可作为Apache HttpClient的直接替代品。

作为跨平台移动开发世界里的新成员，Facebook的**React Native**将React.js的开发模型引入给了IOS和安卓的开发者。React Native程序使用JavaScript语言开发，但是并不像其他的混合式开发框架一样（例如Ionic），React Native给予了开发者在目标平台调用原生UI组件的能力。这种方式我们也在其他框架看到过（例如Calatrava），但是 React Native已经就着React.js的势头，激活了巨大的开发者社区。这种架构将会在未来的移动应用开发领域扮演一个非常重要的角色。

基于微服务架构的系统要求我们更深刻地思考如何处理故障隔离和测试。**TLA+**是一种正式的语言规范，可以被用在这两种场景中。针对故障隔离，TLA+可以识别系统的不变量，从而直接进行监控。比如，一个系统的不变量可以是不同服务间的请求数量比例，那么超过这个比例就会发出警报。TLA+还可以识别分布式系统中细微的设计缺陷，比如，亚马逊用TLA+编写了模型检测的规范，从而能够在Dynamo DB发步到产品环境之前识别那些细微的缺陷。对于绝大多数系统来说，去编写规范然后进行模型检查的投资成本可能过大。而对于那些非常复杂的关键系统，或者有一定数量级用户的系统，我们认为TLA+是非常有价值的，值得去学习。

Traveling Ruby 使得发布可移植的、开箱即用的，平台无关的 Ruby 二进制文件变成可能，并且它不需要安装解释器、包，或者是额外的gem。他解除了 Ruby 应用执行环境与其开发环境之间的耦合。

ThoughtWorks是一家集合了在软件咨询、交付以及产品领域富有激情并且极具前瞻性的技术工作者的软件公司。我们利用颠覆性思维帮助客户成就非凡使命，同时致力于IT产业乃至社会的变革。我们为追求卓越的软件团队提供创造性的工具。我们的产品帮助企业不断进步，不断交付满足他们重要需求的高质量软件。ThoughtWorks已经从20多年前一个芝加哥的小团队，成长为现在拥有超过3500人，分布于全球12个国家，拥有35间办公室的全球企业。这12个国家是：澳大利亚、巴西、加拿大、中国、厄瓜多尔、德国、印度、新加坡、南非、乌干达、英国和美国。

ThoughtWorks®