云生态 | 专刊

Cloud Ecosystem

第4期
Vol.04

打造中国最优质的云生态媒体

热点回顾

技术热点

利用JS在AWS上构建大规模弹性Web应用

Netflix的网站优化经验

对话大咖

红帽OpenShift总经理访谈录

观点&趋势

腾讯游戏是如何使用Docker的

为什么你应当将应用迁移到云服务上

生态圈新闻

云生态专刊 04

本期主编 魏星

流程编辑 丁晓昀

发行人 霍泰稳

联系我们

提供反馈 feedback@cn.infoq.com

商务合作 sales@cn.infoq.com

内容合作 editors@cn.infoq.com

魏星 InfoQ技术编辑，热爱互联网安全与开源，信奉技术的力量。致力于做一些有价值的事，并且越来越慢。人生理想：壕，不犹豫。

卷首语

不一样的云，一样的安全

“保障数据安全，是实现云计算环境的根本”，赛门铁克企业产品和服务集团总裁 Francisde Souza 表示，“如果安全性无法满足，那么迈入云端就没有任何意义”。毫无疑问，安全是云计算的一块基石，云安全问题不率先得到解决的话，云计算的发展必将受阻。一方面，云计算提高了企业的网络防护能力，另一方面云计算也使安全风险更为集中，网络安全边界一旦被突破，后果往往是灾难性的。云计算市场研究机构 Cloud Tweaks 指出，云计算安全有一些基本要求，包括数据存储制度，实现云资源访问控制的用户识别管理系统等。

据了解，云服务的资质或安全认证至少包括 IDC/ISP 牌照、可信云认证、C-STAR 认证、ISO27001、等保三级、国家网络安全审查，等等。资质认证固然是安全建设的重要方面，但云安全的要求往往更高。在阿里云安全峰会上，安全专家吴瀚清提出了云安全要做到“态势感知”的理念。

2015 年 6 月 11 日，阿里巴巴集团宣布收购被誉为“中国的 FireEye”的翰海源，后者将整体并入阿里巴巴安全部。“毫无疑问，安全是云计算的一部分，也是阿里云的核心竞争力，我们需要提前为即将到来的挑战进行安全研发投入。”阿里巴巴安全部资深总监肖力表示。

在微软亚太研究中心的一次媒体见面会上，微软大中华区首席法律顾问 Tim Cranton 表示，微软公司在处理安全问题上积累了非常丰富的经验。微软美国总部法务部下设有数字安全中心，专门处理和网络安全有关的问题；在中国，微软成立了数字安全研究实验室与中国的网络安全专家、消费者和计算机应急中心等执法部门进行合作，以保障 Microsoft Azure 的安全性。

IBM Bluemix 中国研发中心总经理兰健在谈到安全问题时反复强调，Bluemix 入华十分重视合法、合规建设。今年 7 月正式发布的华为云则是全球首个通过 C-STAR 认证的云服务厂商。

在数据存储安全方面我们看到，用户对 Oracle 产品进行安全测试一度引起官方的反感，PostgreSQL 则表示欢迎用户使用开源产品。RedHat 的 Ceph 社区总监 Patrick McGarry 认为，商业存储和开源存储当前各有市场。截至到 2015 年 5 月，Ceph 在块存储方面的市场占有率高达 44%，并且还将继续增长。此外，对象存储技术如 NAS、SAN 也在各云厂商中得到了广泛应用。大型互联网公司如 Netflix，已经放弃自己的机房而全部采用亚马逊云服务。

随着企业产品与服务的不断云化，用户对云服务的需求也在不断提高。我们相信，在做到高效、稳定的同时，能保护好用户数据的云厂商将赢得最终的胜利。

研究：云开发者最赚钱

作者 Abel Avram 译者 谢丽

VisionMobile 发布了《开发者经济：2015 年第 3 季度开发者国度状况调查报告》。调查显示，大多数开发者都是男性且为年轻人；Windows 引领桌面开发，浏览器次之；开发者喜欢将代码保存在私有云上，而且云开发者赚钱最多。

调查得出的其中一个首要结论是绝大多数开发者都是男性，女性开发者占比在 3%（南美）和 10%（北美）之间不等。此外，开发者的平均年龄约为 30 岁。（见表 1）

研究发现，在移动领域，iOS 在高端领域仍然处于领导地位，iOS 开发者从每笔交易中赚得的钱也更多，但是，“在接下来的几年中，通过自己的 Android 设备进行购买的用户数量除了弥补这种差距之外还有盈余。”因此，Android 是最流行的移动平台（71%），iOS 排第二（51%），Windows 排第三（27%）。许多开发者都针对多个平台进行开发，37% 的开发者针对 Android 和 iOS，11% 的开发者针对所有三种主流的操作系统。

44% 的开发者表达了从 Windows Phone 迁移到 Windows 10 的意愿，但由于市场份额较小以及过去 6 个月中所显现出来的开发者兴趣下降（从 30% 降至 27%），研究者预计，针对微软移动平台的开发者会减少。

在桌面领域，Windows 是主要的操作系统，41% 的开发者将其作为主要的目标平台。接下来是浏览器，占据 37% 的份额，然后依次是 Linux（11%）和 Mac OS X（6%）。研究者发现了一件有趣的事，就是有 2% 的开发者为 Chrome OS 开发应用，而后者的“市场份额可以忽略不计”，因为：

开发者对谷歌瘦客户端的兴趣部分来自好奇，作为一项新技术，开发者热衷于了解其能力和局限，而谷歌也一直在努力使开发者可以很容易地为其平台打包应用程序。将一个 Web 应用针对 Chrome OS 打包微不足道，但开发者正在这样做，这一事实表明，开发者对使用瘦客户端越来越感兴趣。

表 1

地区	男性（%）	平均年龄（年）
北美	90	34
南美	97	30
欧洲	96	32
非洲	91	27
亚洲	95	27
大洋洲	93	34
全世界	~94	~30

表 2

月收入	移动（%）	桌面（%）	IoT（%）	云（%）
$0	19	15	27	15
$1–500	32	34	32	28
$500–5000	21	22	20	23
$5–10k	7	5	5	6
$10–50k	8	7	7	10
$50–500k	9	10	5	9
$500k+	4	7	4	9

研究表明，出于安全和弹性的考虑，相比公有云，大多数云开发者（超过 50%）都更喜欢私有云。其他开发者分布情况如下：AWS（16%）、Azure（13%）、谷歌（8%）、Digital Ocean（4%）、Heroku（2%）。云开发使用的主要语言为：AWS - Java（24%）、PHP（21%）；Azure - C#（54%）；Google - Java（35%）、Digital Ocean - PHP（29%）、Python（22%）；Heroku - Ruby（31%）、JavaScript（29%）。

在月收入方面，只有少数人可以赚到大钱，云服务开发者最赚钱，桌面开发者次之。而在移动（41%）和 IoT（59%）市场，有许多开发者的月收入都低于研究者设定的每月 500 美元的“贫困线”。（见表 2）

这一期开发者经济研究调查了来自 149 个国家的 13000 多名开发者。要了解更多有关调查方法的细节或者研究所涉主题（包括编程语言偏好、IoT 参与度、移动电子商务），请下载报告。

云迁移的核对清单

作者 Richard Seroter 译者 邵思华

软件组织正在快速地实施云技术，但迁移始终是一个无法回避的挑战。哪些部分是需要你密切留意的？哪些应用程序更适合于进行迁移？如何对应用程序进行重构以适用于云端？经历了这一转变的先行者为我们留下了什么启示？在这一系列文章中，你将从那些在帮助企业成功地迁移至云环境方面富有经验的专家那里获得实用的建议。这一领域应得到高度关注，我们希望你也能够参与这方面的讨论。

如果你正准备将应用程序搬到公有云环境，那么你正赶上了这一波潮流。根据 451 Research 的调查显示，在 2015 年整个 IT 界有 46% 的成本都是针对非本地环境的系统，并且这一数字在今后三年内有望攀上 50%。企业仍然将节约成本视为购买云服务的主要原因，但越来越多的组织开始将“尽早推向市场”、“更好的可用性”以及“建立新的收益流”等战略目标作为云实施的主要动力。不过，根据近期的一份调查问卷表示，在所有的参与者之中，只有 27% 的组织对于他们的云迁移体验感到非常满意。如何将云服务引入你的 IT 环境，并且成功地将应用与数据迁移到新的环境，这方面显然还存在着很大的讨论空间。

在本文中，我们将探索整个云迁移周期中的四个阶段。可以肯定的是，（云）迁移是一项不会“真正”结束的任务。随着你不断地引入新的服务以解决新的业务问题，这一过程会不断地重新启动，只是其技术与目的可能会有所不同。以下这份清单也许能够帮助你踏上这一旅程，但重要的是定义一份属于你自己的清单，以适应于你的特定情况。

评估

没有一种云服务能够适用于所有组织，许多组织一上来就试图确定一个首选的云环境，却并不了解这一选择是否与组织的成熟度、文化以及应用组合相匹配。那么，有哪些问题是你和这些潜在的提供商需要首先解答的呢？

你是否适用于这个云环境？云服务的提供商多如繁星，并非其中每一个都适合你的特定需求。有些云环境能够在几秒钟之内创建几千台虚拟服务器，如果你的组织正好需要这样的能力，那自然是一个好的选择。有些云环境能够提供服务周到、并且质量上乘的托管服务，但这些服务未必与你的运维模式相匹配。你需要考虑对你的组织来说哪些因素是最重要的，并且积极地尝试一些在成本允许范围之内的云提供商。请避免盲目地选择某个特定云领域中的“领先者”，这一选择可能会无意间对于你成功地交付服务带来负面的影响。

你对于运行能力的需求是怎样的？根据你希望迁移的应用的类型的不同，能够选择的提供商也有所不同。你的应用程序组合是否由大量现代的、适应云的应用程序组成，能够充分利用大量廉价的、临时的服务器或容器？你是否需要一个增长速度相对缓慢的池，但它能为你提供几百个持久的计算资源？仔细考虑你的应用特性，确保所选择的提供商能够满足你对于运算能力、存储能力以及网络吞吐需求的上限。

实际的成本是多少？“云成本”并不是在价格表上可以一目了然得出的数字。确保你对实际应用场景进行了建模，并且考虑对于跨区域分发、存储备份、带宽消耗、API 调用等需求的针对性收费。此外，别忘了将项目上线、支持计划以及创建新环境（例如性能测试与预发布环境）等这些在本地系统中不存在的成本也列为考虑因素。

应用的“缺陷”在何处，这些缺陷在迁移后是否依然存在？在经过对云提供商的首轮评估后，你可能已经建立了一个已预见的、或是实际存在的缺陷列表。因此，你应当与提供商进行交流，以寻找 a） 是否有一些未来的计划能够弥补这些缺陷，或者 b） 是否有些可行的临时方案能够弥补这些缺陷。一种可能的临时方案是对于引起这种缺陷的基础设施或是应用模式进行重构，让其更好地适应云提供商提定义的模型。

你现有的工具能否适应这一环境？如果你在同一家公司已经待了一段时间，那么很可能已经习惯于某些现有的工具（及流程）了。如果你无法轻易地让你的工具运行在所选择的云环境中，请确保这个供应商所支持的工具集能够取代你现有的工具，而不会使你感到不快。

计划

恭喜你，你已经为一个特定的业务领域选择了一个云提供商。但现在才是最困难的部分：对迁移进行计划！在设定这一迁移策略时，有哪些东西是你需要着重考虑的呢？

这次迁移包括的应用、功能和环境是怎样的？在理想的情况下，最好不要将最复杂的、对业务影响最大的应用放在第一次云迁移的计划中。但不管怎样，请确保你为所有的应用列出一份详细的清单，并指出其中最适合你迁移到云环境的选择（无论是策略上还是战术上）。

整个应用程序架构的结构是怎样的？你的应用程序架构很可能会在云环境中产生变化。云服务器、数据服务和网络与本地环境的行为相比都有所不同，因此你可能要对你的引用架构以及部署流程进行一些改进，让它们能够适合这个全新的世界。

新的环境可能会带来哪些性能瓶颈？迁移到云环境就能够保证让应用程序的性能得到飞跃性的提高？绝非如此。如果你将一个一体性应用放到云端，并将它的各个部分分布到多个云服务中，这种方式可能会引入意外的延迟。而如

果你的应用程序还与未迁移到云环境中的本地系统有所交集，由于两者之间的远距离连接，也有可能会导致性能问题。云服务器提供了多种不同类型及能力的系统，请确保为你的应用程序选择必要的CPU、内存和磁盘能力，以满足甚至超过之前的性能水平。

你的混合式集成计划是什么？如果你认为能够一次性将整个应用组合都迁移到云端，这种想法绝对是不切实际的，甚至可能永远不会发生。你需要将云环境视为你的现有环境的一种逻辑延伸，在此基础上再考虑如何将现有的数据、网络以及认证扩展到云端。

你是否已经确定了第一批应用者？云迁移过程或许会产生破坏性，因此重要的是在组织中找到合适的人，他们非常乐于帮助你定义新的标准，并在这次重要的转换过程中为整个组织提供指导。

用户如何访问这一环境？你的同事大概已经有许多要记住的密码了，如果你还要引入一整套全新的复杂的证书，用以访问一系列关键的云服务，那是他们绝对不想看到的。可以研究一下你的云提供商所提供的单点登录（SSO）机制，对于任何打算采用云服务的项目，都将这一点作为一个前期设计的考虑因素。

你如何对员工进行培训？让你的整个团队深入地了解新的云服务是非常重要的。你还要考虑到所有的受众（例如项目经理、开发者、架构师、系统管理员等等），并制订一些培训资料，帮助员工适应新的云环境。

为了充分利用新的服务，需要对内部流程进行哪些改变？现有的硬件申请、变更管理、测试以及部署等已确定的过程或许在使用云服务时会产生某种变化。如果你打算在这个更敏捷、自服务的环境中照搬现有的流程，可能会失去你原本计划进行云迁移时所构想的各种优点。因此，对于必须改变的部分，需要进行一次坦诚的评估。

你如何将新的代码、数据和配置部署到新的环境？新的云服务未必能够配合现有的系统管理工具，如果你正在使用现代配置管理工具，例如Chef或Ansible，或许有某种扩展能够让它在你的目的云环境中无缝地运行。请确保你对于部署和配置流程进行一次全面的模拟，并确定需要改进的地方。

在迁移后对服务进行运维的计划是什么？迁移只是应用在新的云环境中运行的起步，如果需要进一步的改动，又该如何处理？如果发生了某种问题，如何进行问题诊断？你会使用何处监控工具用于分析关键性能指标？仔细观察在云环境中应用请求的整个生命周期，考虑所有可能的调整与维护操作。

你是否设计了某种接入策略？如果你的应用程序用户能够忍受在迁移过程带来的较长的停机时间，那么你或许能够继续使用某种简单但具有破坏性的接入计划。但如果你希望将停机时间降至最低，那么你必须考虑应用某些策略，在你搭建新环境的同时与主环境始终保持同步，直到将所有访问都转移至新的实例为止。

整个财务流程如何进行？没错，你很可能需要为云存储付费，那么怎样处理这部分费用呢？是计划对每个使用云资源的业务部门分别收费，还是选择从共用资金中抽取一部分进行全额支付？请确保你已经仔细地考虑了整个付费流程，在支付的同时记得获取发票。

你是否已经进行了一次小规模的试用，并已对以上关注点建立了相应的计划？无论如何，不要仅仅进行了一些书面原型设计就开始进入迁移过程。你需要进行实际操作，在目标云平台上搭建应用程序并进行试用。在试用过程中熟悉整个环境的界面、功能以及各种限制，以这种方式获得实践性的知识。

迁移

在经过了适当的前期计划之后，迁移过程本身应当是波澜不惊的。可以肯定的是，在实际的迁移过程进行时，总是会出现各种意想不到的情况。但如果你已经能够解答以上这些问题，那么你的团队应当有能力处理那些意外情况。

你如何将应用及数据分布在云环境中？有许多方法能够将你的应用及数据保存在云端。对于中型应用来说，你可以选择使用简单的copy命令，通过网络连接传递数据。但对于大型数据集来说，这可能会让你的云提供商为此收取大量的带宽费用，还会延长数据转移时间。在这些情况下，可以采取一些更好的方式。 (a)将数据压缩后拷贝到目标云环境中的某个存储位置，之后再将其转移到最终的目的地。(b)将物理数据磁盘转移至云提供商处（前提是这个云环境支持这种操作）。

在传输过程中设置了哪些安全控制手段？在迁移过程中，你可能会用到预发布服务器或临时对象存储库。请确保你已经完整地考虑了数据与访问安全方面的问题，尤其是敏感的数据。

迁移虚拟机。将完整的VM进行迁移是一种将应用迁移至云端的方式，但根据这些VM如何在本地环境网络中设置的情况，有可能会发生意料之外的副作用，例如这些VM所在的域、它们所用的磁盘类型，以及其它各种情况。虽然将VM进行“Lift and Shift”式的整体迁移通常来说是云迁移的最简单方式，但它的复杂性往往走出想象。此外，这种方式不会促使你重新考虑应用程序的架构，以及为了应对云环境而对应用进行重构的策略。

迁移数据。你的数据可能会迁移至某个数据即服务环境，或某种自托管的数据库实例中。请确保你了解提供商具备哪些可用的工具，以及这些工具在数据容量或结构方面存在着哪些限制。

迁移应用。如果你的应用程序部署工具能够原生支持指向云基础设施、容器或应用平台，那说明这个工具功能比较出色。但你也可能属于尚未具备这一能力的少数派！可能需要花上一些时间对你的本地工具进行一些重新配置，才能够将代码部署到云端，或者你也可以试用一

些新的工具，以使整个流程更顺畅。

重建元数据。许多公司总是表示，他们希望获得云端的“可移植能力”，而试图避免“绑定”在某个特定提供商的云平台上。这个……祝你好运吧。虽说像虚拟机或应用程序的代码这些资源可以相对比较容易地搬到云环境中，但环境元数据往往是特定于提供商的。每个云平台的帐户结构、用户、权限、策略、负载均衡器等等都是不同的。请确保你了解如何在特定的目标云平台上建立这些支持性配置信息。

验证

当迁移过程结束后，必须对应用程序进行全方面验证，以确保它完全按预期的方式运行。

应用是否可访问？这种测试很简单，但重要的是要全方面地检查应用服务的方方面面，确保用户能够访问这些服务，而且内部的组件也能够互相通信，并且没有出现错误。

是否所有的数据都已经正确地传输到云端了？通过自动化的检查，或者在最糟糕的情况下可以采取手工检查，以检验是否事务型数据以及引用数据都已经成功地传输至云端。如果你的数据关系非常复杂，那么一旦迁移过程出错，有可能会造成一连串的问题。

管理工具能否访问云环境？正常情况下，在计划阶段就应该对这一点进行校验了，但现在是最后一次机会，以验证所有的管理工具都能够访问云环境中的应用程序，并且能够对其进行监控，而不出现任何问题。

总结

每一天都有新的组织在成功地实施云服务，通过将应用程序迁移至这个更敏捷的环境，为他们的IT环境引入了新的活力。而在迁移过程中，不实际的期望最有可能导致整个过程的挫折。要摆脱这种焦虑，最好的方式是对组织的目标与意愿进行详细的评估，对于计划中的迁移流程中的各种问题给出实践性的回答。

你对于提高云迁移的成功率还有其它建议吗？请在留言中给出你的反馈！

2015 年 6 月的红帽峰会上最大的新闻莫过于 OpenShift v3 的发布，该版本最大的特点就是对 Docker 提供了原生的支持。此外，大会上频繁露脸的 Atomic 项目及相关产品线也均是围绕容器技术建立的技术栈。

下面，InfoQ 中文站对红帽 OpenShift PaaS 业务部门的负责人 Ashesh Badani 进行了采访，了解他对于 PaaS 市场、容器技术发展的一些看法，以及对 OpenShift 的后续计划。

InfoQ：根据目前大家所看到的，全球的 PaaS 市场在过去这几年发展的并不好。您觉得这是为什么？

Ashesh Badani：我认为这是一个市场定义的问题。其实我们红帽的 OpenShift 产品线过去几年发展的非常迅猛，只不过不同的时期我们总会看到不同的需求。2012 年我们的客户最关注配置管理方面的改良，2013 年的需求更多的转向持续集成和交付工作流，2014 年的需求更多的转向 DevOps 以及与应用软件开发生命周期的深度集成，而到了今年，大家对大型集群资源中的不可变应用组件更感兴趣。

说到底，“PaaS”这个概念就是个大杂烩，只要是对业务有意义的都被归纳到这个概念下面，市场其实是很大的，现在 Cisco、CA、Amadeus 和 T-Systems 等大型企业现在都在使用 OpenShift（详见 OpenShift 客户列表）。而且自从 6 月份我们在红帽峰会发布 OpenShift 3 以来，也有很多人来询问，现在已经有 100 多家企业 / 组织加入了 Beta 测试。我们的 OpenShift 在线版也一直发展的很快，现在已经有 250 多万应用托管在上面，而且应用数量每年都在成倍增长。

InfoQ：最新发布的 OpenShift 3 全面拥抱了

Docker 生态圈，我们听说它几乎整个重写了一遍，把以前那一套 gear/cartage 技术都扔了。从你的角度来看，Docker 对于一个 PaaS 业务的发展有什么特别的意义？

Ashesh Badani：对于客户和 ISV 而言，打包格式和 API 的统一化是非常有意义的，这也是诸多公司（包括红帽、IBM 和微软等）联合发起开放容器项目（Open Container Initiative）的原因。容器打包格式的标准化给用户带来了极大的灵活性和一致性，因此这是我们发展的方向。

其实 2011 年发布的第一版 OpenShift 就使用了容器技术，包括 Linux 进程和网络隔离、用 cgroups 和 namespaces 提供高密度高效率的应用环境。现在已经有上千计的应用以通用的 Docker 格式发布为镜像，这让应用普遍具备良好的可用性和可移植性。无论对于 OpenShift 平台的用户、合作伙伴还是供应商，这都是很好的事情。

OpenShift 3 跟之前版本不同的地方在于它对于 Docker 容器的原生支持，这意味着你可以不受限制的在平台上运行任意的 Docker 镜像，而不像以前那样可能会在部署和管理的过程中遇到一些问题。当然了，安全性也很重要，我们在这方面也收到很多反馈，也在这方面投入很多。

总而言之，从投入产出的角度来看，Docker 是一种相对保险的软件分发方式，一个开发者用 Docker 格式发布的软件镜像可以得到多数应用平台的支持。我们正是看到这一点，所以决定让 OpenShift 为 Docker 提供原生支持。此外，再加上我们通过 Kubernetes 提供的资源管理能力和我们从 RHEL 生态圈延续过来的安全实践，OpenShift 完全有能力将这些新兴技术提供为生产环境级别的实施。

InfoQ：Docker 未来的发展是由社区决定的，而社区成员各自都有各自的想法和目的。从红帽的角度，你希望在 Docker 项目和其社区生态中看到哪些变化？

Ashesh Badani：其实开放容器项目就是一个很好的发展方向，核心的运行时和容器格式规范现在都把握在社区手上，这个社区包括了红帽、Docker 公司以及其他的技术公司和用户。此外，这个项目里还有一些 AppC 标准的工作在进行。我们相信更多的、多家公司和组织参与的开放的合作和标准化工作会让企业有更大的信心使用这项技术。

随着容器技术的普及，我们希望在社区里看到更多对于安全、性能和管理性的关注，这些是我们客户所需要的，也是红帽在上游社区的主要工作之一。

InfoQ：你们有一个 Nulecule 项目，我们在 John Mark Walker 的博客上看到他介绍你们为什么要做这个标准化方面的事情。你觉得这件事情对业务的价值体现在哪些地方？

Ashesh Badani：如上所述，标准化和可移植性是企业和 ISV 们对这个技术的信心所在，而这个事情不仅仅是一个文件格式这么简单。公网上成千上万的 Docker 文件，不是随便哪个都可以被部署到企业环境上的。我们需要知道，哪个版本是有供应商在支持的？谁对这个应用的部署生命周期负责？镜像一旦有安全问题，谁会负责打补丁和更新？大规模容器集群的容错要如何做？

建立一个公开透明的社区来制定标准，这是第一步。而上述那些企业的需求，则需要红帽和 OpenShift 来介入实现。我们还有一个 OpenShift Commons 项目，让我们的客户和合作伙伴们协同建立一些最佳实践并共享这些信息。

InfoQ：在你们提供的各个堆栈的解决方案当中，OpenShift 处于最上层。好像现在的订阅模式是，只要订阅了 OpenShift，则你们的基础架构层面的解决方案（比如 Atomic Enterprise 和 OpenStack，或者针对在线版则是 AWS）也被包含在其中？

Ashesh Badani：OpenShift 目前提供三种版本：在线版（Online）、托管版（Dedicated）和企业版（Enterprise）。在线版是多用户、多租户的公有 PaaS 服务，这个是跑在 AWS EC2 上面，只要 AWS 有数据中心的地方就有这个服务。托管版是基于 OpenShift 在线版提供的，相当于在公有 PaaS 平台上专门保留一部分资源给独立客户，他们可以在上面进行应用开发，而我们的运维团队提供管理服务。企业版是我们的私有 PaaS 产品，客户可以决定是要部署在怎样的平台上，可以是物理机、虚拟机、私有云（比如 OpenStack）或者公有云（比如 AWS、Google）。

Atomic Enterprise 的确是包含在 OpenShift 当中的，它提供大规模容器集群编排的能力。我们很多客户希望将 OpenShift 部署在 OpenStack 上面，也有很多客户希望将 OpenShift 部署在自己传统的虚拟化环境上，所以我们对这些需求都进行了覆盖。我们的公有云提供的是比较高层面的抽象，所以基础架构的资源自然也是包含在这个服务当中。

InfoQ：整个订阅模式是如何运作的？OpenShift 的运营重点主要是企业版吗？你们会把在线版当作一个重点的运营对象吗（就好像其他的互联网公司那样）？

Ashesh Badani：从第一天起，OpenShift 企业版就提供了按年订阅、按照 socket-pair 或 core pack 提供增值服务的订阅模式。现在的新版也是差不多的订阅模式，这对于物理机、虚拟机（RHEV/KVM、VMware、微软 Hyper-V）、私有云（OpenStack）和认证的公有云（AWS、Google 等）上面的部署是同样适用的。

对于在线版，我们有银级和铜级的套餐，支持服务也分为不同级别。

OpenShift 托管版目前还处于早期试用阶段。由于它既有私有云的专有性和公有云的运维服务，现在有很多客户都对这个版本感兴趣。客户也可以选择部署一些额外的服务，如通过安装 JBoss 企业版来获取 Java EE 支持，安装 ActiveMQ 以提供消息服务，以及移动应用服务（这个很快会上线）。托管版会有一个应用商场（Marketplace），上面还会提供一些额外的数据库、监控服务以及第三方合作伙伴的服务。

我们对在线版当然是认真的。在线版现在有超过 250 万应用，以成倍的年增长率在增加。上面的应用种类广泛，有移动应用、分析应用和 Web 应用。北卡罗莱纳州的州网站就架设运行在我们的在线版上。还有一些亚洲和拉丁美洲的大型服务提供商想要基于 OpenShift 在线版提供他们的公有 PaaS 云服务。

嘉宾简介

Ashesh Badani，红帽公司副总裁及 OpenShift PaaS 总经理，负责红帽 OpenShift PaaS 业务部门。之前，Badani 曾负责过红帽 JBoss 中间件产品线的管理及市场营销等工作。加入红帽公司之前，Badani 曾在 SUN 公司工作，是整合与应用平台产品方面的产品管理与营销总监。

利用JS在AWS上构建大规模弹性Web应用

作者 谢丽

JAWS是一个高度可扩展的Web应用程序模板，由个人数据库提供商Servant开源。它使用Amazon Web Services提供的新工具（DynamoDB、Lambda、API Gateway、AWS S3）重新定义了如何使用JS（后台Node.js，前端jQuery）构建大规模弹性Web应用程序，其目标是：

- 不使用服务器：永远不需要处理服务器扩展、部署、维护和监控方面的问题；
- 组件隔离：JAWS后台完全由AWS Lambda函数构成。每个函数可以单独开发、更新和配置；
- 无限扩展：由Lambda函数构成的后台支持高并发，很容易实现跨区域冗余；
- 成本尽可能低：Lambda函数只有被调用时才会运行，而只有运行时才需要付费。

JAWS主要包含如下四个部分：

- API：JAWS整个后台都是由Lambda函数构成，组织在api文件夹中。每个API URL指向一个Lambda函数。这样，每个API Route的代码都是完全隔离的，开发者可以针对特定的API URL开发、更新、配置、部署及维护代码。其作用相当于传统MVC结构中的“Controller”。
- Lib：lib是一个npm模块，其中包含了可以在所有Lambda函数中重用的代码，可以认为是传统MVC结构中的“Model”。按照设计，Lambda可以按需requrire它的代码，而不是全部代码，例如，只加载User模型需要的代码：var ModelUser=require('jaws-lib').models.User;。

- CLI：JAWS 提供了一个命令行接口，用于 Lambda 函数的测试和部署。
- Site：网站或客户端应用程序文件夹。为了缩短响应时间，静态资产可以上传到 AWS S3 上。

下一步，Servant 计划在 JAWS 中引入 AWS API Gateway Swagger 导入工具，为现有的 API 函数编写 swagger.json，向 CLI 添加 Swagger 导入命令。在同 Hacker News 网友进行讨论的过程中，项目作者 ac360 指出：

下一步，开发者可以通过 Swagger 在 JSON 中定义 API，然后导入 AWS API Gateway，实现 API 的即时创建 / 更新。这应该会极大地缩短开发时间，简化 JAWS REST API 的构建。

这意味着，你仍然可以进行大规模的版本化发布，而且只需要简单地更新 Swagger 文件中的 JSON。这将使 JAWS 工作流有一个优美的结构，而且非常简单。

有许多网友都认为 JAWS 是一个不错的项目，ahallock 就是其中之一。不过，他希望 JAWS 提供 Lambda 与 RDS 的集成，因为他不想使用 DynamoDB。对此，ac360 回复说：

我一直计划增加 RDS 集成，尤其是 Aurora 支持。计划一直没变！只是需要一些时间。

另外，ac360 指出：

我在 AWS Pop-Up Loft 上增加了一个 JAWS 优化章节……这个页面非常受欢迎。

感兴趣的读者可以安装试用并继续关注。

Netflix的网站优化经验

作者 邵思华

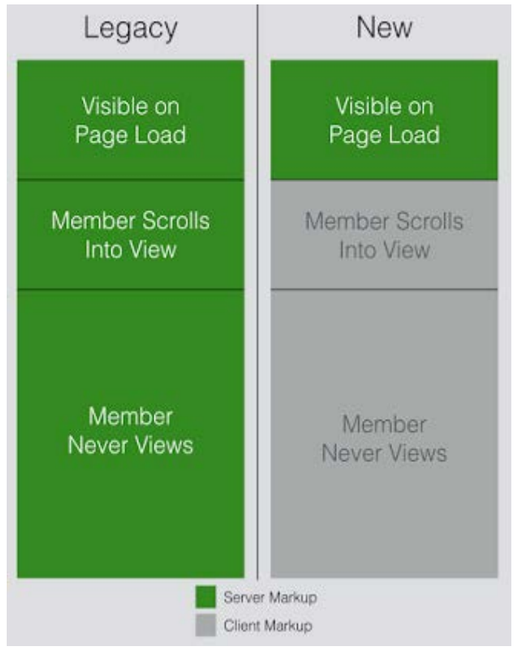
服务端与客户端渲染

Netflix 团队首先要做的一件事是改进他们的整体前端架构。改版前的 netflix.com 网站对于服务端生成 HTML 标记与客户端的增强这两个过程进行了严格的分离，采用这一设计的主要原因在于前后端所使用的编程语言不同。服务端主要使用 Java 的技术栈以生成基本的 HTML 页面，而在浏览器端的工作则主要是通过 jQuery 等 JavaScript 库的使用为服务端生成的 HTML 添加一些客户端的行为。

这种分离式前端架构可以说是教科书一般的标准架构，但 Netflix 团队认为这种方式存在着一些不足之处，因为用户每次都需要等待服务端生成完整的 html 页面结构之后，才能够看到页面显示在浏览器中。这其中有很大一部分内容是用户很少会关注的，但仍然不得不为了加载这些内容延迟而延长页面的渲染时间。

因此，UI 工程团队专门针对这一点进行了全新的设计。改进后的服务端所生成的 html 只包含页面中的一小部分内容，使客户端的视图能够尽快地显示在用户眼前。为了了解用户对此改动的认可度，UI 团队将其设计为一种可配置的架构，可以非常方便地调整服务端所生成的 html 应当包含多少个视图。这种做法的好处很明显：首先是服务端生成的数据减少了，因此处理时间也相应地减少了。其次由于 http 的响应负载也减少了，DOM 的渲染时间也因此加快了速度。当页面完成渲染后，客户端 JavaScript 可以按需加载用户所感兴趣的其余视图。

UI 团体对此总结道，由于服务端与客户端渲染方式得到了更大的灵活性，为他们在这两种方式之间如何取得平衡提供了更多的选择。这一改动最终不仅使页面启动速度加快，同时也保证了平滑的视图转换过程。



通用 JavaScript

UI 团队的另一个目标是实现服务端与客户端代码的通用化，这就迫使他们重新思考整个渲染管道的设计。之前所采用的那种分离式服务端生成与客户端增强的做法已经难以满足他们的需求了，主要问题有以下三点：

- 在两种编程语言之间来回切换是一种负担；
- 如果要对 html 进行改进，那么对于服务端的生成与客户端的增强都有着很强的依赖性；
- 团队更希望通过同一种 API 生成 HTML 标记。

UI 团队最终选择了以 Node.js 与 React.js 实现一种通用 JavaScript 的前端架构，这使他们能够实现在服务端进行渲染，等基本的 html 与 React.js 组件完成初始化之后，再由客户端完成其它部分的渲染。因此，无论渲染过程是在哪里发生的，应用程序都能够得到相同的输出结果，服务端与客户端的代码也没有了严格的区分，它们全部是按照通用 JavaScript 的方式设计的。也正是这种共通的渲染逻辑，让 UI 团队意识到只在服务端进行最小化的 HTML 渲染，由客户端完成其余部分加载这种方式的可行性。

减少 JavaScript 负载

具有丰富交互性体验的网站通常需要用户下载大量的 JavaScript 代码，这也一定程度上影响了浏览器的性能。为此，UI 团队在重构过程中将各种依赖转换为较小的模块，并只为当前访问者输出相应的 JavaScript。关于如何实现这一过程的具体设计，来自 Netflix 的高级前端工程师 Alex Liu 专门在一篇文章中记录了具体的设计过程。

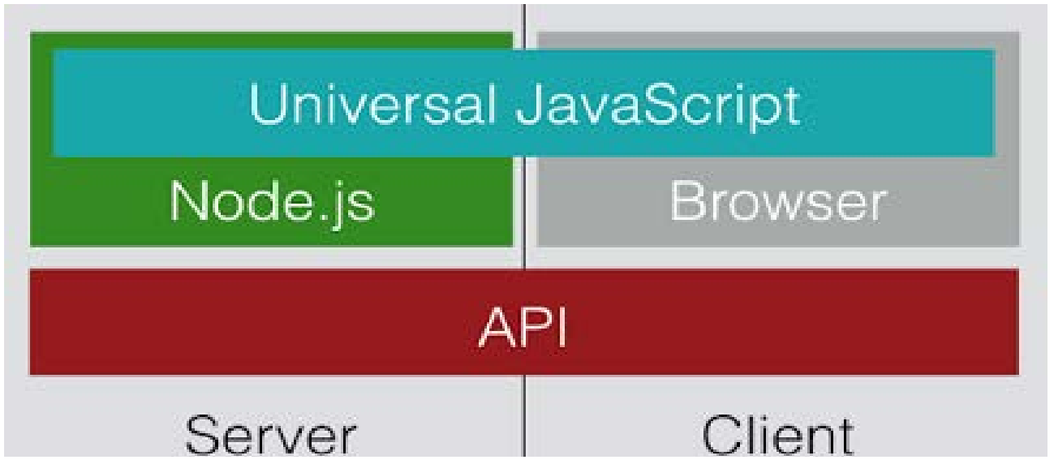
经过重构之后，老版本设计中的各种大型依赖已经不复存在，它们被替换为一些全新的、更高效的库。其直接结果就是输出的 JavaScript 负载减少了许多，用户开始浏览时不再需要加载大量的 JavaScript 代码。而 UI 团队并不满足于当前的成果，他们还将不断地对 JavaScript 的负载进行改进。

页面可交互时间

为了对重构后的效果进行测试，以更好地理解它对用户所产生的影响，UI 团队对于页面的可交互时间（Time to Interactive - tti）这项指标进行了专门的监控。

可交互时间是指从页面刚刚启动到用户能够与 UI 进行交互的这一段时间间隔，这里并不需要完整地加载整个页面，只需要用户能够通过输入设备与 UI 之间进行交互即可。

UI 团队建议使用由 W3C 定义的 Navigation Timing API，在能够支持的浏览器上收集访问者的数据，并进行统计分析。



总结

在 Netflix 团队看来，高性能不是一种可有可无的目标，而是设计优秀的用户体验过程中必不可少的一环。团队将继续寻求业界的最佳实践，以实现更好的用户体验。在接下来的一段时间内，Netflix 将研究一些新兴的 web 标准，例如服务器线程（Service Workers）、ASM.js 以及 WebAssembly 等等，看看这些技术能否帮助他们的网站性能更上一层楼。

更多与 Netflix 相关的文章

腾讯游戏是如何使用 Docker 的

作者 郭蕾

腾讯第一季度的财报显示，腾讯游戏的收入占腾讯总营收的 59.4%，很显然，腾讯游戏已经是腾讯最赚钱的部门，当然，腾讯也是国内最大的游戏发行商。游戏行业相对于其他行业来说特点非常明显，一是需要同时运营多款游戏，二是游戏业务的生命周期长短不一。不管是从数量还是周期来看，游戏行业特殊的业务都为技术团队提出了更高的要求。腾讯游戏从 2014 年下半年开始就在生产环境中使用 Docker，并取得了不错的成果。目前《我叫 MT2》等多款重量级游戏都跑在容器中，且整体运行良好。在 8 月 28 日的 CNUTCon 全球容器技术大会上，腾讯游戏的高级工程师尹烨将会介绍腾讯游戏业务使用 Docker 的进展及收益，并从内核、网络、存储、运营等方面深入探讨腾讯游戏在实践过程中遇到的问题及解决方案，最后还会复盘反思 Docker 对于游戏业务的价值。本文是会前 InfoQ 记者对尹烨的采访。

InfoQ：腾讯游戏是什么时候开始使用 Docker 的？能介绍下目前的一些应用情况吗？

尹烨：我们是从 2014 年 6 月份开始接触 Docker，那时 Docker 在国内才刚刚开始兴起，了解的人还很少。Docker 让容器的管理变得非常简单，再加上创造性的分层镜像的技术，给人眼前一亮。我们希望通过 Docker，构建腾讯游戏内部的容器平台，一方面通过容器提高资源利用率，另一方面通过镜像分发技术标准化、统一化应用部署流程。

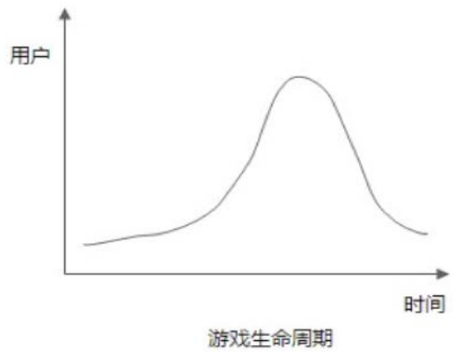
经过半年的调研、各种测试、系统设计和开发，14 年底，整个系统开始上线试运行。但是对于一项全新技术的应用，大家都很谨慎，因为很多游戏业务的在线玩家很多，我们的压力很大。第一个接入我们 Docker 平台的是腾讯的一款游戏，叫《QQ 宠物企鹅》。这款游戏的架构在容灾方面设计得很好，前端可平行扩展，所以就选它作为试金石了。跑了几个星期，运行正常，然后开始慢慢扩展到其它业务。

现在，我们的 Docker 平台上已经跑了数十款端游、页游和手游的各种游戏应用，特别是新上的手游业务，其中，我们代理的一款重量级手游《我叫 MT2》，一个业务就使用了 700 多个容器。现在整个平台总共有 700 多台物理机，3000 多个 Docker 容器。这个数字在业界并不算多，我们自己也没有刻意去追求数量，相对数量，我们更愿意以稳为先。目前，整个平台运行了大半年，整体运行良好。

InfoQ：与其它行业相比，游戏行业有什么特殊性？ Docker 在这样的业务中有什么样的优势？它可以发挥怎么样的价值？

尹烨：相比于其它业务，一是游戏业务更加复杂多样，有端游、手游和页游，有的是分区分服，有的是全区全服；另外，我们又分自研和代理游戏，更加增加了复杂性。这也给业务的运维部署带来了许多不便，尽管我们内部有很成熟的部署平台。而 Docker 统一的镜像分发方式，可以标准化程序的分发部署，解放运维的生产力。特别是代理游戏，如果都以 image 方式交付，可以极大提高效率。

另一方面，一般来说，游戏业务的生命周期长短不一，这需要弹性的资源管理和交付。所以，腾讯游戏很早就开始使用 XEN/KVM 等虚拟机技。相比于虚拟机，容器更加轻量，效率更高，资源的交付和销毁更快。另外，还可以通过修改 cgroup 参数，在线调整容器的资源配额，更加灵活弹性。



游戏生命周期

InfoQ：腾讯游戏的 Docker 应用场景是怎么样的？

尹烨： Docker 开创性的提出了“Build、Ship、Run”的哲学。总的来看，现在主要有两种使用 Docker 的方式。一是基于 Docker 搭建 CI/CD 平台，重点放在 Build 和 Ship 上面，一般用于开发、测试环境；另外就是将 Docker 容器当作轻量级虚拟机，更加关注 Run 的问题，大规模的用于生产环境。个人认为，这两种方式无所谓谁好谁坏，长远来看，二者会渐渐趋于统一。

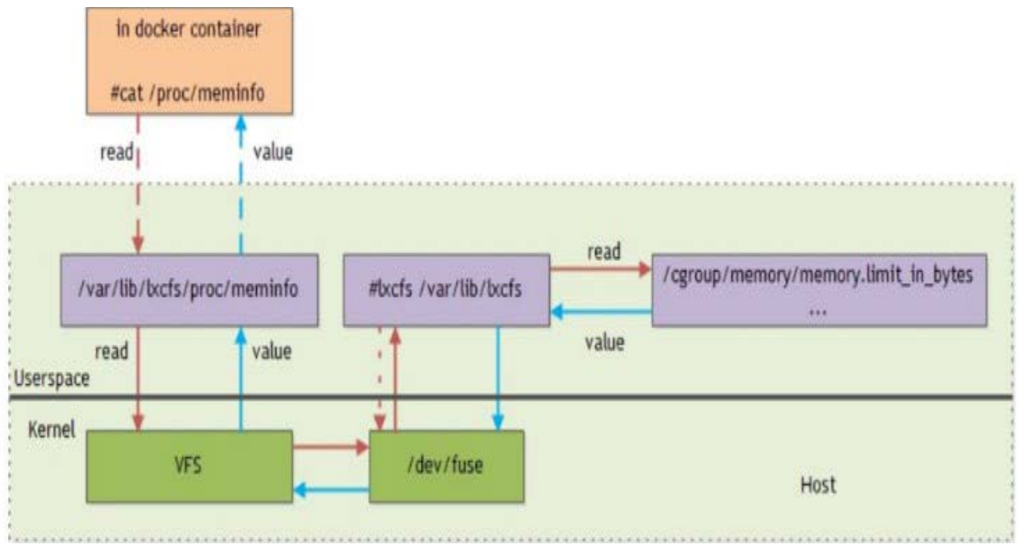
腾讯内部有很成熟的开发、部署工具和流程，我们作为平台支撑部门，去推动业务开发改变原来的开发模式需要的较长的时间周期。所以，我们现在更多的是将 Docker 容器作为轻量级的虚拟机来使用，我们在 Run 上面花了很多时间和精力。同时，我们也在探索通过 Image 方式去标准化业务部署流程。但是，我们不太会去做 CI/CD 的事情，我们更关注提供一个高效的容器资源调度管理平台，然后以 API 的方式对外，提供给开发和运维同学使用，比如，与互娱的蓝鲸平台打通。

InfoQ：能否介绍下你们线上的 Docker 集群所使用的技术栈？

尹烨：我们的使用 Docker 的初衷是替代虚拟机，所以我们直接将 Docker 跑在物理机上。我们使用 Docker 面临的第一个问题就是操作系统内核的问题。腾讯内部一般使用自己的 OS（tlinux）团队维护的内核，这个内核历史比较久，不支持 Docker，我们就选择了 CentOS 6.5 的内核。实际上，由于 CentOS 的内核不像 Ubuntu，演进得很慢，CentOS 6.5 的内核也很老，但基本能把 Docker 跑起来。但在实际使用过程中遇到了一些内核方面问题，现在 tlinux 团队已经提供了 3.10.x 内核的支持，我们也在逐渐往 3.10.x 的内核迁移。

第二个问题就是容器集群的管理调度，那时候虽然出现一些专门针对 Docker 的容器管理工具，比如 Fig、Shipyard 等，但这些工具无法胜任生产环境大规模集群管理调度。刚好那时 Google 开源了 Kubernetes，它是 Borg 的开源版本实现。源于对 Google 的崇敬，我们研究了一下源码，基于 0.4 版本，针对我们的环境做了一些定制修改，用于我们的集群管理调度。现在我们最大的单个 Kubernetes 集群 700 多台物理机、将近 3000 个容器，生产一个容器只需几秒钟。

容器的监控问题也花了我们很多精力。监控、告警是运营系统最核心的功能之一，腾讯内部有一套很成熟的监控告警平台，而且开发运维同学已经习惯这套平台，如果我们针对 Docker 容器再开发一个监控告警平台，会花费很多精力，而且没有太大的意义。所以，我们尽量去兼容公司现有的监控告警平台。每个容器内部会运行一个代理，从 /proc 下面获取 CPU、内存、IO 的信息，然后上报公司的监控告警平台。但是，默认情况下，容器内部的 proc 显示的是 Host 信息，我们需要用 Host 上 cgroup 中的统计信息来覆盖容器内部的部分 proc 信息。我们基于开源的 lxcfs，做了一些改造实现了这个需求。



这些解决方案都是基于开源系统来实现的，当然，我们也会把我们自己觉得有意义的修改回馈给社区，我们给 Docker、Kubernetes 和 lxcfs 等开源项目贡献了一些patch。融入社区，与社区共同发展，这是一件很有意义的事情。

InfoQ：在我的印象里，游戏还是相对较保守的行业。你们在内部推进 Docker 过程中遇到过哪些阻力？是如何解决的？

尹烨：首先，我们会在 Docker 新功能接入与业务原有习惯之间做好平衡，尽量降低业务从原来的物理机或虚拟机切换 Docker 的门槛，现阶段业务接入我们的 Docker 平台几乎是“零门槛”。正如前面所述，我们的 Docker 平台上已经跑了数十款端游、页游和手游。

其次，Docker 相对原有的开发部署方式变化很大，与其它新事物一样，让大家全部适应这种方式是需要一些时间，但 Docker 本身的特性是能够在游戏运营的各环节中带来诸多便利，我们的业务主观上对新技术的应用还是欢迎的，双方共同配合，共同挖掘 Docker 在游戏运营的中的优势，所以 Docker 推广目前没有遇到太大的阻力。

InfoQ：应用过程中，哪些问题是 Docker 目前无法解决的？你们的解决方案是怎样的？

尹烨：我们在实践过程中遇到了很多问题，有些是内核的问题，也有些是 Docker 本身的问题。由于篇幅问题，这里仅举一些比较大的问题。详细的分享留到 8 月底的容器技术大会吧。

我们遇到的第一个大的挑战就是网络的问题，Docker 默认使用 NAT 方式，这种方式性能很差，而且容器的 IP 对外不可见。一般来说，游戏业务对网络实时性和性能要求较高，NAT 这种方式性能损失太大，根本不能用于实际业务中。另外，腾讯内部的很多程序对 IP 都是很敏感的，比如只有特定的 IP 才能拉取用户的资料，如果这些服务没有独立的 IP，是无法正常运行的。

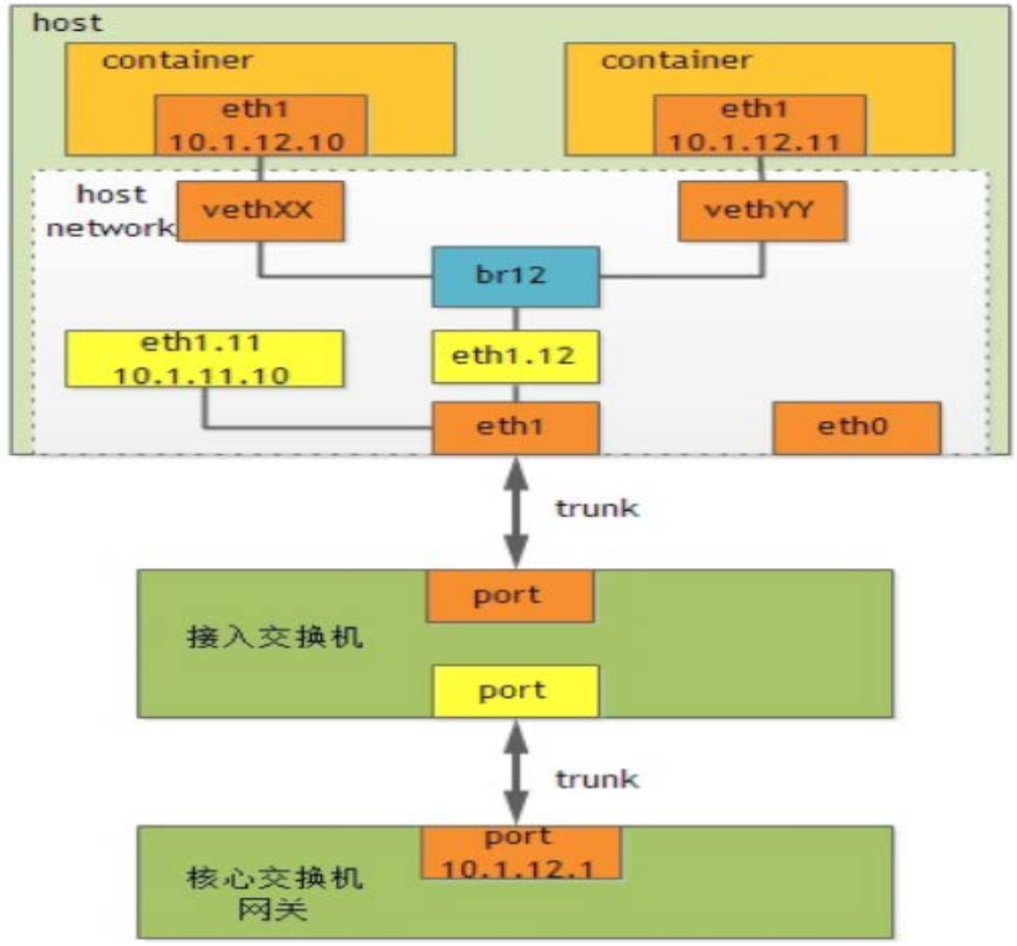
我们针对腾讯的大二层网络环境做了较大的调整，整体架构如图所示。

整体架构比较简单，与原来虚拟机的网络架构一致。每个容器都有一个独立可以路由的 IP，网络性能大幅提高，基本能满足业务的需求。而且，每个容器都可以携带 IP 在同一个核心交换机下任意漂移，业务通过 IP 漂移可以做很多有意义的事情，比如 Host 故障快速恢复等。另外，我们针对一些对网络性能要求高的应用，直接使用 SR-IOV，可以完全达到物理机的网络性能。

容器相对于虚拟机，有很多优势，但是也有一些劣势，比如安全性、隔离性等。由于我们是内部业务，所以安全性的问题不是那么突出，但隔离性的问题还是给我们带来了很多麻烦。性能监控我们通过 lxcfs 基本解决，但是还是有一些问题无法解决，比如内核参数的问题。很多内核参数没有实现 namespace 隔离，CentOS 6.5 的内核下，在容器内部修改，会影响整个 Host，我们只能在 Host 上设置一个最优的值，然后告诉业务，让他们不要在容器内部修改内核参数。3.10.x 的内核要好一些，对于没有实现 namespace 的参数，在容器内部不可见，这可以防止业务私自修改内核参数，避免对别的业务造成影响。但是，有些业务对内核参数有特殊要求，我们只能让业务选择虚拟机。

再举个例子，一些业务会将程序进行 CPU 绑定，这可以避免 CPU 切换带来的性能损失，由于程序无法获取 cgroup 对容器 cpuset 的限制，绑定会失败，这需要业务程序先获取容器的 cpuset，但还是给业务带来了不便。

再比如，现在 cgroup 对 buffer IO 并不能进行 throttle 限制，不过内核社区已经在解决了，但离生产环境使用，可能还需要些时间。



还有 Docker daemon 进程升级的问题，现有 Docker 实现下，Docker daemon 一退出，所有容器都会停止，这会给大大限制 Docker 本身的升级。但最近社区已经在讨论这个问题，希望这个问题在不久的将来得到解决。

InfoQ：使用过程中有哪些坑？你们有做哪些重点改进？

尹烨：上面已经讨论了很多我们在使用 Docker 遇到的问题，当然还有更多，这里不再一一论述。这里再举个简单的例子吧。Docker daemon 进程在退出时，会给所有的容器的 init 进程发送 SIGTERM 信号，但是如果容器的 init 进程忽略了 SIGTERM 信号，就会导致 daemon 进程一直等待，不会结束。我们修改了 Docker，发送 SIGTERM 信号后，等待一段时间，如果容器还是没有退出，就继续发送 SIGKILL 信号，让容器强制退出。我们将这个修改提交到了 Docker 社区，因为一些原因并没有被接受，不过已经有另外的 PR 解决了。

InfoQ：这么长时间的应用，有做过复盘吗？未来有什么计划？

尹烨：的确，相对于 Docker、Kubernetes 的发展速度，大半年的时间已经很长了。我

们使用的Docker版本还是基于1.3.2的，Kubernetes的版本是基本0.4的，已经很老了，但是基本上是满足我们现在的要求，而且系统运行也很稳定，所以，短时间内不会做大的调整。但是，我们也看到最近Docker发布一些非常有意思的变化，比如network plugin、volume plugin，还实现了默认的overlay network。Plugin机制会让Docker更加开放，生态圈也会发展得更快。但总体来说，这些新特性还处于experimental阶段，等这些特性成熟稳定之后，我们会考虑切换我们的Docker版本。

但是，我们也不会坐着等待，也会尝试一些我们需要的东西。比如，最近我们正在实现将Docker与Ceph结合，我们已经实现了Ceph rbd graph driver，将Docker的rootfs跑在了Ceph存储集群上面，结合IP漂移，可以实现更快的故障恢复。我们也将其实现提交给了Docker社区，因为一些技术原因，并没有被社区接受。目前看来，以plugin graph driver的方式提供会更好，但是现在Docker还不支持plugin graph driver，不过Docker社区正在实现，相信不久就会支持。我们已经在GitHub上创建一个开源的plugin graph driver项目。

另外，对于Kubernetes的应用，我们也还没有完全发挥其优势。Kubernetes的负责人Brendan Burns曾说过，“Make writing BigTable a CS 101 Exercise”，Kubernetes有很多非常超前的设计思想，当然，也会改变业务原有的一些软件架构，真正应用到实际业务中还需要一些时间。

另外，我们也会继续探索与业务开发、运维的结合方式，进一步发挥Docker的优势，提高我们的运营效率，更好的支撑游戏业务的发展。

嘉宾简介

尹烨，腾讯互娱运营部高级工程师。2011年毕业加入腾讯，现在主要负责Docker等相关技术在腾讯游戏业务的实践。主要关注Linux内核、虚拟化、存储等领域，给libcontainer/kubernetes等开源项目贡献过代码。

软件组织正在快速地实施云技术，但迁移始终是一个无法回避的挑战。哪些部分是需要你密切留意的？哪些应用程序更适合于进行迁移？如何对应用程序进行重构以适用于云端？经历了这一转变的先行者为我们留下了什么启示？在这一系列文章中，你将从那些在帮助企业成功地迁移至云环境方面富有经验的专家那里获得实用的建议。这一领域应得到高度关注，我们希望你也能够参与这方面的讨论。

你的客户或用户真的在意你的应用程序是运行在Tier 1的数据中心，还是运行在公有云上吗？当然不会，但他们确实会定期关注在使用你的服务时所经历的体验。客户期望的服务是快速并且可靠的，它不会因为访问量的突然上升而降低响应度，而是始终保持优良的表现。他们也期望你能够更新你的服务交付产品，让它与最新的技术发展保持一致。
为了理解这一点，让我们用业务方面的说法来进行一个类比，可以将以上需求解释为：

你现有的应用程序（例如一个医疗应用）的基础设施是否足够敏捷与灵活，能够快速地推出一个新的特性？比如在可穿戴设备（例如智能手表）上用亮灯及语音提醒患者到了服用药品的时间。

在本文中，我们将探索将现有的应用迁移到云端所产生的各种益处的细节，使你在进行决策时做到心中有数。

为什么要进行迁移？

- 在首尔举办的Big Bang团体“2015世界巡演”的在线售票过程中，热情的歌迷不仅很快抢完了门票，还使得服务器一度崩溃。

- Flipkart 在举行一次有历以来力度最大的促销活动时，网站发生崩溃。

灵活的能力

以上这种情况并不罕见，其结果是媒体反而加大了对这一事件的关注。有趣的是，如果一个初创公司的应用首次遇到这种因为突发性流量而产生崩溃的情形，这种体验并不那么糟糕，反倒是一次宝贵的经验，因为崩溃是难以避免的。

那么，你的应用是否能够经受住访问量突然空前增长的考验呢？

为了回答这个问题，你可能先要询问一下你的基础设施提供商：为了应对这种突发性的增长，他们能够在多少时间内增加硬件能力。你的提供商是否能够做到接近即时地（在几分钟内）设置（或解除）资源？

与传统的服务提供商不同，云服务在设计时就提供了内置的工具，以建立快速的灵活性，它能够确保以下能力：

1. 一旦由于突发性流量造成访问量达到峰值，能够设置额外的资源（例如虚拟机 VM）以维持对用户的响应能力。
2. 一旦突发性流量回复正常，能够立即解除这些额外的资源（将虚拟机释放，返回给云服务）。这能替你节省很多钱！
3. 你的应用能够按需立即访问无限量的资源。

简单的总结一下。在任何一个时间点上，根据你的负载情况变化，云服务总是能够为你提供适当的设置能力。设置的资源既不会过多也不会过少，并且能够快速、自动地完成。

对于传统的基础设施提供商来说，要对负载的增或减作出反应，通常需要几个小时（实际上很可能要几天）。因为他们需要手工介入这一过程，才能够实现这种可响应性的机制。

注：请不要混淆了**灵活性**与**可伸缩性**。只要在云环境中的基础设施上部署应用，就能够隐式地获得这种灵活性，而应用程序的可伸缩性是由它的架构所决定的。如果某个应用程序能够通过往物理机器中加入（纵向扩展 / 垂直伸缩）额外的能力（例如 RAM 或 CPU），或是为资源池中加入更多的物理机器（横向扩展 / 水平伸缩），而因此能够承受更多的负载，我们就说这个应用是可伸缩的。

很显然，仅仅将应用迁移到云服务并不总是能够确保它的可伸缩性。如果某个应用程序的数据库是基于 Oracle 或 SQL Server 的标准版本，它不支持分区（实现可伸缩架构的前提条件之一），在这种情况下要实现大规模化，可能要对整体功能进行大规模的重新设计。而 NoSQL 数据库就实现了高度的可伸缩性。与之类似，你的应用程序或许托管在一个多核的服务器上，但如果你的应用没有实现多线程应用，那么这点优势也没有什么实际用处。这也是 Google 为什么要推出 Go 这门新语言的原因，就是为了让并发（多线程）编程更简单。

不过，你必须理解一点，并非所有的应用都需要这种灵活性能力。举例来说，如果你的业务规模很小，只是在一个内部的 HR（人力资源）管理应用中为 50 人左右的员工提供服务，并且员工数目不太可能会有爆发式的增长，那么为了灵活性而将整个应用迁移至云端就没什么必要了。

降低成本

成本的降低包括两个方面。

1. 无资本支出

由于你不需要购买 IT 方面的基础设施，因此也没有资本支出方面的成本了。云提供商不需要你做出任何长期的承诺。

2. 现用现付

云服务的基本特征之一就是按使用量付费。服务计划是根据预配置的 VM 的运行小时数，以及带宽、数据传输及存储量的大小而决定的。请确保你进行了适当的配置与设置，只为你所需的能力付费。

这种特征与传统的数据中心完全相反，后者在大多数情况下，你都需要额外设置超过正常范围的机器，虽然你只在一些极端的情况下才会用到这些能力，但仍然要为它们买单！

敏捷性 —— 更快地进入市场

作为一名企业家，你必须具备长远的眼光，看准时机快速地决定开发应用程序的某个新特性、提交代码、将其发布到市场，让你的客户为此感到震撼（也包括你的竞争对手！）。云服务提供商能够为你带来自动化的基础设施以及快速的转变能力，与之相比，传统服务提供商在基础设施的申请、订单处理、获取以及安装上显得非常官僚。

你是否注意到一点，在进入 2015 之后，电子商务应用逐渐从移动优先转变为应用优先策略？假设这种策略正适合你的目标，那么你是否能够做到足够敏捷，以快速地响应这种技术上的转变呢？

专注于业务

与 IT 相关的诸多琐事都能够在云服务中实现自动化，以传统的方式管理这些琐事无疑是对你的宝贵时间与精力的一种浪费，它对于你的核心业务没有产生任何直接贡献。

请保持你远离那些使你的注意力分散的东西，将你的劳动力保持在更高价值的活动上并取得一致。别忘记 —— 时间就是金钱。

更高的可用性、可靠性和性能

除了云服务的服务水平协议（SLA），还有什么服务能够实现三个 9 甚至四个 9（99.99%）的可用性呢？虽然传统的数据中心或许会声称他们也能够达到相同的能力，但其中的大多数并不具备某种良好的评估与监控服务，而这是对他们的 SLA 能力进行审计所不可或缺的。

为了提高可用性与可靠性，云服务能够实现跨多个可用性区域（AWS、Rackspace），还有公有云提供商实现的联合网络（federated network），以及多虚拟机（multi-hypervisor）技术，这些都是为了实现 100% 的可用性而设计的特性。

对许多现有的应用程序来说，他们的用户往往来自于某个特定的区域。在迁移到云端之后，由于选择了合适的地理区域的云提供商，应用的网络延迟也降低了。

容灾性（DR）

正如他们所说：

> 如果某件事有可能会变得更糟，那么它就一定会变糟。
>
> —— 墨菲定律的某种变体

你的应用程序是否能够经受住灾难的考验呢？例如网络与电源故障、火灾、地震、暴雨、洪水，或是保存你的服务器资源的大厦遭受了严重的物理破坏呢？

容灾计划（DRP）需要细致的计划以及架构设计，

它能够意识到故障的发生并进行快速响应，以缓解对业务造成的损失。DR 是一个复杂的命题，因为大多数企业都没有准备好 DRP，或者说他们所准备的 DRP 会由于整个计划中的某些小缺陷而注定会失败，而这些缺陷是由于缺乏这方面的演练而产生的。只有这些罕见的灾难发生之后，他们才会意识到这些缺陷。

对于业务的连续性来说，成熟的 DRP 是十分关键的。云环境在设计上是十分灵活的，你需要选择正确的工具集与配置，让灾难对你的业务产生的影响降至最低。

云环境是安全的

100% 的安全性只能是一种幻觉。如果你必须在现有的选择中进行决策，那么云服务的安全性不会低于任何一种现有的系统。云服务的提供商以他们的创新性而闻名，在任何一方面，他们所实现的物理及逻辑安全实践都要胜于单一的本地数据中心的运维。许多云提供商如今都已经获得 ISO、PCI DSS、欧盟模式条款（EU Model Clauses）及其它安全组织的认证。

此外，并非所有应用程序都需要银行级别的安全性，对吧？如果你的应用中真的包含了高度机密的数据，或是必须符合某种特定的安全与隐私条规（例如 HIPPAA 或 HITECH），那么你可以选择混合式的云服务（推荐医疗应用使用这种方式）。

这些不时出现的关于云服务安全性的问题，更多的是一种 FUD（恐惧、不确定性和怀疑），而并非事实。

其它一些无法忽视的益处

以上所提到的这些益处是当你迁移到云环境中可以立即获得的好处，除此之外还有一些无形的益处存在。公有云服务是无污染的，因为云资源池也是云环境的根本特征之一。

云服务提高了协作能力，并实现高效的文档控制，使你能够在任何地方开展高效的办公。虽然在迁移过程中需要你对应用程序的各方面进行或多或少的重新设计，但这也为你带来了一个良机，可以实现某些尖端的技术，例如使用 Docker（一种基于虚拟化技术的容器，它已得到了空前的应用）实现更简便的部署。此外，随着物联网在 2015 走向舞台的中央，你可能需要在现在的应用中整合微服务架构，以应对在不久的将来客户可能会产生的请求。

怎样的应用程序适合迁移至云服务？

根据经验来看，在过去七年间所开发的多数应用程序，包括 n 层或流行的 3 层结构 web 应用、批处理应用和后端服务或许比较适合进行迁移，而更早的应用程序或许需要投入更多的精力进行改造。这一点完全取决于现有应用程序的架构。

有多种因素可能会导致你现有的应用程序与云迁移过程不兼容（或者可能需要投入巨大的精力进行改造）：

1. 紧耦合的架构，硬编码的配置。
2. 依赖于某种云服务所不支持的数据库。
3. 需要某些特殊的硬件功能，例如基于硬件的加密、大型主机等等。
4. 依赖于多种第三方应用。
5. 许可问题。

以下这份表格可以作为一份参考，或许能够帮助你进行快速地决策。

封装式应用 Email、协作及生产力应用	**医疗应用**	**遗留应用**
周期性应用 工资单处理、纳税申报、学校招生、某些博彩应用、会议管理等	**严守合规性** 包括 HIPAA - HITECH，美欧安全港，欧洲通用数据保护条例（GDPR）等	**数据库支持** Microsoft SQL Server 2008 R2 之前的版本、MySQL 5.1 之前的版本，及 Oracle 11g 之前版本的数据不适合进行云迁移，因为需要投入额外的精力将它们首先升级到所支持的版本 **有限的，或是无技术支持的** 拿 MS Azure 举个例子，微软对于 SQL Server 2205 之前的版本的技术支持非常有限，或是完全不提供支持。Azure VM 镜像（模板）只包括 SQL Server 2008 R2 之后的版本
可预见发生访问量激增的应用 黑色星期五或购物季时的电子商务应用、皇室的婚礼、世界杯网站、订票应用	**银行应用** 符合 PCI DSS 规范，安全方面的顾虑，对于大型主机的依赖	**许可问题** 如果你现有的许可符合许可证移动性的条款，例如 BYOL（自有许可），就能够将它转到云服务中。对于大多数遗留应用来说，这一选择并不适用
无法预见访问量激增的应用 由于被权威性的网站所提及导致访问量突然上升	**具有地理位置限制的应用** 大多数云数据中心位于北美或西欧。在设计上，云环境中的数据没有边界限制，除非有某些限制的存在。如果你的国家对于你的应用中的数据合规性有某些特殊的需求，在迁移之前要仔细考虑这一点	**ERP 系统** 存在架构瓶颈，例如紧耦合、硬编码的配置、对第三方应用的依赖，在某些情况下会用到特殊的基于硬件的加密功能
初创应用 / 在线游戏 访问量的增减取决于你的业务是成功还是失败		
大数据（分析）应用		
需要超级计算能力 超高带宽、增强型网络、以及非常高的计算能力 —— 如 HPC （高性能计算）		
微服务		
概念型原型、开发及 QA 环境		
备份、归档及存储		

行动胜于语言

这下这份表格中的组织已经将现有的应用迁移到云环境中，表格中也提及了他们在迁移完成后所获的益处。

序号	组织	挑战	迁移至	益处
1	**Animoto** 美国的视频制作以及图片幻灯片制作公司	现有的应用是在自有的服务器上运行的。 在 2008 年 4 月，通过 Facebook 上的应用达到了用户高峰，在 3 天之年的注册人数达到了 75 万	AWS	在峰值时，一小时内约有 2 万 5 千人会使用 Animoto！为了在三天之内达到这种能力，他们的应用从 50 个 EC2 实例扩展到最多 3500 个 EC2 的实例。 注：EC2 实例即虚拟服务器
2	**redBus** 印度的在线巴士订票服务	在一个传统的数据中心运行它的处理工作，它的基础设施无法有效地应对处理量的起伏，这影响了他们的生产力。 并且服务器的采购及配置非常耗时	AWS	1. 总体成本降低了大约 30-40% 2. 延迟降低了 （大约 4 倍）。由于延迟的降低，redBus 的访问量大约上涨了 3 倍
3	**Expedia** 美国处于领先地位的旅游公司	必须在物理上接近客户的地方运行他们的 Expedia Suggest 服务（ESS），以实现快速的可响应服务，使网络延迟降至最低	AWS	平均网络延迟从 700 毫秒降至 50 毫秒以下
4	**小米** 中国的高端智能手机制造商	需要一种解决方案能够为世界各国的人群提供可靠性以及令人满意的连接速度，并且在公司成长时能够无缝地拓展他们的服务	AWS	1. 下载速度快了 30% 2. 服务交付周期变短了，并且快速地推出了应用下载中心 3. 更好的访问速度，减少了前期的投入
5	**Aucor** 芬兰的 Web 设计公司	他们的私有虚拟服务器在小规模时运行情况良好，但随着项目数量的不断上升，所需的服务器数目过多，还需要聘用全职的系统管理员	GAE	能够每秒处理超过 7 万个请求，并且让用户完全感受不到延迟
6	**Khan Academy** 美国的非盈利性教育组织	最初他们维护着一个网站用于他们不断增加的视频库，这一结构已经经过了许多年，但随着访问量的增长，该平台遇到了各种限制	GAE	1. 每个月能够支持 380 万独立请求，每个上学日还要提供 150 万个实践问题并处理回答 2. 保存着 2000 个以上的视频片段，并且还在继续增长 3. 该系统能够轻易地处理用量的增长

结论

随着数以万计的应用迁移至云服务并开始利用它的能力，事实已经证明了云服务的实用性，并且它已经实现了整个产业的预期。在全球已经有很大一部分公司以某种形式在使用云服务了，无论这些公司的地理位置在哪里以及他们的专业领域是什么。

回头看看 2008 年，当时云计算的“期望夸大”（指虽然宣传力度很高，但无法继续推动）正处于巅峰期。某个价值数百亿美元的 IT 公司的 CEO 曾表示：

可以说，计算机产业是唯一一个比女性时尚更受时尚所驱动的产业了。也许我确实是个傻瓜，但是我真的不知道其它人都在说些什么。这到底是什么？

这一家 IT 巨擘由于忽视了这一趋势，目前已经被视为这方面的落后者，它的财富也在不断下滑。但近几年他们也参与到云计算的竞争中，并试图努力追赶领先者的脚步！因此，重要的是为你的转变进行计划，而不要等到为时已晚再开始行动。

作者简介

Basant Singh 是一位软件开发者，具有 12 年应用程序及数据库架构设计的经验。在过去五年间，他通过他的编程技术帮助初创公司超越最小可行产品这一阶段。目前，他正充满热情地投入到他的下一个项目中，它使用了 Go 编程语言与 Docker 技术。在他职业生涯的前期受雇于某个跨国企业时，他为全球各处的财富 500 强上的客户创建了多个应用程序。
他的博客是 Techno-Pulse 和 GolangPro。Twitter 帐号是 @SinghBasant。

软件组织正在快速地实施云技术，但迁移始终是一个无法回避的挑战。哪些部分是需要你密切留意的？哪些应用程序更适合于进行迁移？如何对应用程序进行重构以适用于云端？经历了这一转变的先行者为我们留下了什么启示？在这一系列文章中，你将从那些在帮助企业成功地迁移至云环境方面富有经验的专家那里获得实用的建议。这一领域应得到高度关注，我们希望你也能够参与这方面的讨论。

亚马逊正式发布关系型数据库 Amazon Aurora

在去年的 AWS re:Invent 大会上，亚马逊宣布了 Amazon Aurora。Aurora 是一个关系型数据库，其设计目标是提供高性能和高可用性（99.99%），并且存储可以轻松高效地扩展到 64TB。近日，AWS 首席传道士 Jeff Barr 宣布 Amazon Aurora 正式发布，但目前只有美国东部（北弗吉尼亚）、美国西部（俄勒冈） 和欧洲（爱尔兰）等三个地区的用户可以使用。目前 Amazon Aurora 已经可以用于生产环境。

Solaris 支 持 Docker，Oracle 加入 OCI 阵营

2015 年 7 月 31 日，Oracle 宣布旗下的操作系统 Solaris 将开始支持 Docker，以增加对开源的 Linux 容器技术的支持。Oracle 并不是第一家宣布在自家操作系统中支持 Docker 技术的公司。去年 10 月 Windows 宣布在 Windows Server 2016 中将支持容器技术。像 Oracle 一样，微软是在自己的技术上支持 Docker，而不是借用 Linux 技术。在去年 12 月，IBM 也宣布通过 Bluemix 提供一个基于 Docker 的容器服务。Oracle、微软和 IBM 目前都是开放容器项目（Open Container Initiative ， OCI）的成员，OCI 目前一共包括 30 多家成员公司。

阿里云峰会在京召开，首次披露云生态路线图

2015 年 7 月 22 日，首届阿里云分享日 × 云栖大会北京峰会召开。会上，阿里云集中发布了 11 款新产品、50 多个行业解决方案，首次披露云计算生态路线图全貌。新品包括读写性能超群的 SSD 云盘、可一键搭建混合云的 VPC 服务、基于 PostgreSQL 并兼容 Oracle 的云数据

库、支持 Redis 的 KV 数据库以及应用于基因测序与渲染等行业的批量计算等。

迅雷“星域”重新定义 CDN

2015 年 6 月 25 日，迅雷携手小米公司在北京正式发布“星域 CDN”系统。新一代 CDN 采用并实现了无限节点布局，这是国内首家提出并实现无限节点的 CDN 服务商。发布会上陈磊坦言，CDN 是最没有黏性的云产品，价格不是决定因素，质量才是关键。“互联网上几乎所有的内容，迅雷都有索引。”

红帽发布企业版 Linux OpenStack 7

近日，红帽公司宣布红帽企业 Linux OpenStack 平台 7 全面上市。该版本推出了一个全新的部署和管理工具，旨在简化安装过程和日常管理任务，同时支持系统实时编排更新和版本升级。红帽企业 Linux OpenStack 平台最初于 2013 年发布，此次发布是该平台的第五版，已被全球多个领域（如政府、金融、电信和教育）的客户成功部署。

华为发布企业云战略

2015 年 7 月 30 日，华为在北京召开发布会，正式宣布面向中国市场的企业云服务。发布会上，华为轮值 CEO 徐直军表示：“云服务模式正在成为企业 IT 的新模式，这已经成为产业界的共识。为了满足企业市场客户与合作伙伴在网络时代的新需要，同时也为了提供更贴近用户的云计算解决方案，华为决定推出企业云服务，这是华为 ICT 产品和解决方案的自然延伸，是实现华为在 ICT 领域投资回报的新商业模式。”

青云推出 MongoDB 集群服务

2015 年 8 月 3 日，基础云服务商青云 QingCloud 宣布，正式推出基于 MongoDB 的集群服务。青云 QingCloud CTO 甘泉（Reno Gan）表示，MongoDB 的发布进一步完善了 QingCloud 的数据库和缓存服务。未来，QingCloud 还会推出 SQL Server 服务、事务型数据库集群服务，以及 Cassandra、HBase 等数据库相关服务，并与即将上线的 Spark、Hadoop、Storm 共同为用户提供完整的、一站式的数据存储和分析平台。

容联云通讯发布 IM Plus 重构即时通讯

2015 年 8 月 5 日，即时通讯云服务商容联云通讯在京举行 IM 产品新版发布会，IM Plus 在原有即时通讯功能的基础上，加入了音 / 视频聊天以及会议、会议管理功能。据容联 IM 产品总监张靖宇介绍：容联 IM Plus 主要有功能全、技术强、集成块三大特性。目前平台上已经累计有超过 10 万名开发者，8 万个应用，其中包含 15,000 个以上的付费应用。

InfoQ

郭蕾 / Gary

InfoQ主编

负责InfoQ网站内容的输入和输出
关注云计算、容器、开源等领域
信奉见城彻的那句话:
偏执、冒险、狂妄的人终是英雄。
我不是英雄，
但我会努力成为英雄。

我在InfoQ约吗？

简历请投至：ada@infoq.com

InfoQ 中文站 2015迷你书

云生态专刊
2015年03期

《云生态专刊》是InfoQ为大家推出的一个新产品，目标是“打造中国最优质的云生态媒体”。

开源启示录
第一季

开源软件的未来在于建立一个良性循环，以参与促进繁荣，以繁荣促进参与。在这里，我们为大家呈现本期迷你书，在揭示些许开源软件规律的之外，更希望看到有更多人和企业参与到开源软件中来。

顶尖技术团队访谈录
第二季

《中国顶尖技术团队访谈录》·第二季挑选的九个团队虽然都来自互联网企业，却是风格各异。希望通过这样的记录，能够让一家家品牌背后的技术人员形象更加鲜活，让更多人感受到他们的可爱与坚持。

架构师 月刊

《架构师》月刊是由InfoQ中文站针对高级技术开发和管理人员所推出的电子刊物。