

云生态

Cloud Ecosystem

特刊

第7期
Vol.07

技术
热点

对话
大咖

观点
&
趋势

生态圈
新闻

InfoQ

打造中国最优质的云生态媒体



热点回顾

- Netflix已经全部迁移到云端
- Spotify联姻Google，迁移入云



遗留系统迁移到云端的3步走方略

本文讲从大的方向上，讲述了我们在合作的过程中，将老的系统向云上迁移的经验。

关于云迁移的经验总结

本文总结了云迁移项目的关键要点。

只需6步，教你把服务迁移到云端

本文总结了Ghost(Pro)的基础设施从专用服务器迁移到DigitalOcean Droplets的过程。

案例实践

- 让AWS Lambda运行在多个云平台
- 借助Spinnaker简化业务全局云部署
- 解密Spotify基础设施和数据的云迁移
- 微吼：业务入云是一条不归路

云生态特刊

- 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



FACILITATING THE STUDY AND COMMUNICATION OF TECHNICIANS

让技术人员学习和交流更简单

EEO EXTRA GEEKS' ORGANIZATION
NETWORKS

高端技术人员
学习型社交网络



InfoQ ueue

专注中高端技术
人员的社区媒体



StuQ ueue

实践驱动的
IT职业学习和服务平台



GIT GEEKBANG
INTERNATIONAL
TRAINING
极客邦培训

一线专家驱动的
企业培训服务





InfoQ 活动专区全新上线

更多活动·更多选择

MORE ACTIVITIES MORE OPTIONS

一站获取所有活动信息

Geekbang
极客邦科技

全球领先的技术人学习和交流平台

InfoQ | EGO | StuQ | GIT
技术媒体 职业社交 在线教育 企业培训



扫描二维码
前往专区

卷首语 | 魏星 Mikey 云计算专栏主编



入云是一条不归路

有人称 2015 年是中国云计算的元年，其依据是国内云计算的落地进入加速阶段。随着最近一个月 Netflix、Spotify 这样的海外大企业迁入云端的细节更多地披露出来，我们认为，“入云”这个话题是时候拿出来说说了。

业内有专家称，中美云计算发展的差距大概为两年。在物联网蓬勃发展的时侯，更多的设备连接到云端，更多的数据传输至云端，而企业业务形态势必因此发生改变——这一点传统行业的厂商感受更为切肤，其业务迁移的意愿和需求也更为强烈。与此相应的是，国内云计算市场的竞争显然十分激烈，运营商、系统集成商、硬件设备厂商、互联网巨头、解决方案供应商、IDC 乃至 CDN 厂商纷纷杀入战作一团，国际巨头更是丝毫没有放松。但，怎么帮助企业顺利地把业务从原有的 IT 系统中迁移到云端，大家恐怕要众说纷纭。

也许样板的力量正是一个契机，作为观察者，应该走进厂商和用户那里，听听双方对这个问题的看法。走访中，微吼的林总说了这样一句话，“业务入云是一条不归路”。这句话让我印象颇为深刻，是为题记。

Netflix

作者 张英洁

已经全部迁移到云端



我们把 Netflix 迁移到云端的旅程开始于 2008 年 8 月，那时我们正在经历一场主数据库错误，长达 3 天无法向会员发放 DVD。正是在那个时候，我们意识到必须从纵向扩展的单点系统失败模式——比如我们数据中心的关系数据库——转向高度可靠的横向扩展的云端分布式系统。我们选择了 AWS 云服务作为我们的云服务提供商，因为它为我们提供了最大的规模和最广阔的服务与功能。大部分我们的系统在 2015 年前就已经被迁移到了云

端，其中包括全部面向客户的服务。从那时起，我们就开始为计费基础架构和客户与员工数据管理寻求安全且持久的云服务。我们很高兴地在 2016 年的 1 月份宣布，在长达 7 年的勤奋努力之后，我们终于完成了云迁移，并且关闭了流媒体服务使用的剩余数据中心！

迁往云端给 Netflix 带来了许多益处。我们现在拥有了 8 倍于 2008 年的流媒体会员数量，而且他们的参与度也更高了，整体看来在 8 年里这个



图 1

数据增长了 3 个数量级。（见图 1）

Netflix 产品本身继续迅速发展，开启了很多新的消耗资源的功能，并且依赖于不断增长的数据量。对我们的数据中心来说，支撑如此迅猛的增长极端困难；我们根本不可能让服务器足够快。云的灵活性允许我们在几分钟内增加数千个虚拟服务器和 PB 级存储，从而使上述扩张成为可能。2016 年 1 月 6 日，Netflix 将它的服务扩展到 130 多个新国家，成为一个真正意义上的全球互联网电视网络。充分利用遍布全球的多个 AWS 云区域，使我们能够动态地变换并扩张我们的全球基础架构容量，从而为 Netflix 世界各地的会员创造出更好、更舒适

的视频播放体验。

在所有的可扩展计算和存储需求方面，我们都依靠云计算。我们的业务逻辑、分布式数据库和大数据处理 / 分析、推荐、转码和数以百计构成 Netflix 应用的其他功能，都是如此。视频是通过 Netflix 开放连接来传送的，这个开放连接是一个全球分布式内容分发网络，可以有效地将我们的数据推送到会员的设备上。

云计算还使我们显著提高了自己的服务可用性。以前，我们的数据中心有非常多的运行中断，尤其是在云迁移的早期，我们重点解决了一些不可避免的云端问题。在整体可用性方

面，我们的服务稳定提升，已经接近了期望中的目标——99.99%的服务正常运行时。在任何大规模分布式系统中失败都是不可避免的，即使基于云的系统也一样。然而，云计算却可以使一个基于不稳定的冗余组件建立高可用的服务。在架构设计中采取冗余和优雅降级的原则，加上通过 Simian Army 严控生产环境，即使云基础设施和我们自己的系统都出现问题，仍然不会影响用户的体验。

成本削减并不是我们迁移到云端的主要原因。然而，我们在云方面的花费最终只是那些数据中心花费的一小部分——这是一个令人欣慰的额外好处。这可能还是由于云的灵活性，使我们能够不断优化实例类型的配置，并几乎瞬时地增加和收缩占用的资源，而不用维持大量的备用资源。我们还能从规模经济中受益，而规模经济只在庞大的云生态系统中才成为可能。

鉴于云的益处如此明显，那为什么我们花了整整 7 年才完成迁移呢？事实上，迁往云端是件非常艰苦的工作，在前进的道路上我们不得不做出许多艰难的选择。毋庸置疑，最简便的迁往云端的方法就是复制整个系统直接放到 AWS 上。但是这样做，你最终就把数据中心所有的问题和局限一并带去了云端。作为替代，我们选择

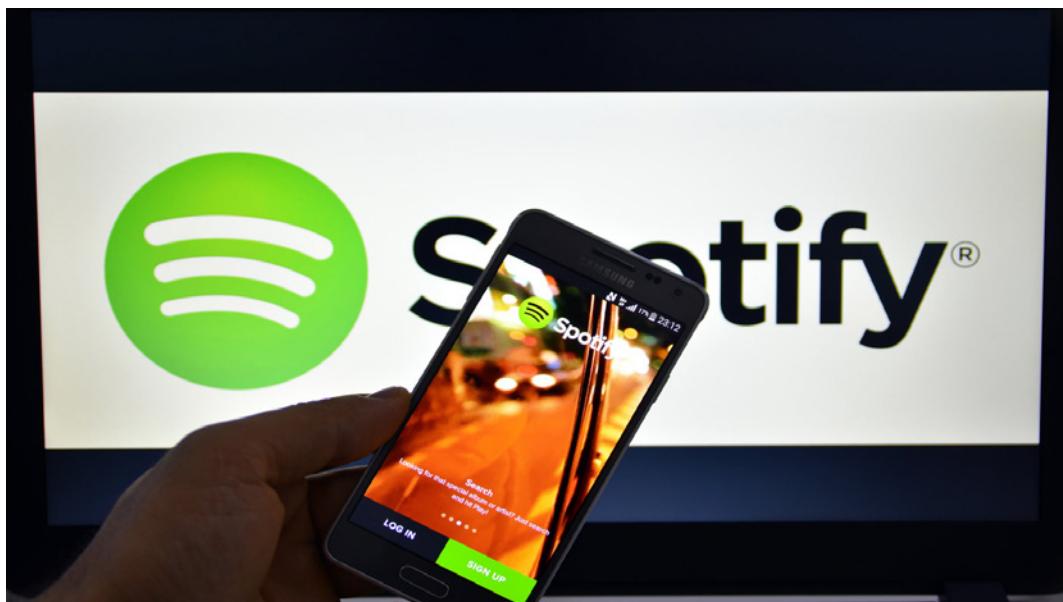
了云原生的方法，纵向重建所有的服务，从根本上改变我们运营公司的方式。在结构上，我们从一个单一庞大的 APP 转向了数以百计的微服务，转向非规格化，转向我们使用 NoSQL 数据库的数据模型。预算审批、集中发布的协调工作和多个星期的硬件置配循环，让位于持续的交付，让位于工程团队制作使用自助服务工具，并在松散连接的 DevOps 环境中独立决策，这些都加速了创新。我们必须建立许多新的系统，学习新的技能。把 Netflix 转变成一个云原生的公司需要时间和努力，但这也把我们置于更有利的位置，让我们继续增长并成长为全球化电视网络。

在过去的几年中，Netflix 的流媒体技术已经走过了漫长的道路。最终不被我们先前面临的限制所束缚，这感觉真是太棒了。对我们业界的很多人来说，云计算仍然是相当新的事物，还有许多问题要去解答，许多难题要去克服。通过 Netflix Open Source 这样的开源举措，我们希望继续和伟大的科技头脑合作，共同应对所有这些挑战。

Spotify 联姻 Google

作者 陈星璐

迁移入云



过去 10 年间，AWS 的业务超越各大竞争对手，夺取了大量市场份额。但在本周二（2016 年 2 月 23 日），流媒体音乐服务 Spotify 宣布计划将后端服务从 AWS 迁移到谷歌云平台。Spotify 工程和基础设施副总裁 Nicholas Harteau 近日在他的博客中写到：“我们很少谈论令人兴奋的技术基础设施——音乐背后的真正力量。但在今天这个特殊的日子里我们宣布，我们正在与谷歌的云计算平台团队合作，为 Spotify 提供无处不在的平台

基础设施。”同时他表示，跟传统的基础设施架构相比，云服务商所提供的存储、计算和网络更低廉、高效和稳定。Spotify 目前在全球有 7,500 万免费会员和 2,000 万付费会员。

今年 1 月，Spotify 正式宣布推出影片内容到 App 上，有人认为激增的数据和流量是促使 Spotify 迁移至谷歌云的原因之一。

Google 宣称，Spotify 资料查询工作将可在 1 至 2 分钟内完成。

Google Cloud Platform 首席销售工程师 Guillaume Leygues 在官方博客中表示，Spotify 已经开始迁移工作，正在部署如 Direct Peering、Cloud VPN 和 Cloud Router 等网络，来迁移 PB 级的数据到谷歌云平台。Spotify 迁移至谷歌云平台分 2 个部分，分别是服务和数据。在服务方面，Spotify 采用微服务架构，分批次迁移至 Google 云储存、云计算等产品；在数据方面，Spotify 将从 Hadoop、MapReduce、Hive 和自建的仪表盘工具迁移到 Google 的 Cloud Pub/Sub、Dataflow、BigQuery 和 Cloud Dataproc 等产品中。

Spotify 之前在自己的数据库中完成数据分析功能，使用该公司自行购买或租赁的设备。云计算则使之可以更加高效地展开扩张或运营，而且成本也低于自行管理私有数据中心。

“Spotify 决定不再自建数据中心。”谷歌云平台首席销售工程师 Guillaume Leygues 说。

Spotify 工程和基础设施副总裁 Nicholas Harteau 表示，谷歌的大数据分析能力发挥了重要作用，“这是谷歌的优势所在，我认为他们将在这方面持续保持优势”。他特别提到：

“谷歌一直是这个领域的思想先驱，从其数据的复杂性和质量上均有体现。

从传统的 Dataproc 批处理，Pub/Sub 的事件传递，到 BigQuery 的几乎不可思议的能力，构建谷歌数据基础设施为我们提供一个最重要的显着优势。”

但也有不同的声音。

德意志银行分析师 Karl Keirstead 本月早些时候指出，谷歌云计算平台仍以较大的差距在所谓的公有云市场位列第三，大幅落后于亚马逊和微软。

根据美国《华尔街日报》报导指出，Spotify 将继续使用亚马逊简易储存服务储存音乐文件，也将持续透过多个内容渠道商，向用户提供音乐服务。谷歌云计算平台服务是否能借此一举拉近与亚马逊、微软之间的差距，或许还存在不少变数。

来自 Hacker News 的一位用户则表示：“我倾向于回避谷歌云计算的主要原因是，它被中国封锁。而 AWS 不是（这也意味着 Heroku 并没有被封锁，因为 Heroku 在 AWS 上运行，所以我使用 AWS IaaS、Heroku PaaS）。如果你打算提供多语种内容，期待有来自另一个世界的观众，确保你的内容不会被封锁这一点的确值得考虑。另一个原因是，我发现谷歌云计算控制台一团糟。我必须同时使用两个版本的控制台来完成不同的任务，在这方面，Heroku 更整洁、更易操作。”

无论如何，美国市场研究公司 Forrester Research 分析师 Dave Bartoletti 将该交易称作“谷歌的一场重大胜利”。

亚马逊对此拒绝置评。

Forrester Research 分析师 Dave Bartoletti 同样也认为 Spotify 选择谷歌主要是看中该公司的 BigQuery 数据分析处理服务。“正是软件开发者服务为其带来了竞争优势，包括分析、大数据处理和信息服务等。”

但是，BigQuery 还是有很多明显的缺陷。当我们谈论大数据时，我们总是把 Variability（数据多样化）当成最重要的一点来讨论，很显然，BigQuery 所支持的数据类型还不够多样，或者说，非常单一。但从大数据的处理能力上来看，其 TB 级数据查询结果秒出的效率的确让人惊叹。

回顾技术世界的大事记，你会更惊讶的是：Spotify 的这一举动，绝非偶然。

早在去年年底（2015 年 11 月），谷歌的第八位员工、云业务主管 Urs Holzle 就表示，谷歌云计算平台业务的营收将在未来 5 年内超过广告业务收入，他们的目标是在 2020 年之前将谷歌转型为一家云公司。他说，云市场与智能手机市场有点相似。在 2007 年推出的 iPhone 创造了智能手机的需

求。尽管 Android 起步相对晚一些，但它现在已经成为全球最流行的操作系统。Holzle 说：“我希望我们能够复制 Android 的成功。”

虽然，当时的业内人士普遍认为谷歌的云业务落后于竞争对手，落后于亚马逊的 AWS 和微软 Azure，这主要是因为它始终未能拉到能够给公司带来巨额收入的大企业客户。但 Holzle 认为这种情况很快就会改变。

实际上，谷歌很快就会宣布一些能够打消业内人士疑虑的消息。3 个月后 Spotify 宣布了与谷歌合作的消息，不禁让人回想起 Holzle 的这番言论，也许那个时候，他们的联姻就已经悄悄开始了。只是亚马逊还被蒙在鼓里——该公司之前曾经将 Spotify 作为重要客户对外宣传，还专门撰写了一份案例研究。

技术和商业的世界里，似乎没有忠诚可言。但，还好有情怀。“在 Spotify，我们痴迷于提供流媒体的体验，那种感觉就好像，把世界上所有的音乐都放在了你的手中。”

关于云迁移的经验总结



作者 Alois Reitbauer 译者 邱广



Alois Reitbauer 是 Ruxit 公司的首席布道师。在他职业生涯的绝大部分时间里，他都在进行监控工具构建以及应用性能微调。他常以演讲嘉宾的身份出席会议，同时他还是一名博客作者、作家和寿司狂人，Alois 近期的工作地点是 Linz、Boston 和 San Francisco。

在近期举办的 Dynatrace Perform 大会上，我与不同类型的技术公司就云迁移这个话题进行了广泛的交流。所接触的专家从 SaaS 公司到电子商务公司和云服务提供商。公司规模大小不一，既有初创型规模的，也有大至巴西最大的电子商务公司之一，以及建立 SaaS 业务的公司。尽管这些公司之间颇为不同，但他们对云迁移项目

的关键要点有着高度一致的看法。本文涵盖了前 5 大共识。

总会发生迫使你转向云的事情

讨论小组都认同，实施云迁移不是平白无故做出的决定。所有的与会者都是因为遇上了不可抗拒的事件，才被迫转向云的。对 B2W 来说，这个

不可抗拒的事件就是，在一次严重的生产事故后，他们决定朝微服务的方向去重构整个环境。当基于交易量来动态扩展独立组件成为云战略的一个关键部分时，迁移到云端也就成为了他们战略的关键组成。对行业领先的 BAR 考试服务提供商 BARBRI 而言，驱使他们决定向云迁移的关键动力来源于数据中心的重建需求。

向云迁移的关键约束在于成本

当问及迁移到云的动机时，第一个想到的就是降低成本。所有与会者都同意，降低运行成本是云迁移的核心驱动力。对更小的初创公司来说，IaaS 的弹性模型使它们能够对所需基础设施进行及时的投资。对成熟的公司而言，走向云迁移的关键事件是对重建数据中心的再投资。一旦你面临一笔大的投资，就会对战略问题三思而后行。“对我来说，当下在数据中心上的花费是我的首要成本。” BARBRI 的 IT 主管 Mark Kaplan 说。需要重点指出的是，低成本的收益需要一段时间才能显现出来。在过渡期，公司同时运行云和非云设施。“仍然需要为接下来的两年时间做好成本节约计划，以保障项目成功”，Kaplan 指出。

APM 工具的相关工作应在 CIO 议程上

实施云迁移同时带来了一件有

意思的事情，APM 工具的相关工作不能缺席 CIO 议程。通过云服务的弹性和灵活性，公司能够更直接和快速地进行成本优化。BARBRI 的 Greg Birdwell 指出，“我们使用 APM 工具不光是为了监控基础设施的健康度。如果我发现有服务消耗的 CPU 或内存资源较其它服务少很多，我就能切换到更廉价的实例上去。这能立即为公司节约成本。” BARBRI 的 Mark Kaplan 说，他们使用 Ruxit 来精确地缕清环境的依赖关系和资源需求，这些都是完成迁移的基础。全球各地的 CIO 们盯着 APM 工具，根据监控数据计算成本收益，这样的景象也许我们还要再过一段时间才能看到，但事情的发展正在朝这个方向演进。

敏捷压倒成本收益

从讨论小组那获得的一个令人非常惊讶的事实是，尽管降低成本是云迁移的一大核心动力，但所有的与会者都认同他们现在花在云基础设施上的钱更多了。“如果运行在一个传统的数据中心上，对我们会更便宜。” Stilnest 的 Michael Aigner 说。“但我们还是把宝全部押在云上，因为我们获得了比节省一小笔钱更宝贵的东西：敏捷。”所有的小组成员都认为云服务通过支持新功能的敏捷开发模式，从而帮助产品获得更短的上市时间，这点是客户们非常欣赏的。

云迁移影响了文化

与会者认为，云迁移不是一个技术或经济游戏规则颠覆者，实际上它广泛地影响到了公司的文化。一旦开始运行在云上，你就已经转向 DevOps 了。“我们所有的基础设施现在都在代码里” Stilnest 的 Michael Aigner 说道。“我期望我们的运维团队能更像开发人员，像开发人员那样工作”，B2W 公司的 Alexander Ramos 说。开发人员持续不断地投入到生产问题的解决中，根据 Ruxit 的 Anita Engleider 的说法，以 DevOps 牵头常常促使开发团队成为应用运维的一部分。Engleider 认为“不管怎样，开发人员都比我更了解他们自己的代码，所以他们应当为监控负责”。B2W 公司的 Alexander 把这点推向了一个极端：“我给开发人员发了寻呼机，他们和其他人一样都随时待命。这改变了很多东西。”

遴选云服务提供商是一大挑战

讨论小组认为，遴选出合适的云服务提供商是一个重大的挑战。云服务提供商的选择依赖于你对基础设施的需求以及云服务提供商自身的灵活度。同时，当评估价格点时，就不仅仅只需考虑基础设施状况了。“如果

你想要办件事情，就会希望电话那端能有人指望得上。可靠可信任的合作伙伴对云迁移的每一步都很关键。” CenturyLink Cloud 的 Bob Stolzberg 说。

成功的迁移建立在严谨规划的基础上

CenturyLink 的 Bob Stolzberg 指出，管理层的赞助支持和端到端的规划对成功交付非常关键。Bob Stolzberg 经历过了非常多的迁移项目，他的建议相当清晰：“制定一个执行计划，争取成功完成，但不要追求完美”。Stolzberg 说，“你还需要制定回滚策略。如果事情搞砸了，就会想要重回安全节点。”

云迁移不代表迁移到公有云上

对许多公司来说，云迁移意味着迁移到公有云上去，事实上往公有云迁移只是其中的一个选项。尤其是当监管方面的要求非常重要时，公有云就力不从心了。Avocado 咨询公司的 Romain Bigeard 指出，这类情况公司的出路是投奔私有云。他认为“利用私有云基础设施也能够获得云环境带来的许多好处”。对这类公司，关键是构建一个可编程的基础设施，进而

提高在软件交付过程中的敏捷度。

做出正确的行动

对会谈参与方做的一个非正式的调查显示了来宾们云迁移战略里头更深层次的内容。大多数决定先从迁移新应用到云端开始。一旦积累到一些经验了，然后开始迁移已有的前置应用到云端。要么一次性把所有的都迁移过去，要么一开始只迁移某些应用服务到云端。

基础设施即服务是迁移最常见的原因

尽管云服务提供商还提供了类似容器即服务（Containers-as-a-service）和 PaaS 的选项，来宾们仍然更青睐基础设施即服务。此外还有许多额外的云服务可以优化应用交付。

理解依赖关系是最大的挑战，紧随其后的是规模估算

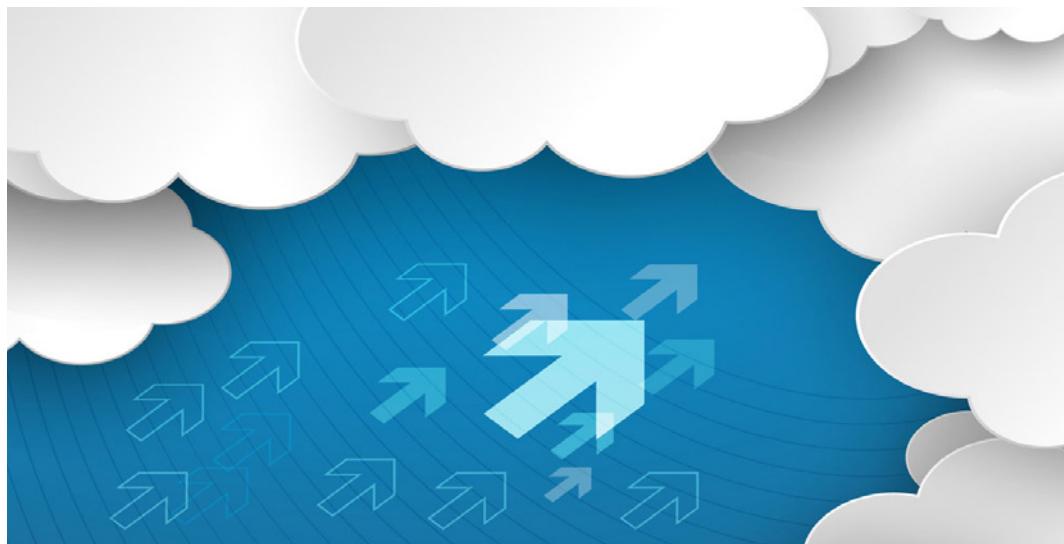
云迁移最大的挑战是理解现有基础设施内的依赖关系，这点得到了广泛的认同。“有时我想知道为什么这两个应用之间存在会话。但找不到说明文档，所以我不得不甩开 Ruxit 上的所有数据” B2W 的 Alexander 说。排在挑战榜单第二位的是规模估算。

不同规模的云实例的价格差异非常明显，因此从节约成本的角度来看，关键在于选用正确的实例类型。传统上，公司为安全起见会购买足够大的机器。而在云上，公司只想买最小可能的基础设施，有需要时再进行扩展。尤其是在变动场景下，目前数据中心服务器的规模常常是过度配置了。正如前述所提到的，基础设施上的花费直接即时，节约成本是关键。

云监控也不尽相同

一旦迁移到云端上了，很多事情就变了。所有这些全新的云服务成为应用的一部分的同时，也意味着需要使用工具进行原生监控和管理应用。挑战榜下一个也是云环境的扩展。成本节约背后所有的理论要求只运行所需的，在纸面上也许很好做到，但是如果除了基于资源消耗的扩展，工具对其它方面只提供了非常有限的功能，这时自动扩展环境显得很困难。成本节约最终会是成果显著的，但在一开始你需要把注意力放在确保迁移努力确有成效上面，而不是让其影响了性能和用户体验。等所有事情都正常运行了，才在扩容和缩容这两个方向上进行容量规划，并经常性地这样操作。在一个基于价格模型的消费世界里，时间能造就其中的财务差异。

遗留系统迁移到云端的3步走方略



作者 杨栋

背景

在互联网大行其道唯快不破的今天，毫不夸张的说，对市场的响应速度甚至会决定一家企业的命运。我们的客户（房地产垂直搜索平台）就是这么一家互联网公司，为了缩短开发周期，减少系统投放市场的时间，我们将现有的基于传统数据中心的基础设施迁移到云端，以便获取这种的能力。本文讲从大的方向上，讲述了我们在合作的过程中，一起将老的系统向云上迁移的经验，以及其中的一些

实践。在此之前，我们先来看看他们的现有系统。

现有系统

图 1 是对现有核心系统的一个简单抽象，我们维护了三条业务线（在这里是指以业务为基础的，有独立的产品经理，销售团队，独立结算的团队，下文简称 LoB），每个业务线对应的是不同类别的房产的搜索网站，比如商业和住宅。User 是用户管理模块，该模块能提供用户管理，书签和搜索管理；Location 是位置模块，根据用

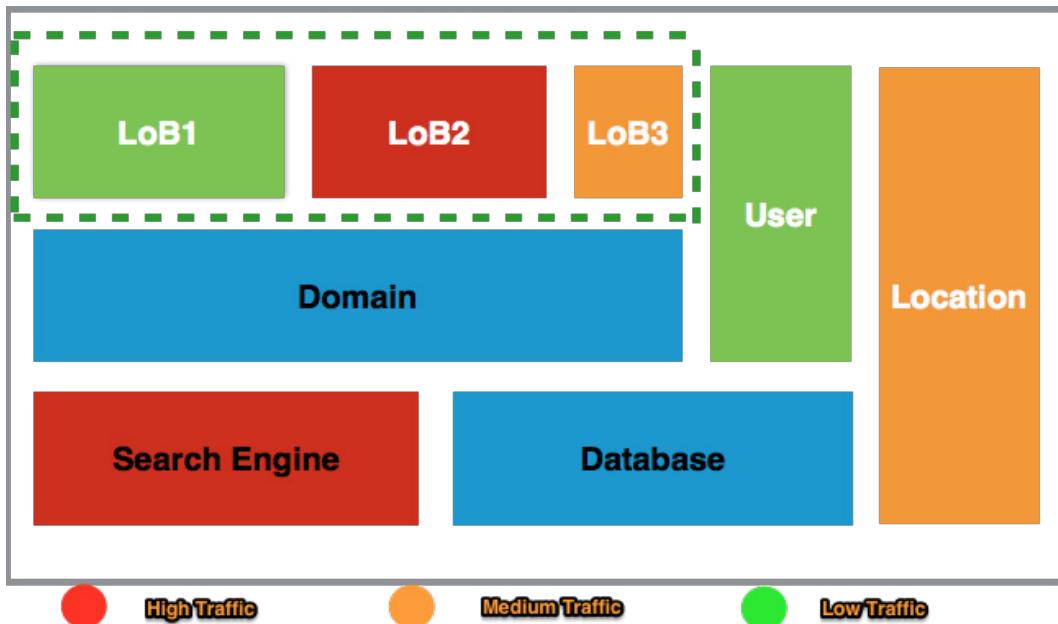


图 1

户的搜索条件给出实际地址和相邻地址; Search Engine用于存储所有信息，并提供搜索功能。可以看出三条 LoB 虽然是不同的网站，但是它们提供的服务是类似的。不仅如此，它们外观相似度也非常之高，除了主题之外几乎没有差别，同样的用户体验，相同的页面布局，还有非常显眼的标识用 来说明他们来自于同一家公司。

从上述原因来看，它们应该集成在一个系统内，但其实不然，尽管这些业务线有如此高的相似度，但是从以下方面它们有着非常大的差别：

- 部署和配置不同

由于不同的房产有不同的搜索

条件，因此在部署的时候，需要对不同的网站进行不同的配置。

- 特性和发布时间不同

不同的房产有不同的目标市场，需求决定了要开发哪些特性，做什么样的市场推广活动，因此每个业务线都有自己的销售和开发团队，并且根据市场变化制定相应的特性开发和发布计划。

- 不同的流量

图一中不同颜色表示不同的流量等级，红色表示流量最大，

黄色表示流量中等，而绿色则表示流量较小。对于不同类型的房产，市场的需求是不同的。

- 目标客户不同，市场定位不同
商业地产的目标客户是那些需要开展商业活动的商家们，而住宅房产的目标客户则是那些想要生活或者为子女提供更好教育机会的普通老百姓。

业务愿景

在了解了现有系统之后，我们再来了解一下他们的业务愿景，因为没有一家公司的 IT 改革是脱离业务驱动的，理解业务愿景有助于更加清晰地理解向云迁移背后的原因。同时云策略的适用场景有一个更加直观的认识。

- 3 年后的年营收翻一番
- 基于 LoB 的运营模式

这意味着每个团队将是完全独立的全功能团队，他们拥有独立的系统，具备独立开发，部署，运维，市场，销售的能力。这就为每个团队提供了非常大的自主权利，对业务的扩张和收缩提供非常好的自适应能力。

- 效率

这里所说的效率主要是指 IT

生产活动中的效率问题，本质上讲他们是一个互联网公司，如何能够提高 IT 系统的效率来支撑业务的发展这是他们所看重的。比如，提高人的效率，开发，上线，测试，运维，线上反馈等等，各个方面的效率问题。

- 全球扩张计划

中国恐怕是全球最活跃的房地产市场之一了，同时对海外房产的投资在中国持续升温，他们没有理由放过这个机会的，不仅如此包括在德国，意大利，中国香港都有他们的身影。

基于以上的种种不同，图一所示的系统架构显然无法满足客户对商业愿景的实现，主要的问题有以下几点：

- 无法独立运营

由于这是一个所有 LoB 都整合在一起的系统，因此业务线之间存在着耦合。试想一下这种场景，LoB1 根据市场的反馈已经完成了对房产中介品牌的增强，并且希望在涨价之前上线，因为这将是一个涨价的合理理由。与此同时 LoB2 正在开发新的页面来满足房产拥有者品牌的需求，但是这个特性只是开发了一半。按照当前的模式

我们必须等待 LoB2 完成特性的开发之后才能一起上线，这样对于 LoB1 就错失了一次涨价的机会，而下一次涨价窗口将是几个月之后。

- 资源利用效率低

通过流量监控我们发现网站的访问量并不是一成不变的，以年为单位，网站流量在圣诞节之前会降低到平时的 50% 左右，圣诞节之后大概又会回升到平均水平 200% 左右，而这样的大流量会持续约 1 周左右的时间。其实这样的情况不难理解，因为在圣诞节期间大家休假，收假之后会迎来一波工作潮。从图一中我们也可以清晰地看到不同组件的不同流量，为了保证整体的响应速度，这个系统始终是以比较高流量的情况部署的，但是由于每个 LoB 无法独立部署，导致资源浪费的情况非常明显。

- 扩张成本高

前文我们提到了全球扩张计划，他们想进入中国市场，于是需要为中国的用户设计一个独立的网站用于提供房源，同时也要将这一扩张作为模式，为将来的向其他国家的房地产

市场扩展做准备，为了实现这一目标，我们需要 IT 基础设施的支撑，能够快速灵活的横向扩张。但是基于现有的数据中心的模式，这将是一个痛苦的过程。

架构愿景

通过对原有 IT 架构的分析，我们发现它是很难支持业务愿景的实现的，因此针对这样的业务愿景，我们勾勒出了能够很好支撑业务愿景的 IT 架构愿景。

- 独立业务线
必须能够按照业务线来独立运营
- 扩展性
容易扩展或伸缩
- 关注
解决从开发到部署的所有问题，开发人员更加集中的关注如何更快的交付特性，而非各种环境问题，部署问题，发布问题
- 效率
提高资源利用率，根据不同的流量情况自适应分配资源。

基于这样的愿景，我们发现云平台能够很好帮助我们实现这样的一种 IT 愿景，如图二描述了系统迁移到云

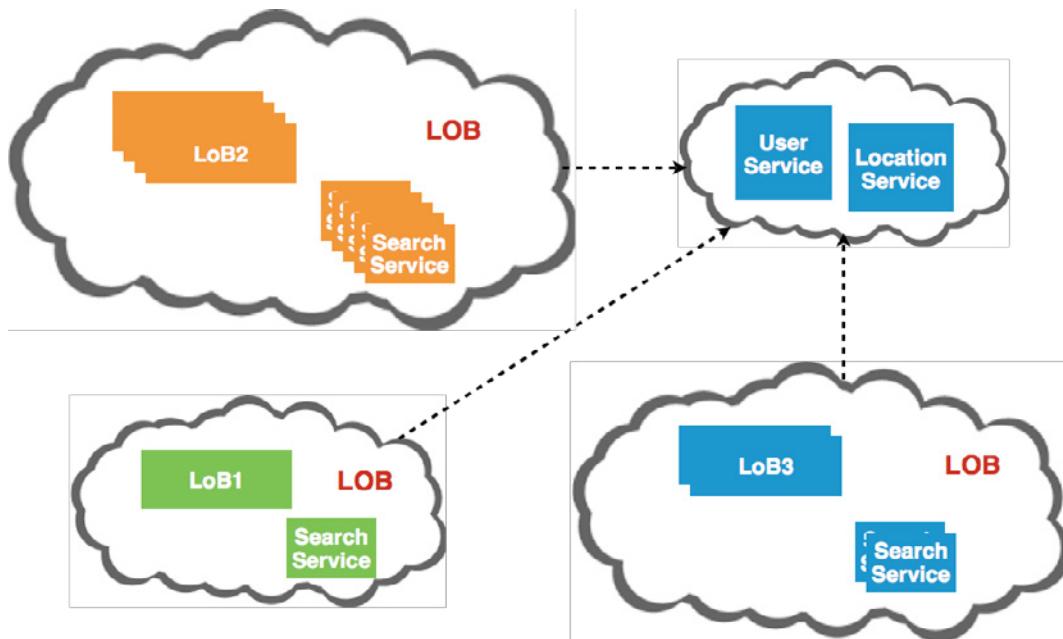


图 2

端之后的架构。

首先，所有的业务线都是独立运营，他们能够自主选择自己何时发布上线，自主选择合适的 SLA 以及资源来适应流量的变化；其次，对于所有业务线共同分享的组件，独立于业务线之外，分开部署和运营，并且它们也能够根据流量的变化调整不同的资源进行适应。

迁移三部曲

当我们知道了现有系统的目标状态之后，下一步就是如何实现这个目标了。

如图 3 所示，在向云端迁移的时，

从现有系统到目标系统的迁移，这个过程不是一蹴而就的，不是一次迁移就能完成的，而是周期性地持续进行，每一次的前进都是相关系统集成的结果。如果我们将目光聚焦在其中某一个迁移的周期内，这个过程大概分为三个阶段：

第一阶段：识别

识别就是要弄清楚迁移什么。

对于原有的集成在一起的系统来说，这一过程与聚合更好相反，是把所有聚合在一起的功能特性分析并拆解，这个活动的目的就是深入理解当

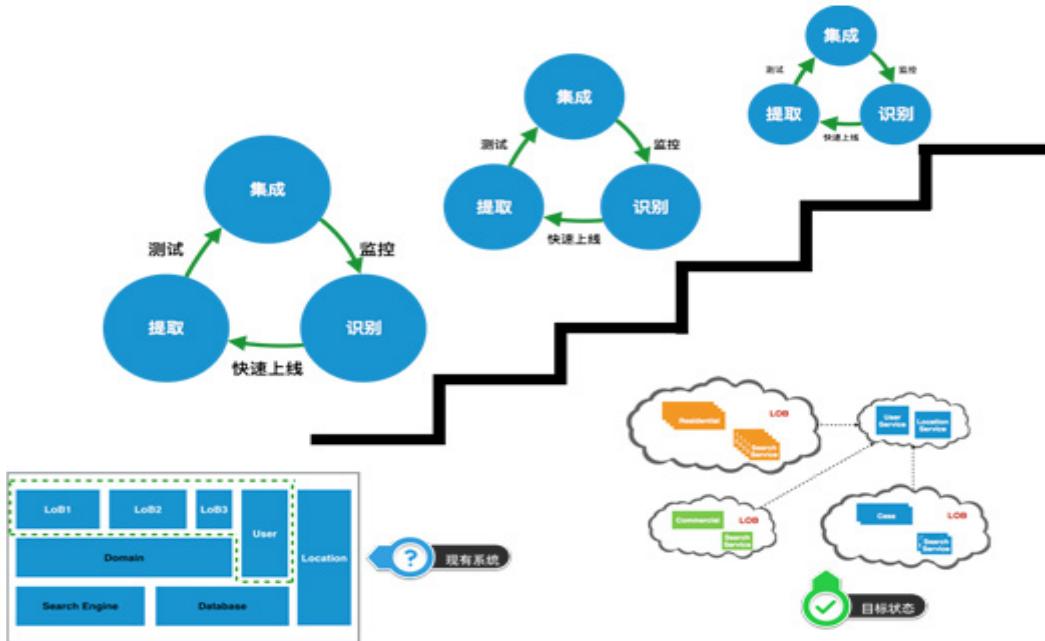


图 3

前系统承担的职责。在弄清楚职责之后，我们就可以更好的识别哪些职责是可以被剥离，拆分，并独立出去的。比如对这个房产搜索网站来说，识别之后发现我们的它的职责主要有：前端展示（桌面，移动），房产搜索，搜索的管理，用户管理，地理位置查询，同时还提供了部分 API。在完成了识别之后，就是我们要选择从哪些职责开始入手迁移。这个过程并不是随机的，而是要根据现有的团队能力，业务目标，综合所决定的。

我的建议是从简单的，相对独立的职责入手，如果同时还能对业务发展有所支撑，那就是再好不过了，因

为难度低，所以能够在迁移初期给团队带来自信和经验，随着迁移经验和自信的积累，那些耦合度高，依赖多的职责也能够轻松的迁移。以该房产搜索平台为例，在迁移初期，我们选择用户管理职责来迁移，一方面是因为它相对简单，相对独立，另一方面则是因为它对我们进行 iOS 开发提供了 API 支撑。

第二阶段：提取

在识别出系统职责并确定要迁移的职责之后，则是提取该职责为独立云服务的阶段了。

如图 4 所示，这一阶段的核心就是将识别出来的职责独立于原系统之外，成为独立的云服务。这个过程有几点需要注意的是：快速创建，它需要快，多快呢？理想状态是创建成功（空服务）后就直接上线（灰度发布），或许这个目标在刚开始迁移的时候比较不容易实现，但是随着迁移自信和经验的积累，它是完全可行的，当然这也是为什么我们要从简单并相对独立的职责做起的另一个原因了；快速部署，我们创建新的服务是从空服务开始的，也就是说除了能在产品环境运行之外，它什么都没有。在这个阶段，我们的目标是将服务提取的两大痛苦阶段：创建和部署，变得简单高效。

第三阶段：集成

集成就是将新的云服务与原系统

进行对接。

如果说前面两个阶段都是在做准备的话，那么这个阶段就是实施迁移的过程了，可以说这是最为重要的阶段，因为它是新老系统交割的一个时期。在这个阶段会有这样一些活动，首先，识别新服务需要哪些对外的接口，这需要对现存系统有比较深入的了解（当然如果接口比较简单，这一步也可以放在提取阶段来实现）；其次，将现有系统中对要迁移职责的依赖切换到新服务上，这个过程有可能是一次完成，也有可能需要持续完成，主要取决于职责的独立程度以及现有系统依赖管理的复杂程度。最后，将该职责在原系统中移除。以用户管理职责为例，这一阶段就是将现有系统与用户服务集成，并将其该职责从系统中移除。

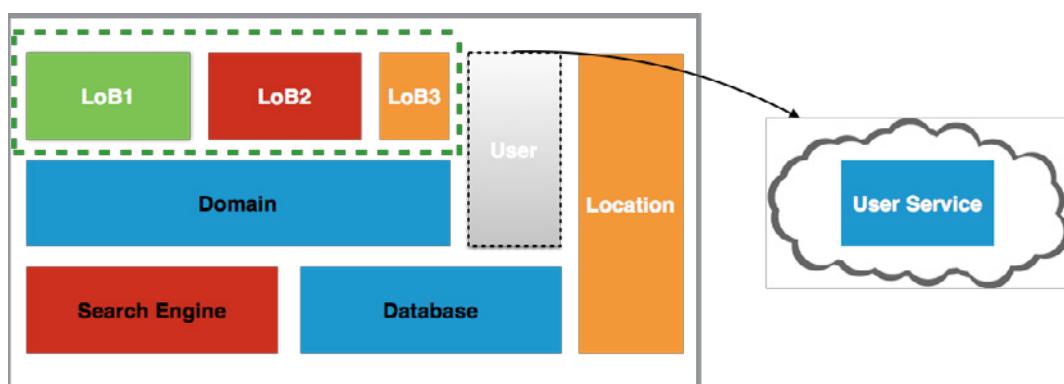


图 4

迁移技巧

理解完这三个阶段之后，不难看出，每两个阶段之间的转换是否高效，流畅，无障碍，对整个的系统向云端迁移都起着至关重要的作用。在长时间的迁移经验的积累下，我们发现以下的一些技巧能够帮助我们在整个迁移阶段中，平滑过渡。如图五所示，当识别出迁移职责后，Stencil + DevOps 能够帮助我们快速创建和部署云服务，在原系统与云服务对接时，可以由测试来驱动，对接之后，监控能为我们本次迁移周期提供很好的反馈，以便开启下一个迁移周期。下面

我们同样以该网站为例，来介绍一下不同阶段之间转换的技巧：（见图 5）

技巧一：快速上线：Stencil + devOps

前文提到，这一过程一定要高效，也就是要迅速地创建并部署新的服务，只有这样做我们才能讲这一过程常态化，如何才能做到呢？我们采用了 Stencil+devOps。

Stencil 就是一个服务模版，它能帮助我们快速生成一个空的服务，包括遵循组织规范的目录结构，标准的监控配置和接口，初始化的构建脚本等，使用 Stencil 的好处就是能够快速创建符合组织标准化的

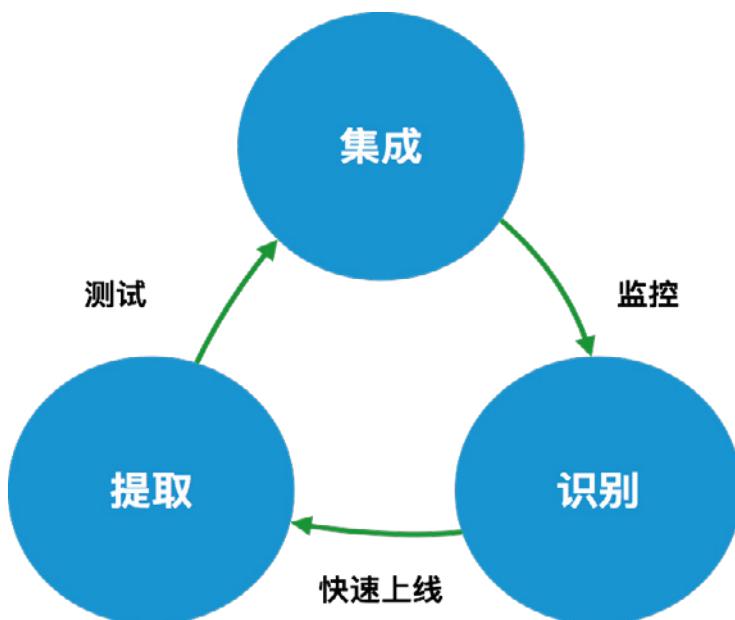


图 5

服务；devops 则用于服务的部署上线，维护，持续集成和发布环境的搭建，当然它同样遵循着组织的标准规范。如下所示，该模版主要包括 3 个部分：应用本身，部署脚本（AWS Cloudformation），docker 配置（用于构建）。

技巧二：测试：消费者驱动测试

当我们快速上线一个空服务之后，下一步就是如何快速做两个系统之间的集成，在进行系统间集成时，基本可以分为两个小步骤：定义新服务的接口+与现有系统的对接。定义服务接口可以是很简单的过程，也可以是

很复杂，主要取决于原有系统中该职责的复杂程度。但不管复杂与否，定义服务接口的过程都是一致的——消费驱动。不同于将新服务的所有接口预定义出来，它是按照消费者的需求，驱动新服务提供应有接口的，当然这里的消费者指的是现有系统。

从图 6 中我们可以看出来，现有系统（即消费者）和新服务（即服务）集成过程是由两组失败+成功的测试组成，或者说是由两组 BDD 组成。首先是消费者侧的 BDD，中间人扮演了服务的角色，并且我们假定新服务的接口已经按照消费者的要求实现完成，此时由于消费者并没有完成相应代码

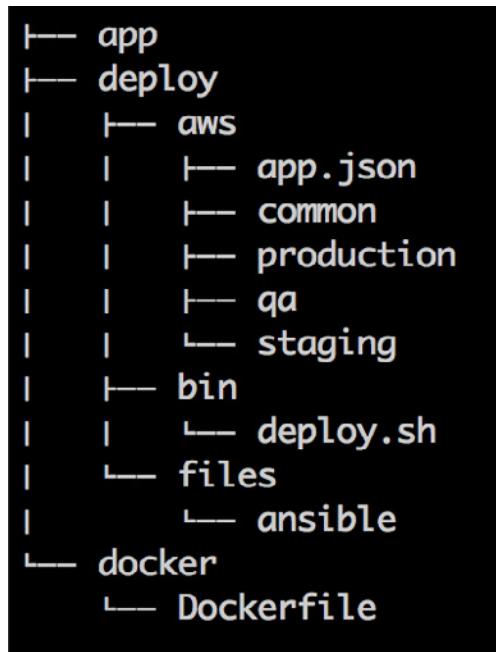


图 6

的编写，所以我们会得到一个失败的测试，接着便是真正的编码，到我们得到一个成功的测试时，意味着消费者端的集成工作就完成了；接着就是服务侧的 BDD，此时的中间人扮演的是消费者的角色，由于服务并没有定义期望的接口，所以我们会得到一个失败的测试，经过编码，我们得到了成功的测试，此时就意味着两个系统之间完成了互联互通，并且有了测试来保证。

技巧三：监控

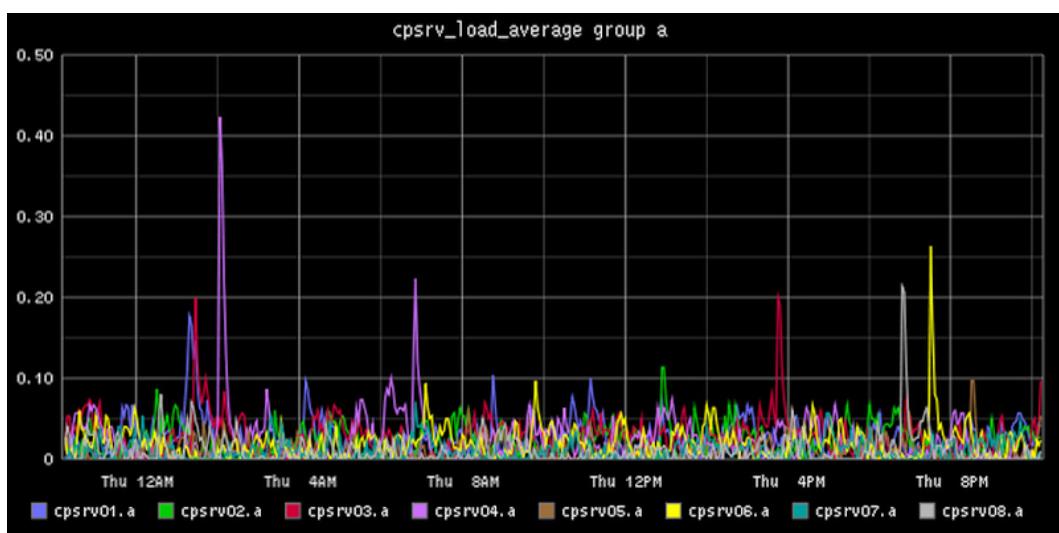
在我们拆分原有系统之前，所有的模块都被整合在同一个系统之中，对系统的监控是统一的。但是当我们将在现有系统拆分之后，就转变为多个独立系统，如何保证新服务良好运行，

或者说如何实时监控新服务的运行状态对整个系统的稳定至关重要。在实施监控的时候，我们应该考虑这些方面：

- 资源利用度

以某个虚拟机节点为整体做的关于系统资源的监控。

如图 7 所示，比如说 CPU 利用率，硬盘读写，内存利用率等，主要的目的是检查当前的节点是否出现过载或空闲的状态，得到这些状态信息之后，就能做出相应的处理。过载就说明资源无法满足当前的流量，应该增加节点，部署更多的服务来适应大流量情况；空闲就意味着资源是有富余的，应该减少节点，以便降低成本。



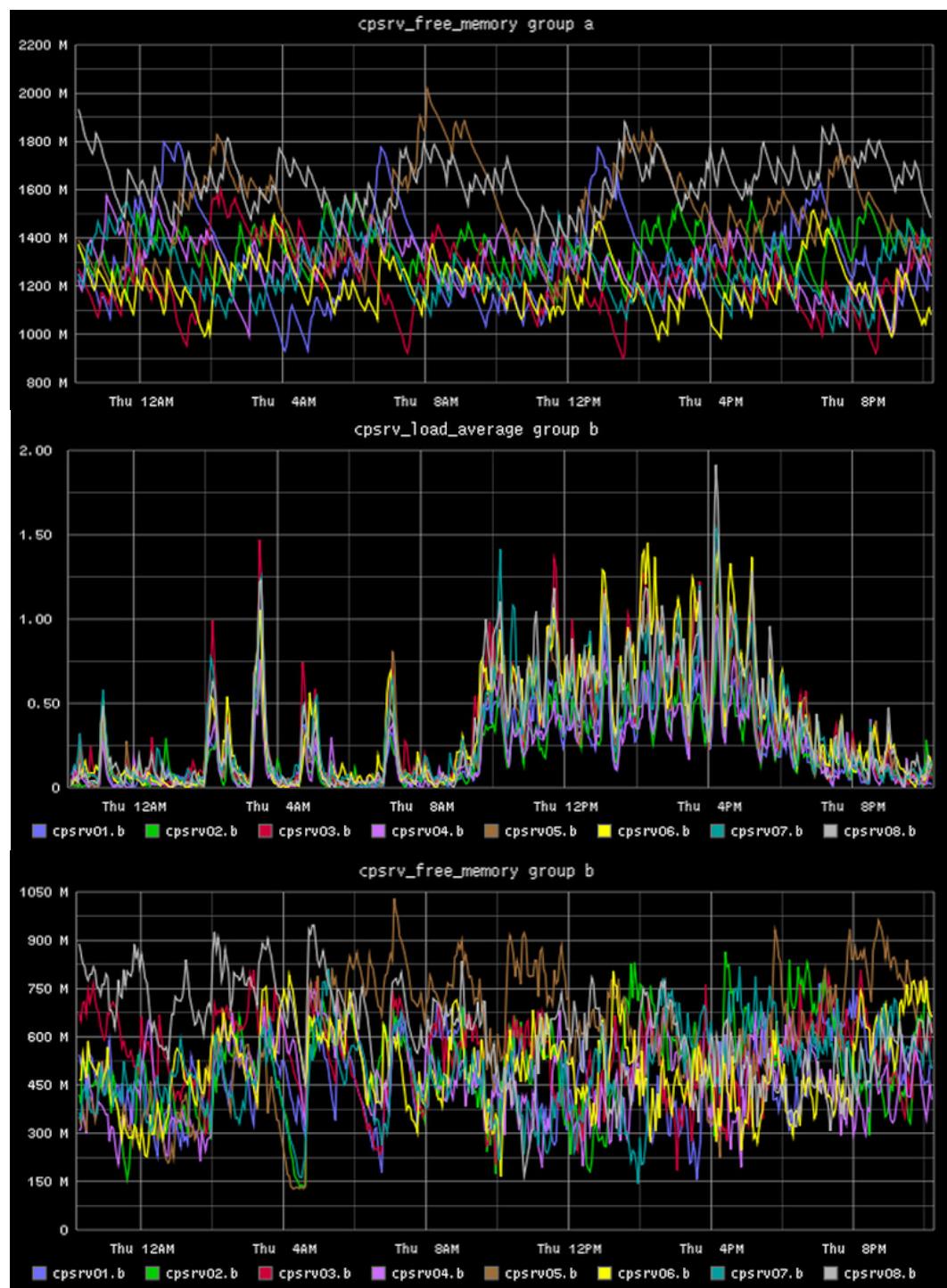


图 7

- 健康检查

以服务为整体做的粗粒度的监控。

如图 8 所示，主要的目的是为了检查当前服务是否处于运行状态。如果检测出服务属于不可用状态的话，云端的负载均衡会根据预先的配置，删除该服务，并新增加一个全新的服务用于填补空缺。也就是说，基于当前的架构我们更倾向于重建服务而非修复不可用的服务。

- 日志

所有服务内部状态的监控。

如图 9 所示，上面两种监控都无法得到服务内部运行状态，因此日志监控在这里就显示出了极大的重要性，尤其是两个系统之间集成的日志，通过对系统日志的监控，使我们具有了监控系统内部逻辑的能力，有了这种能力我们就能够追踪事故的发生并得到关键信息，进而优化服务。

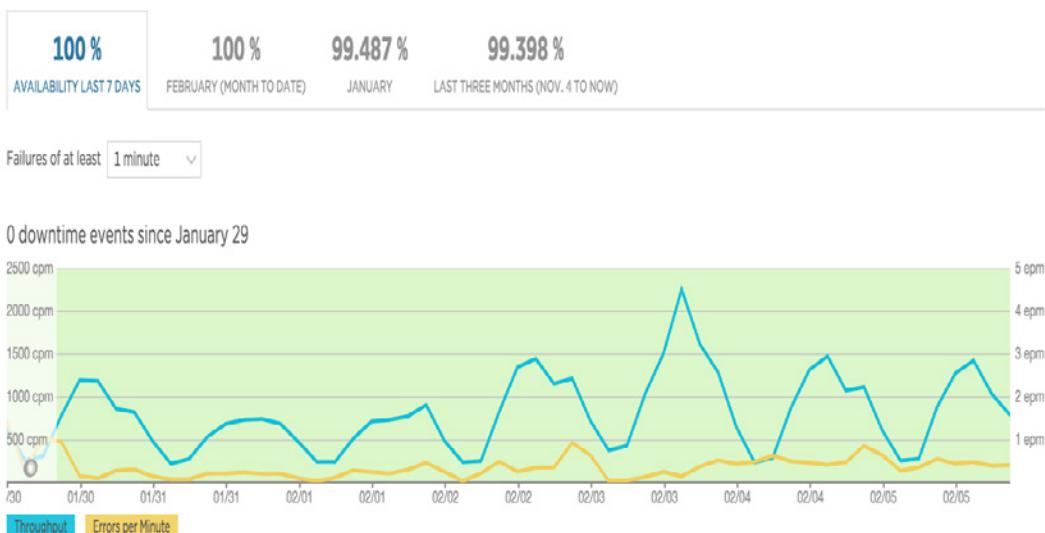


图 8

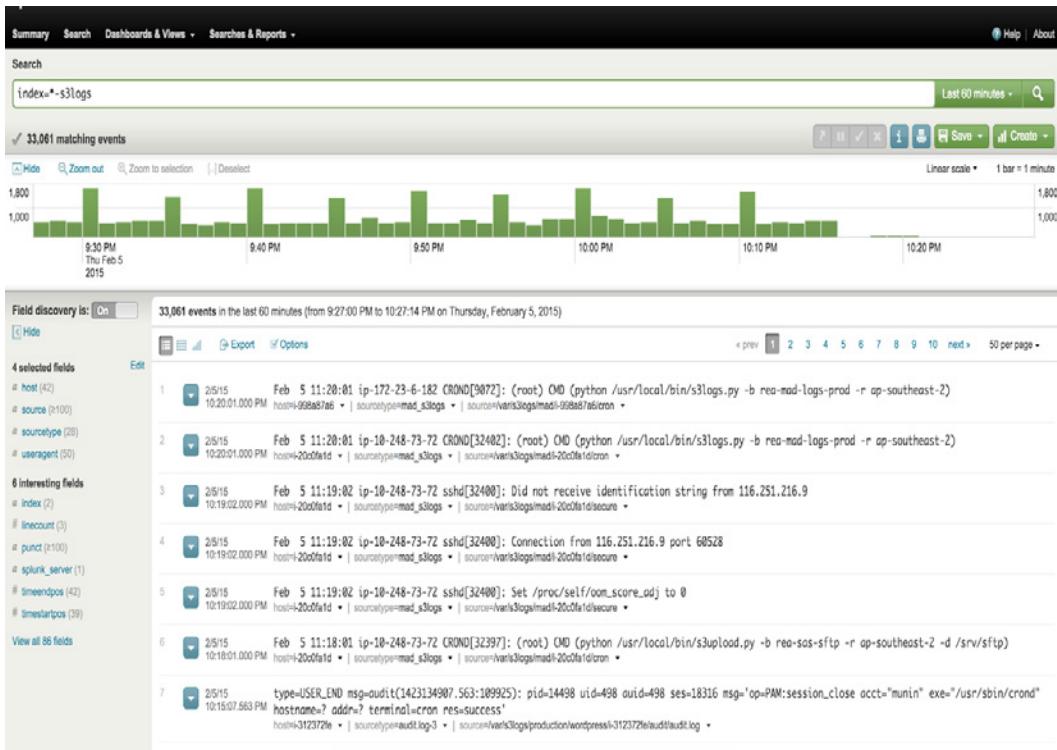


图 9

总结

从传统数据中心的基础设施向云端迁移是一个长期的过程，有可能还有很多未知的问题等着我们，但是我们发现在这同时又是一个对现有系统重新认识的过程，在这期间，我们有发现了其中的一些规律，每个功能的迁移都有各自的特点，同时又是相似的，这种相似性提取出来就是，识别，提取和集成。

只需6步，教你把服务迁移到云端

作者 张英洁

Ghost 是一个开源博客平台，而 Ghost (Pro) 是它的托管平台。这篇文章的作者是 Ghost 的高级 DevOps 工程师 Sebastian Gierlinger。他用简单的 6 个步骤，总结了 Ghost (Pro) 的基础设施从专用服务器迁移到 DigitalOcean Droplets 的过程。

以一年多运营 Ghost (Pro) 的经验来看，我们认为自己的下一代基础设施需要满足如下需求：

- 能够在几分钟内对服务器扩容；
- 拥有服务数千个博客的大内存；
- 提供强大的客户支持；
- 在不做重大重构的前提下迁移软件。

任何一个项目的迁移，都以一定程度的不确定性开始。一开始就预料到所有的迁移步骤是不可能的，更不要说预料到任何一个即将遇到的问题。但鉴于这是一篇回顾文章，出于方便

理解的目的，迁移过程包括以下几个大步骤：

- 为现有的服务器基础设施建立目录；
- 确保公网安全；
- 确保专用网络安全；
- 备份数据库；
- 更新 DNS (切换到目标服务器)；
- 下线旧的运行环境。

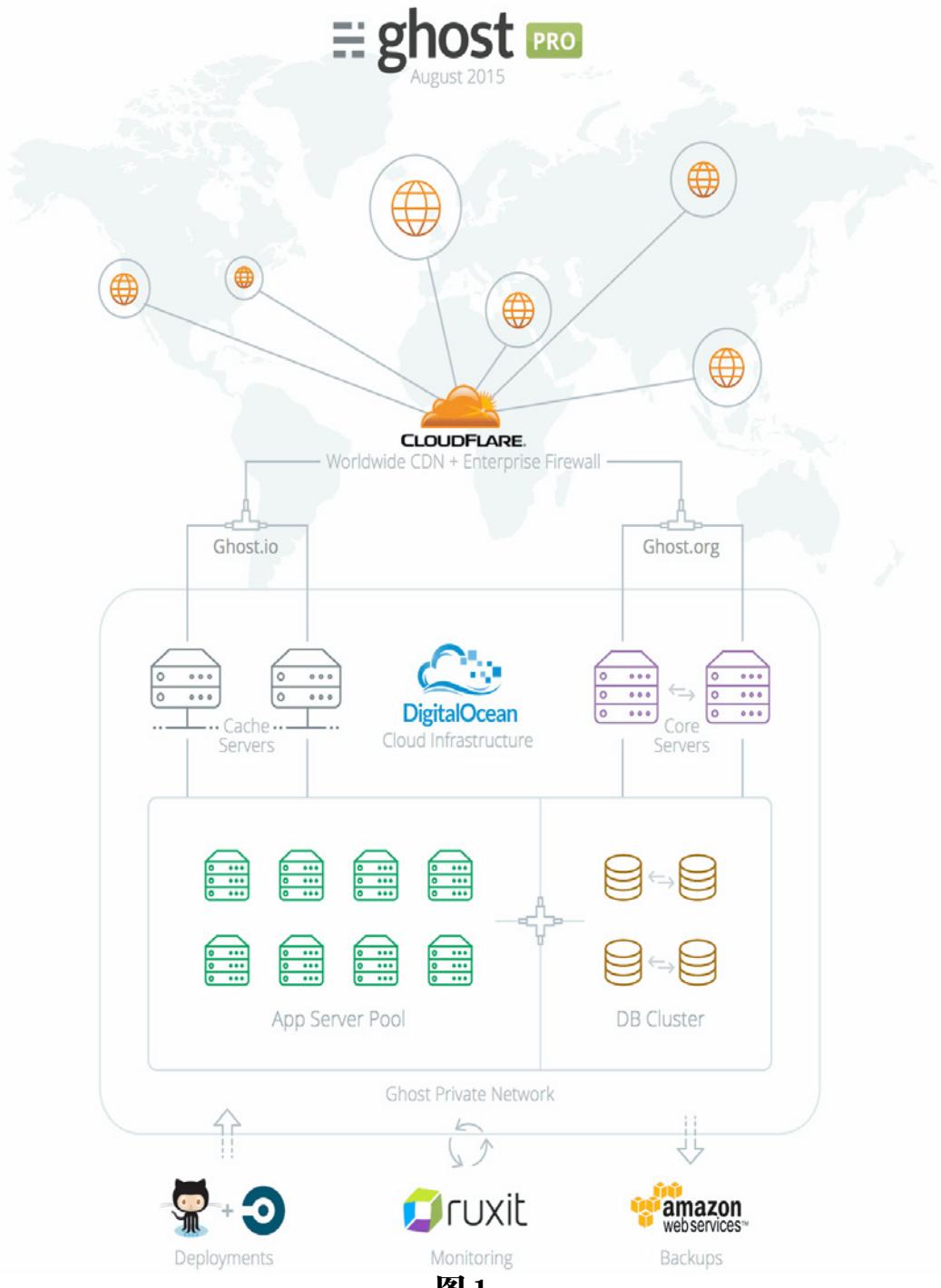
迁移后的 Ghost (Pro) 系统架构是见图 1。

下面就让我们来详细看看迁移过程的 6 个步骤。

Step 1：创建目录

第一步是用配置管理工具创建一个目录，用来显示现存的服务器基础设施上运行着什么。

我们并没有一个完整的目录，包



含所有安装的软件、防火墙设置和其他服务器配置。为了解决这个问题，我们引入了一个配置管理系统工具箱。配置管理的一大好处就是，一旦系统建立起来，它既可以作为记录文档，又可以用作一个部署工具。

我们考虑过几个比较流行的配置管理工具，包括 Puppet、Chef、Ansible 和 SaltStack。我们需要一个既能完成工作又不容易因为复杂性而过载的工具。最终就选择了 SaltStack。

下面是几个选择 SaltStack 的原因：

- 它很轻量，易于安装和维护；
- 它采用 master/minion 的架构，可以将更改从 master “推送” 到 minion 端，避免了由于频繁请求可能造成的 DDoS 攻击；
- 它拥有一个专门的主服务器，可以防止管理员和开发者直接在未受保护的生产环境部署更改；
- Master 和 Minion 的通信采用 ZeroMQ 而不是 SSH。在处理多个加密并发连接时，ZeroMQ 使用更少的 CPU 资源，效率更高。

另一个附加的好处是 Salt Cloud，它包含在 SaltStack 里，并且

和 DigitalOcean 的 API 无缝兼容，资源可以快速弹性伸缩，我们可以用命令行管理系统资源。

虽然这个新的 Ghost (Pro) 架构初始设置和配置颇花费了一些时间——大概 3 周，但 SaltStack 表现除了它强大的功能。第二步迭代配置用了大概一周，第三步部署托管平台只用了两天。

创建一个目录，并把它迁移到配置管理工具，这是一个迭代的过程，会涵盖几乎整个系统迁移的过程。除了完成迁移，创建目录还暴露了我们结构中需要改进的部分。读者如果想要尝试 SaltStack 更多功能，请参考使用手册：1、2、3。

Step 2：确保公网安全

第二步是为了确保平台内部服务器的公网防火墙安全。

以前，只有我们面向公众的服务器可以通过外网访问；其他服务，比如 APP 和数据库服务，只能通过专用网络访问。迁移到 DigitalOcean 的云端后，所有服务器都有一个公共 IP 地址，我们必须解决这项新的安全隐患。

我们给内部服务器的公网设置了一个 iptables 规则，能够拦截除了 SSH 加密通信的其他所有连接。我们

自己的服务器用的是 Ubuntu，所以用 UFW 作为 iptables 的管理界面。

和其他基础设施一样，防火墙配置是通过 SaltStack 完成的，方便于我们统一管理。

Step 3：确保专用网络安全

第三步是用 VPN 确保所有服务器专用网络的安全。

DigitalOcean 的专用网络允许同一数据中心的 Droplets 彼此通信。这个特点非常棒，它在基础设施内保证了高带宽和低延时，专用网络被共享给数据中心所有的 Droplets，包括那些属于其他 DigitalOcean 客户的 Droplets。这就是说，专用网络保护我们不会被接入一般的互联网连接，但可能接入其他不属于我们的 Droplets。

我们选择配置 VPN 来确保服务器专用网络的安全，它提供的加密和身份认证正是我们所需要的。我们选了 Tinc VPN，因为它有网状路由布局，这意味着它的流量会尽可能直接发送到目标主机。它允许通过直连的 Droplets 发送流量，而不必直接与请求者相连。不像 OpenVPN 那样把 VPN 管理复杂化，我们的 VPN 更像是个传统的专用网络。

就像防火墙配置一样，我们用 SaltStack 集中管理 VPN 配置。这样就可以在所有服务器中自动更新 Tinc VPN 的配置，从而保证了所有 VPN 网络的正确和及时更新。

Step 4：启动数据库备份

第四步是设置备份来迁移数据库。

最初，我们计划在数据库迁移期间把 Ghost (Pro) 下线。这样暂停一下，我们就可以保持数据的连续性，并为 MySQL 数据库备份。然后我们再把备份传到新数据库服务器上，最后重置一下就能完成数据库迁移。然而，分析完现有数据后，我们发现这并不可行。我们有大约 500G 的数据库，这意味着预计需要 6 小时的下载时间，还不包括任何意外的错误。服务下线那么长时间是不能接受的。我们需要其他解决方案。

我们决定首先迁移备份数据库，这样可以保证在迁移数据的一致性，同时也能保证数据库迁移过程中线上服务不中断。下面简述备份步骤：

1. 配置 Master/Slave 备份

把原始数据库服务器设置为 master，新数据库服务器设为 slave。这意味着从原始数据库系统到新架构的数据库备份是单向的。

2. 测试新基础设施

用这样的 master/slave 设置，就可以测试了，看我们新的 Ghost (Pro) 系统是不是和原有设置保持一致。所有测试完成后，我们删除了新的数据库服务器 (slave) 数据来为下一步做准备。

3. 配置 Master/Master 备份

新数据库服务器测试后，我们开启了 master/master 备份，这意味着所有的数据变更都是双向的。一旦开启 master/master 备份，原始数据库被迁移到新服务器上，新的基础设施准备就绪了。更多关于数据库迁移的教程，请参考 1、2。

Step 5：更新 DNS 记录

第五步是更新 DNS 记录，以此把新基础设施投入使用。

更新 DNS 记录有时候会出现问题，因为 DNS 生效需要时间。如果处理得当，生效时间是几个小时，用户可以使用新老系统。我们使用 CloudFlare 管理 DNS 条目，它支持实时修改 DNS，这样我们就能避免可能出现的问题了。

当准备启用新系统时，我们执行了以下步骤。首先，更新 ghost.org，使它指向新的核心服务器；然后，

测试运行；最后，更新 ghost.io，使它指向新缓存服务器，再多测试几次。

Step 6：下线旧的运行环境

最后一步就是下线旧运行环境中的服务器。所有服务都运行在新的 DigitalOcean 中，我们不再需要原来的专用服务器基础设施了。淘汰旧的设备能大大节省我们的开支。

在关停旧的基础设施前，我们需要停掉新旧数据库服务器的备份。

结语

把 Ghost (Pro) 平台从专用服务器迁移到 DigitalOcean 非常成功。我们希望分享自己的经验，因为我们知道业务迁移是很多项目面临的一个挑战。我们希望自己迁移到 DigitalOcean 的分享对其他准备迁移的项目能有借鉴意义。



消息队列和任务处理平台提供商 Iron.io 宣布了 Kratos 项目，该项目使 AWS Lambda 函数可以运行在多个云上。Lambda 函数在 Iron.io 的容器中执行，而这些容器可以运行在任何底层基础设施上。

AWS Lambda 是一个允许开发人员运行代码响应特定事件（如上传一张图片）的服务。这些代码中的函数充当事件处理程序。当事件发生时，AWS 负责启动运行代码所需的实例，并根据需要进行扩展。开发人员完全不知道操作细节。

Kratos 项目在代码和基础设施之间另外引入了一层，旨在提供一个云平台无关的平台。该层由 Iron.io

自定义的容器构成。现有的 Lambda 函数可以运行在这些容器中。Iron.io 计划针对谷歌“云函数（Cloud Functions）”做相同的事情。Iron.io 首席执行官兼联合创始人 Chad Arimura 是这样介绍 Kratos 容器的作用的：

容器打包了所有必要的组件，使代码平台无关，用户可以控制和选择如何及在哪里运行工作负载。我们发现，这就是现代企业希望采用的应用架构方式。

Kratos 项目已经创建了工具，用于将 Lambda 函数封装和打包到容器中。据 Iron.io 首席技术官 Travis Reeder 介绍，虽然它可能不支持实际

的 Lambda API，但这些函数可以使用 Kratos 端点完成相同的事情。

AWS Lambda 于 2014 年 11 月发布，支持 Java、Python 和 node.js。事件可以由各种数据源产生，如 Amazon S3（比如新增或修改一个对象）、Amazon Kinesis（比如记录流数据）和 Amazon DynamoDB（比如记录表更新）。AWS 负责自动扩展运行事件处理程序代码的基础设施。Lambda 处理程序代码可以在 AWS 控制台的编辑器中编辑，也可以打包为 zip 文件上传。代码使用 AWS 特有的结构，因此，如果要运行在 AWS 之外的基础设施上就需要修改。谷歌最近宣布了一个类似的产品，名为云函数。

在 Kratos 项目中，容器会运行专有组件，那这是否会导致客户被

Iron.io 的平台锁定？对此，Arimura 解释说：

我们也可以使用 webhook（比如 SNS/PubSub）连接到所有的 AWS/谷歌服务，但是之后，你就可以控制在哪里运行工作负载，包括使用 vSphere 或 OpenStack 的私有数据中心。

Arimura 进一步补充道，“我们提供了导入程序，可以简化 Lambda 函数的打包和运行，但将来，我们也会包含 API 兼容性，真正实现工具重用”，从而将现有的 Lambda 函数迁移到 Kratos。Kratos 将提供一个 CLI 工具，负责打包及向 Iron.io 的平台推送和注册函数。

Kratos 尚未正式发布，欢迎感兴趣的开发人员在他们的首页上参与 Beta 测试。

扫码关注回复“AWS”

详解亚马逊 S3 十年：不仅仅是存储





借助Spinnaker简化业务全局云部署

作者 Matt Raible 译者 邵思华

Netflix 最近将他们的持续交付平台 Spinnaker 作为开源项目进行了发布。Spinnaker 允许使用者通过创建管道 (pipeline) 的方式展现一个交付流程，并执行这些管道以完成一次部署。Spinnaker 能够向前兼容 Asgard，因此无需一次性完全迁移至 Spinnaker。（见图 1）

用户可在 Spinnaker 中从创建一个部署单元（例如一个 JAR 文件或是 Docker 镜像）开始，直至将应用部署至云环境中。Spinnaker 支持多种云平台，包括 AWS、Google Could Platform 以及 Cloud Foundry。Spinnaker 通常是在一个持续集成作业完成之后启动的，但也可以通过一个 cron 作业、一个 Git 库或者由其他

管道进行手动触发。

Spinnaker 还为用户提供了管理服务器集群的功能，通过应用视图，用户可以对新的服务器组、负载均衡器以及安全组进行编辑、规模调整、删除、禁用以及部署等操作。（见图 2）

Spinnaker 是由基于 JVM 的服务（由 Spring Boot 和 Groovy 所实现），以及由 AngularJS 所创建的 UI 所组成的。

为了进一步了解 Spinnaker 及其开源现状，InfoQ 与来自 Netflix 的 Spinnaker 团队进行了一次访谈，受访者包括负责交付工程的经理 Andy Glover，以及高级软件工程师 Cameron Fieber 和 Chris Berry。

InfoQ：Spinnaker 发布已经有一个多月了，社区对此的反响如何？

Glover： 社区对 Spinnaker 的接纳程度令人震惊！这个平台内置了对多个云提供商的兼容，并且能够通过一种可扩展的模型接入其中。这意味着我们打造了一个大型社区，而不是一系列专注于不同分支的微型社区。这种方式的优势在于社区中的每个人都可以利用各种创新的特性。我们已看到许多来自于新社区成员的 pull request，并且我相信，随着我们继续提升项目的可适配性，将会看到越来越多的贡献。

InfoQ：许多云提供商似乎都建议使用者上传单一的部署文件，并通过他们的 API 或 UI 进行扩展。Spinnaker 的不同之处又体现在哪里呢？

Fieber： Spinnaker 推荐使用不可变基础设施风格的部署方式，它提供了对各种云提供商（AWS AMI、Google Compute

Engine Images 等等）的镜像格式的原生支持。Spinnaker 还支持通过 Quick Patch 进行已排编代码的 push，让团队能够快速地迭代，在现有的实例中进行软件包的推送以及安装，从而减免了新虚拟机上线的等待时间。常见的使用方式是快速地部署

一个测试环境以运行测试，或发布一些有状态服务，例如数据存储的补丁。

InfoQ：你知道是否有用户已经开始使用 Spinnaker 对多个云环境进行部署吗？

Glover： 我知道有一家非常著名的公司已经在多个云提供商环境中进行部署了，不过他们希望我不要提起他们的名字。我觉得应该有其他用户也会这样做，并且随着社区的发展，我们将进一步了解有哪些公司将采取多个云环境的策略。

InfoQ：你怎样比较 Spinnaker 与 Heroku 的管道特性？

Glover： 我认为 Spinnaker 与 Heroku 的管道相比最大的区别在于：

(1) Spinnaker 支持多种可适配的部署端点，例如 AWS、GCE、Pivotal CloudFoundry 等等。(2) Spinnaker 的管道模型非常之灵活，它支持多种不同类型的阶段（stage），而且社区也可以自行开发各种管道并将其接入 Spinnaker 平台。Heroku 管道的设计目标是为了 Heroku 本身服务的，并且他们的管道模型非常僵化。另一方面，Heroku 的管道是通过命令行驱动的。我们目前还没有发布 Spinnaker 的命令行客户端。

InfoQ：从 Spinnaker 在 GitHub 上的项目来看，“gate”这个项目似乎是由 Groovy 编写的，并且使用了 Spring Boot。为什么你们选用了 Groovy 而不是 Java 8 呢？

Fieber: Spinnaker 其实就是 Asgard 项目的后继者（我们还有一个名为 Mimir 的内部工具。译注：Asgard 与 Mimir 都来源于北欧神话），他们都是由 Grails 编写的应用。我们团队对于 Groovy 有充分的了解，感觉它比 Grails 更为轻量级，并且更专注于操作性，因此值得投入精力进行研究。Spring

Boot 是一种很自然的选择，并且 Groovy 很适合应用在这个环境中。由于选择了 Groovy，我们就能够从 Asgard 中选取经过了充分测试的 AWS 代码并在 Spinnaker 中重用。

InfoQ：Spinnaker 的 UI 项目“deck”是由 AngularJS（1.4 版本）编写的，你们的开发过程是否顺利？

Berry: 刚开始的时候是比较顺利的。当我们在 18 个月之前启动这个项目的时候，Angular 表现得十分稳定。并且有大量的库（UI Bootstrap、UI Router 和 Restangular 等等）让 UI 能够十分快速地进行创建与迭代。React 也是一门非常激动人心的技术，

但当时它才刚刚出现不久，而且它的规范与模式还没有 Angular 那么充实。

但随后这个开发过程逐渐变得令人痛苦起来。其中部分原因在于 Netflix 的规模很大，我们某些应用需要在一个屏幕中渲染上千种元素，而 Angular 1.x 在处理这种数量的 DOM 节点时性能跟不上。对于这些页面，我们选择以纯 JS 进行重写，再用一些比较粗糙的方式进行性能对比。最终发现纯 JS 的结果能够满足性能的要求，即便一次性渲染几千个实例也没问题。但这种方式写出来的代码非常脆弱，毕竟 Angular 已经为你完成各种任务铺平了道路。

另一个难题在于如何让 UI 实现模块化与可适配性，让不同的云提供商能够按照他们的需求创建 UI 模块，并且让外部用户能够创建自定义的管道组件。我们在这两方面的工作做得还可以，它不算很差，但也绝对谈不上出色。我们从 UI Router 中直接抄用了大量的代码与概念，让我们的代码能够运行起来，但除了我们团队之外，我并没有看到像 Google、微软和 Pivotal 尝试开发任何自定义的实现。我想一定有某些人已经在做这件事了，只是我们还没看到罢了。

以上这些并不是说我们对于选择 Angular 1.x 感到后悔。在当时来说，

它对于我们确实是正确的选择。现在回过头来看，如果我们能够回到 18 个月之前，那我们或许会对代码进行一些重写，但大概还是会用 Angular 吧。

InfoQ：你们是否计划将 UI 项目迁移至 Angular 2？

Berry： 我们确实有进行迁移的打算，但估计要到 5 至 6 个月之后才会开始。毕竟 Angular 2 还只发布了 beta 版本，并且在工具方面也缺乏支持。那些编写 UI 特性的非 Netflix 用户有许多都不是专职的前端开发者，我们希望确保他们能够轻易地找到构建特性的正确方式，并且在遇到问题时能够方便地进行调试。

我很乐于看到 Angular 2 在明年的发展，并且想多了解一些从 1.x 迁移至 2 的案例。我们只是想对此采取一种相对谨慎的态度，并且从其他人身上多学习一点经验。

InfoQ：Spinnaker 是怎样改善 Netflix 的部署工作的？

Glover： 首先，也是最重要的一点是它为所有人提供了一个标准的交付平台。Spinnaker 让用户能够方便地进行交付，并且对于流程具备充分的信心，这正是团队最需要的东西。通过这个平台，整个 Netflix 服务

能够更频繁地进行部署，并且在运维上具备更大的弹性。Spinnaker 本身与来自 Netflix 的大量其他服务与工具进行了集成，使这些特性更易于为用户所用。举例来说，我们有一个名为 ACA (Automated Canary Analysis —— 自动化金丝雀分析) 的内部服务，这是由 Netflix 的另一个团队所维护的。尽管如此，它也是一个原生的 Spinnaker 管道阶段，能够提供测试服务。在 Spinnaker 出现之前，如果有哪个团队需要使用 ACA，就不得不自行寻找将 ACA 集成进自己的管道的方式。如今随着 Spinnaker 的出现，就为 ACA 的使用定义了一种标准方法，这也最终使 ACA 的使用得到了突飞猛进式的增长，这也提高了我们在 AWS 上的生产环境的可靠性。如果新创建的工具与服务能够提供更好的测试、数据采集或运维的弹性，就可以将它们接入 Spinnaker 平台，让每个人都能够充分利用这些工具与服务。

InfoQ：你对 Spinnaker 的哪个特性最中意？

Glover： Spinnaker 支持一种表达式语言，能够让用户对管道进行参数化。它允许用户创建一些非常复杂的管道，最重要的是还能够进行重用。它们能够在全球范围内进行构建的提送 (promote)、测试与部署。

InfoQ：你对于 Spinnaker 还有什么想补充的吗？

Glover: 虽然 Spinnaker 是由 Netflix 所开发的，但是这个项目的成功离不开与 Google、微软、Pivotal 和 Kenzan 良好的合作与他们的贡献。我们目前的良好发展状况以及将来的发展前景让我们非常振奋。我们目前正在开发的内容包括对容器更深层的支持、整体可适配性与灵活性的增长、以及 UI 和 UX 的改进。而 Spinnaker 社区的发展也让我们觉得非常激动。

Greg Turnquist 是来自 Pivotal 的高级软件工程师，他在一篇博客文章中描述了 Spinnaker 如何与 Cloud Foundry 进行结合工作。我们很有兴趣了解其他人是如何整合使用 Spinnaker 的。

InfoQ：你在什么时机下会建议 Cloud Foundry 用户尝试使用 Spinnaker 进行部署工作？

Turnquist: 对于 Cloud Foundry 的支持是在 Spinnaker 的 master 分支中开发的，其中包括大量的特性。我们目前正在计划通过活跃的客户进行 beta 级别的测试。在我看来，这对于 Cloud Foundry 的用户，无论是 PCF、PWS 还是其他 CF 的认证实例都已经成熟了。

如果你觉得目前手动将新的版本发布到 CF 的时间太长，而希望转而使用管道进行部署、冒烟测试与验证，那么现在正是使用 Spinnaker，剔除你的发布流程中低效部分的时机。

InfoQ：Spinnaker 能否简便地与 Cloud Foundry 进行整合？

Turnquist: 我觉得“简便”这种表述或许不够准确，这个词似乎暗示着整合这两个平台只需很少的工作。实际上我花了很多时间去学习 Spinnaker 的底层概念，并将这些概念与 Cloud Foundry 的概念进行一一对应。随着经验的积累，我开始认识到 Cloud Foundry 能够完美地与 Spinnaker 平台进行配合。我需要学习大量 CF 方面的知识（实际上我是在 Spring 团队工作，而不是在 CF 团队中工作），但我学到的东西越多，这两者的结合就做得越好。

Cloud Foundry 与 Spinnaker 两者都支持将应用的多个版本进行分组以进行统一的升级或回滚、在新版本与旧版本之间实现负载均衡，并且支持开发实例、预发布实例与生产环境的实例。它展现了 Spinnaker 架构的长处与灵活性，并且也展现了 Cloud Foundry 这个平台强大的能力。

InfoQ：Greg，你对于 Spinnaker 的哪个特性最中意？

Turnquist：当我谈到这个平台的时候，给我最多惊喜的是 UI 的管道编辑器，它让我能够进行各种随意的变更。在“Cloud Foundry After Dark”这个 webcast 中，我设计了一个简单的管道，其中只包含一个步骤：部署至生产环境。在我进行描述的同时，主持人 Andrew 要求我进行一些调整，让它能够实现部署至预发布环境、进行冒烟测试以及部署至生产环境。每当他话音刚落，我就已经完成了调

整。随后我们开始运行管道并通过一个对用户十分友好的界面阅读它的输出。这个平台让用户能够随意塑造流程，这是我们不应低估的一个强大特性。

InfoQ 再次感谢 Spinnaker 团队与 Greg Turnquist 能够回答我们的这些问题。在 GitHub 上可以找到 Spinnaker 的源代码。如果读者想要与 Spinnaker 社区进行交流，可以加入它的 Slack 频道、查看 Stack Overflow 上有关 Spinnaker 的问题，或关注它的 Twitter 账号 @spinnakerio。

扫码关注回复“Netflix”
看 Netflix 如何迁移到云端





解密Spotify基础设施和数据的云迁移

作者 Kent Weare 译者 丁涛

SOUNDCLOUD

2016年2月23日，Spotify宣布正将其技术基础设施和数据服务从目前租用的数据中心迁移至Google云平台（Google Cloud Platform，下同）。

Spotify是一个流行的音乐流媒体服务。它目前支持着200万播放列表，为超过7500万听众提供超过2000万小时的音乐。随着该业务逐年成长，Spotify团队正质疑自己是否愿意继续独立运营数据中心。几年前该公司认为公有云服务不能满足他们对于质量、性能和价格的期望，因此他们没有迁移到（公有）云上。Spotify工程和基础设施副总裁Nicholas Harteau，在最近的一篇博文中，解释了现在决定迁移到（公有）云上的部分原因：“云服务提供商提供的存储、计算和网络

服务已经和传统方式一样，做到了高质量、高性能和廉价。于是，我们就很容易地做出了迁移到（公有）云上的决定”。

最近Rightscale做的一项“关于云的调查”表明，在公有云的使用率方面，Google云平台排名第三，位居行业领导者AWS和微软Azure之后。Google在该领域不是领先者，Harteau解释了为什么他们最终选择Google：

“真正起决定作用的是，一直以来我们使用Google的数据平台和工具的体验。好的基础设施不只是保存和运行一些东西，它能让我们所有的团队更高效、高有效地工作。Google的数据栈的确帮我们做到了这些”。

Google对成为Spotify的合作伙

伴同样很兴奋，在它自己的博文中，披露了 Spotify 将如何使用 Google 云平台的一些具体细节。就计算服务而言，Spotify 将依赖于 IOPS SSD 和 local SSD 的高性能存储能力。他们也将利用自动扩容能力以便对“突发的场景”作出响应。其中一个这样的例子是去年 11 月 13 日 Justin Bieber 创造了单日最多音乐流的记录——超过 3600 万条流于该日被收听。使用公有云模型，Spotify 现在可以不用自己搭建基础设施，而是依赖 Google 提供灵活性来支持高峰负荷。

Spotify 也将使用 Google 的网络服务，如：Direct Peering，Cloud VPN 和 Cloud Router，以便高效地在

这两家公司之间传送 PB 级的数据。

就数据服务而言，Spotify 将放弃 Hadoop，MapReduce，Hive 而采用包括 Google 云服务，包括 Google Cloud Pub/Sub, Google Cloud Dataflow, Google BigQuery 和 Google Dataproc。Google 云平台带头的销售工程师 Guillaume Leygues 解释了 Spotify 为什么迁移到这些 Google 服务：“使用 BigQuery 和 Cloud Dataproc，数据团队可以执行复杂的查询并在一两分钟内而不是数小时内得到查询结果。这将使 Spotify 能够执行更多频繁、深入的、交互性的分析，指导产品的开发，新特性的测试和更多智能的、面向用户的特性”。

扫码关注回复“**Hadoop**”

查看大数据专栏系列文章





微吼：业务入云是一条不归路

作者 魏星

编者按：从 2010 开始，Netflix 一点点把自己的业务迁移到了云端——作为最早 All in 的案例，亚马逊将其树为典范四处推广。在接下来的几年里，凯宾斯基酒店集团、Infor 软件、日本运通、诺特丹大学、卫报传媒等等先后 All in 到 AWS。经过多年的发展，人们不再怀疑云就是未来。美国《连线》杂志的 Kevin Kelly 预言，2020 年时将有 60% 的应用运营在云上。2015 年 10 月，AWS 在赌城召开一年一度的 re:Invent 大会，发布了一系列旨在帮助客户迁移应用和数据的工具，业务入云已成趋势。在发掘这个话题时我发现，国内尚没有鲜明的案例，为此我打算走访一些企业，看看大家

对这个问题的看法。以下是我拜访微吼直播时的收获。

InfoQ：微吼是在什么情况下开始考虑往云端迁移的？迁移之前做了哪些调研？

微吼：我们做视频直播有 4、5 年的时间了。那时云计算的态势并不明显，不像现在有这么多云服务可以选择，我们只能自己建设基础设施。大概从 2014 年开始，我们启动了部分业务的迁移，因为这个时候整个云的生态已经起来了。早年的云服务可能只是巨头如华为、阿里在运营，现在除了 BAT 包括青云包括其他一些优秀的云平台的厂商都出来了。

云计算最开始是存储和主机托管方面的业务，这些服务满足了互联网上 80% 的需求；但视频直播比较特殊，对数据传输的稳定性、可靠性、延迟度、编解码等等很多方面的要求比较高，尤其是微吼做的是 to B 的服务，企业级市场对服务质量的要求更为苛刻。最早的云服务都无法满足这些要求。现在云服务生态起来了，我们做了大量的各种性能测试，发现有些云服务质量还可以，有的还不行。但云服务整体的趋势是向好的，所以我们正在慢慢地、越来越多地把业务往云端迁移。

InfoQ：整个迁移过程主要分为几个阶段？应用的迁移和数据的迁移分别是怎么进行的？迁移花了多长时间？

微吼：现在的微吼是一个非常复杂和庞大的系统。首先是富媒体流多种格式的传播，包括 Web 端很多的用户交互、后端负载均衡以及数据交互等等。我们往云端的迁移最初是分步进行的，首先是在云平台上搭建了一个网站的迷你版，用这个微缩的系统原型来做测试。这时的测试用例一般只是我们的工程师利用第三方以及自己的测试工具在全国和全球范围内对其进行压力测试等使用。对于测试效果较好的，我们会迁移一部分边缘的

业务过来，对于测试效果不好的，我们就测试其他的云平台。

原来我们的架构是有核心节点的设计，主机房我们现在用的是全国最好的 IDC。原来我们在华南、华北、华东等主要区域部署了很多边缘节点服务器，现在我们已经把一些流媒体边缘节点放在了第三方云平台。此外，我们把一些非核心业务——比如不是跟直播相关的业务系统——迁移到了云端。我们有大量的存储（都是以 TB 为计量单位）也放在了云端，因为云服务商是目前把存储解决的最好的。其实我们不喜欢单一节点这种系统架构，因为这种架构的容错能力不够强。以前没有云的支持，我们必须自建灾备节点，即便不做基础设施建设，在应用层面我们也要做大量的复杂工作。现在大部分的云厂商都提供了很好的解决方案。

存储业务放到云端之后，我们会慢慢地把核心业务，包括视频直播——事实上我们现在的视频直播流的传输已经用了一部分云服务，包括 CDN 以及 IDC 都在混合使用。而且云服务在这里面的占比越来越大——现在很多云厂商也都在提供 CDN 的服务（研发部门正在对比传统 CDN 与云厂商 CDN 的区别）。尽管我们用的都是最好的传统 CDN，但是离我们期望的效果还

是有些差距（在流量高峰时段，CDN 资源的分配机制会优先保证重点客户，也许这是 CDN 现有的商业模式），我们的解决方案是接入多家 CDN，对其进行动态调配。

从 2014 年开始，往云端的业务迁移不曾中断过，预计还要迁移几年；微吼往云端迁移的意愿很强，至于说迁移的速度，要看云服务的质量是否满足我们的需求。我们不会特别激进地一下把所有的系统都迁移到云端，这对我们来说风险太大，完全迁移到云端可能还需要一段时间的观察。云服务和我们传统的系统架构可能要并存一段时间，我们会充分利用两者的优势，二者在微吼系统以及业务上的占比会是一个长期动态变化的过程。

最开始并没有视频直播的云解决方案，我们现在也开始在更上一层——接近应用层——给客户提供一些解决方案；在底层上我们已经在用一些视频云的服务。

InfoQ：在迁移的过程中遇到过什么问题？这些问题是怎么解决的？

微吼：迁移过程中遇到的问题很多。具体来说就是，响应时间、业务不可用，或者说负载大的时候出口带宽不足，还有网络抖动，甚至是硬盘 I/O。这与其他互联网业务几乎一致，

所不同的是视频直播业务的影响会比较大。因为我们做的是视频直播对性能的要求特别高。比如说，网站和很多应用打开慢很多情况下用户是可以容忍的，但视频直播是不能容忍的；比如我们上下游也有很多供应商，我们对其资源的调配和把控能力远不如自己的那么灵便，如果在直播过程中出现问题，其影响会比较严重。

至少我们目前是不敢把所有的业务放到一个篮子里，而且国内云厂商各有各的优势，很多时候我们需要对业务进行分拆，在经过各种比较和权衡之后，分别迁移到不同的云服务上去。

InfoQ：迁移前后，微吼的系统架构有哪些变化？

微吼：主要表现为从单一节点、传统 IDC 的这种架构，向无节点、分布式系统转型。以前大三层的系统架构变成了分布式的系统架构，包括分布式的文件存储，以前的负载均衡需要我们自己做，现在云服务提供了。另外，我们整个的运维团队也作了一些相应的调整。比如 IDC 时要接触硬件、要去机房现场，现在云端了就不需要做这些工作。除了系统架构的改变，我们运维团队的业务重点也从一些很繁重但基础的工作转移到了我们业务本身。

InfoQ：早期的视频直播服务对跨平台的支持不是很好，一般以 Windows 为主。微吼目前的跨平台、多终端支持情况如何？

微吼：我们是第一家支持全平台的视频直播服务商。在客户端时代我们对这个问题感触很深，最开始我们也开发过客户端，但很快转到了 Web 端。虽然客户端在性能上略有优势，毕竟 Web 端对硬件的访问和调用隔着一个浏览器，但 Web 端跨平台的方便性和易用性的优势是压倒性的。

现在 Web 技术的发展和性能的提升使得客户端与 Web 端的差距正在逐渐缩小。除了 PC 端之外，我们也是最早推出移动端的服务商，全平台是我们追求的目标。用户使用视频直播服务应该是无缝的，无论你使用 PC 还是手机，这一切都是打通的。

InfoQ：随着部分业务迁移的完成，开发和运维工程师对迁移到云平台的反馈如何？

微吼：我们入云的尝试是一定要做的，同时迁移的效果可谓喜忧参半。因为 IDC 的优势很明确，出问题时候的响应会快一些；硬件都是看得见摸得着的，问题的可见性较强，哪怕是这个硬件坏了我热插拔都可以。在云端甚至都没有硬盘的概念了，文件都

不知道存放在哪里。但是，往云端迁移是一条不归路，团队也都认可这个理念。

InfoQ：以前微吼用传统的 IDC 机房，为了应付突然增长的并发需求和保障用户体验，可能需要长期冗余设备和带宽，运营成本和灵活性都会受限。迁移到云端之后这方面是否有改善？现在又面临哪些新的问题和挑战？

微吼：即使我们用 IDC 的时候，我们的带宽资源也是灵活的。我们跟 IDC 的签的协议是在一定量的前提下，我们根据业务量的大小可以随时调整带宽。在这一点上我们不同于一代视频网站，他们平时就需要维护大量的冗余设备和带宽资源；我们平时维护的资源是不高的，当访问量高的时候我们还有 CDN 的支持，IDC 并没有承担这个压力。此外，我们跟 CDN 和 IDC 的结算方式是先使用后结算，即，按需使用、按量计费；平时我们追求的也不是成本最优，我们买的都是最好的服务，企业客户要求的就是服务质量最好。

我们也采用一些虚拟主机，不可否认的一点是资源利用率增加了，但性能有一定差别——但是差别在不断缩小。现在唯一的问题是我们对性能

的可控力不够强，目前这个问题也正在逐步改善。

InfoQ：迁移到云端之后，微吼的运营成本与之前相比有哪些变化？相应的业务量的变化又是怎样的？

微吼：成本肯定是大幅下降，而且这个趋势也会持续下去。首先，现在的硬件包括带宽资源的价格比之前下降了很多。成本最优不是我们所追求的，所以这不是我们关注的重点。其次，我们的业务也在迅速的扩张的。原来的架构其实是一个很轻的架构，我们的瓶颈只可能是在上行带宽，所

以问题也就是部署硬件、开通节点的时间。入云之后我的响应时间会大大缩短，更多底层的工作我们不需要做了，只要去选择更好的云服务即可。

InfoQ：您能简单总结一下微吼迁移到云端的影响吗？

微吼：比如我要在上海部署节点，我要花很多时间去测试、选择合适的机房啊、购买带宽啊。而迁移到云端之后我们不需要去做这些工作了，只需测试一下，然后选择最好的服务。现在，当我们业务拓展的时候，我们只需要考虑性能方面的压力即可。

扫码关注回复“Dropbox”
了解 Dropbox 为何撤离 AWS



ArchSummit

技术峰会让学习交流畅通无阻!

ArchSummit传承经典案例，引领未来技术！

2016年7月15-16日/深圳南山区华侨城洲际酒店
中国·深圳



传承经典

云服务架构探索、发展中的移动架构技术、社交网络等专题带您领略经典行业
的技术变化



跟进前沿

大数据和个性化及高可用
架构专题与您一同探索热
门技术方向的更迭起伏



引领趋势

智能硬件、机器学习、虚拟
现实于您携手观看最新技
术前沿的波澜壮阔

5月15日前
8 折购票
团购优惠更多

sz2016.archsummit.com
购票电话:010-89880682



线上专家零距离沟通
想有更多的思想碰撞吗？
请扫描上方二维码
“ArchSummit技术关注”
为您提供足够的专家交流平台！



华章科技

软件正在改变世界

QCon
全球软件开发大会

扫码进入专题



书号: 978-7-111-52076-4
定价: 59元



书号: 978-7-111-52076-4
定价: 59元



书号: 978-7-111-52076-4
定价: 59元



书号: 978-7-111-52076-4
定价: 59元



书号: 978-7-111-52076-4
定价: 59元



书号: 978-7-111-52076-4
定价: 59元



书号: 978-7-111-52076-4
定价: 59元



书号: 978-7-111-52076-4
定价: 59元



书号: 978-7-111-52076-4
定价: 59元

你应该加入InfoQ 全职编辑团队的 3大理由



可刷脸
蹭饭蹭会



分享就是
最好的广告



跟大牛们混的多了
想不牛都难

InfoQ社区志愿者永久招募中！

6大招聘职位：

1

强力的
技术翻译

2

喜欢四处组织参与
技术活动的
形象大使

3

在任意IT技术领域
信息灵通的
线索发现者

4

深入了解任意IT
技术领域的
专业内容把关人

5

擅长记笔记的
新闻撰写者

6

知道如何
问好问题的
采访者

“

我们是InfoQ编辑，我们是信息的罗宾汉。

现在就发邮件给editors@cn.infoq.com，告诉我们你的专长和意向，我们会将你培养成为一名好编辑：）

”



扫一扫关注InfoQ

Geekbang>

极客邦科技

整合全球优质学习资源，帮助技术人和企业成长

InfoQ

技术媒体

EGO

职业社交

StuQ

在线教育

GiT

企业培训

www.infoq.com