架构师

ARCHITECT

特刊

Kubernetes PaaS冲击波

SPECIAL ISSUE

April,2018

架构师特刊

Geekbang 极客邦科技 | InfoQ

北 京 | 伦 敦 | 纽 约 | 旧金山 | 圣保罗 | 上 海 | 东 京

QCon

全球软件开发大会2018

[上海站]

2018.10.18–20

主办方 Geekbang. 极客邦科技 InfoQ

7折 预售中，现在报名立减2040元

截至2018年7月1日 团购享受更多优惠

目录

PaaS 将吞噬云计算？ Kubernetes 的市场冲击波

作者 徐川

2017年是Kubernetes的胜利之年，很多人还不明白这意味着什么。但如果看一下云计算业界的动向，你会发现，Kubernetes的影响正在扩散。

在本文中我将分享我们的发现，并试图说服你：基于容器+Kubernetes的新型PaaS将会成为云计算的主流。

我将引用很多内容，包含国内外专家的真知灼见，让你看到专家是如何看待此事的，以及分享我们自己做的调研和采访，看看业界实际在发生什么。

Kubernetes（k8s）在很短的一段时间内走过了很长的一段路。仅仅

两年以前，它还需要与CoreOS的Fleet、Docker Swarm、Cloud Foundry Diego、HashiCorp的Nomad、Kontena、Rancher的Cattle、Apache Mesos、Amazon ECS等进行竞争，来证明自己比那些产品都要优秀。而现如今已经是完全不同的一幅景象了。其中的一些公司公开宣布了项目的终止并且开始加入到Kubernetes阵营中，还有一些公司没有公开宣布自己项目的失败，而是在战略上宣布了对Kubernetes的部分支持或者完全整合，这也就意味着他们的容器编排工具将会安静而缓慢地死掉。不论是哪一种情况，k8s都是最后一个活下来的平台。除此之外，不仅仅是用户和白金赞助商们，越来越多的大公司都将继续加入到Kubernetes的生态系统中，将自己的业务完全押注于Kubernetes的成功。我们首先能想到的有Google的Kubernetes Engine、Red Hat的OpenShift、Microsoft的Azure Container Service、IBM的 Cloud Container Service、Oracle的Container Engine。

但是这些意味着什么呢？首先，这意味着开发人员必须要掌握一个与90%的容器工作相关的容器编排平台。这是一个学习Kubernetes很好的理由。同时这还意味着我们已经深深地依赖于Kubernetes，Kubernetes就像容器领域中的Amazon。在Kubernetes上进行设计、实现和运行应用程序可以让你在不同的云提供商、Kubernetes发行版和服务提供商之间自由地对应用程序进行迁移。它能让你有机会找到Kubernetes认证的开发人员，让他们来开发项目并且在以后持续提供支持。Kubernetes不是VM，也不是JVM，它是全新的应用程序可移植层，它是大家共同的选择。

——Bilgin Ibryam，Red Hat首席架构师（Source）

基于“容器 +k8s”的新型 PaaS

Kubernetes并不是传统意义上的PaaS，事实上，传统PaaS可以基于Kubernetes构建。

在过去，PaaS经历了这样的发展：

- 第一代：如最早的Heroku，严格限定的运行时，不可修改的环

境。对于Ruby on Rails这种小型单体应用来说很合适。

- 第二代：Cloud Foundry （DEA版本），可以简单的自定义环境，包括云端构建。也开始对多服务的应用有所支持。
- 第三代：Cloud Foundry （Diego版本），如当前版本的GAE和AWS Elastic Beanstalk，它们都经过之前两代PaaS迭代而来。在这个版本里增加了对容器的支持，更自由的环境配置，对微服务的支持更强大。
- 第四代：Kubernetes以及其它容器编排引擎。这一代的平台变成了Kubernetes本身，它是面向云原生应用计算的、彻底基于分布式和容器的计算平台。
- 第四代PaaS的关注点也和之前不一样，我们可以把前三代PaaS称为应用级PaaS（Application PaaS），它们关注的是应用的运行，第四代称为容器PaaS，或者CaaS、KaaS，它们关注的是应用的打包和分发。

第四代PaaS当然也可以使用其它的技术达到类似的效果，但就像前面所说的，Kubernetes赢得了这场竞争。

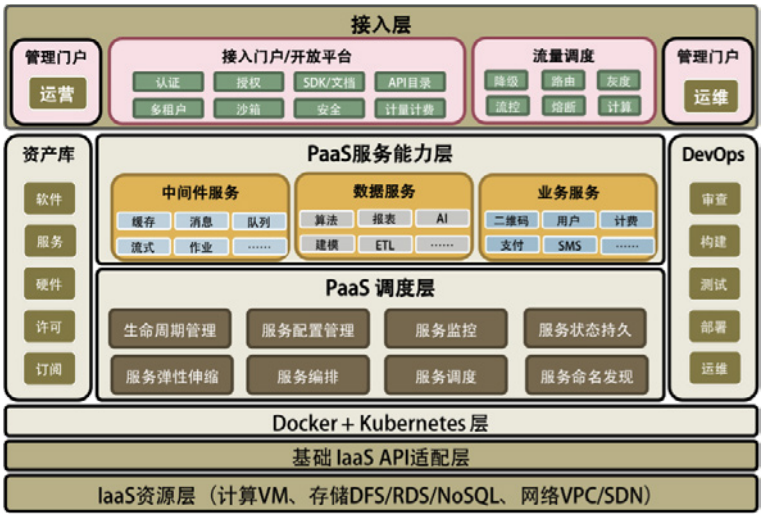
从下面的PaaS平台架构图中可以看到，我用了 Docker+Kubernetes 层来做了一个“技术缓冲层”。也就是说，如果没有 Docker 和 Kubernetes，构建 PaaS 将会复杂很多。当然，如果你正在开发一个类似 PaaS 的平台，那么你会发现自己开发出来的东西会跟 Docker 和 Kubernetes 非常像。相信我，最终你还是会放弃自己的轮子而采用 Docker+Kubernetes 的。

——陈皓 《洞悉PaaS平台的本质》

下图是一个大而全的PaaS平台架构，实际中可以根据需求进行裁剪。

业界趋势：全在做 PaaS

如果我们看一下业界，会发现，从公有云到私有云，从传统企业到互联网新贵，都在拥抱Kubernetes，都在做PaaS。



公有云全在做k8s和容器

从AWS到Google Cloud、微软Azure，到国内的阿里云、腾讯云、华为云等，都在提供k8s容器服务。如果一个公有云到现在还没有提供k8s服务，或者没有计划做，那么可以认为它的技术已经落后于时代了。

公有云提供的k8s和容器服务，具体来说分为两类：

一类是提供多租户的单容器实例，这种其实类似于上面提到的第三类PaaS，用户创建的是单个容器，值得一提的是，这类PaaS仍可构建于k8s之上，并且不少云计算厂商已经采用这种方案。另外，由KataContainer技术逐渐应用到生产环境，带来将无服务器概念和容器结合的Serverless Container Cloud理念，让容器也能兼具传统虚拟化的优点，让这类服务的未来充满了想象空间。

Kubernetes所要扮演的角色，乃是取代传统的Infrastructure Layer并鼓励技术人员进行上层的“二次创新”，而并不是直接面对最终用户。真正为最终用户提供云服务的，很大概率应该是构建于Kubernetes之上的、更加简洁高效的服务型API，而Serverless，尤其是Serverless Container Cloud的设计，正是这种需求下最为贴切的实现方式之一。

——张磊，浙江大学博士研究员，Hyper项目成员，Kubernetes项目资深成员与社区维护者

另一类是提供Kubernetes引擎，这种情况下用户创建的是Kubernetes集群，如GKE、Azure AKS、腾讯云CCS等。

第二类服务是目前公有云研发的重点，发布的时间基本集中在去年下半年到现在，我们采访和调研了微软Azure、腾讯云、华为云，情况基本类似，具体内容可进一步阅读：

剑指Kubernetes 揭秘腾讯云的PaaS技术选型策略

华为云的Kubernetes实践之路

k8s将成私有云的标准解法

私有云的情况分为两类，一类是企业搭建数据中心和私有云自用，另一类是服务提供商，为客户提供私有云解决方案。在这两类情况中我们都看到Kubernetes被使用的越来越多，并且无论是企业、服务提供商，还是客户都尝到了Kubernetes PaaS的甜头。

对于自用型私有云来说，系统的演进是一个复杂的问题，盲目采用新技术有时不仅无助于业务，还造成资源浪费。k8s的表现如何呢？我们让京东的经验来说话吧：

（采用容器和Kubernetes的）JDOS 2.0接入了包含大数据、Web利用、深度学习等多种类型的利用，并为每一种利用依据类型采取了不同的资源限制方式，并打上了Kubernetes的不同标签。基于多样的标签，我们实现了更加多样和灵便的调度方式，并在部份IDC试验性地混合部署了在线任务和离线任务。相较于1.0，总体资源应用率提升了约30%。

——鲍永成，京东基础平台部技术总监

对于服务提供商来说，Kubernetes健康的生态可以保证它们有大量的第三方软件和工具使用，同时PaaS易于开发和代码/应用复用的特性，也降低了它们交付项目的成本，并缩短了交付周期。

对于客户来说，基于Kubernetes的PaaS可以实现应用自由迁移，这使企业可以采用多重云策略，并变相提升了对供应商的议价能力。

云计算经过了十多年的发展，已然进入的云原生的新阶段，企业应用

优先考虑部署在云环境，如何顺应云原生的大潮，使用容器和Kubernetes构建云原生平台，践行DevOps理念和敏捷IT，开源软件和社区如何助力IT转型，所有这些问题的解决方案就是PaaS平台，其对于企业的重要性不言而喻。

——宋净超 TalkingData容器平台负责人（Source）

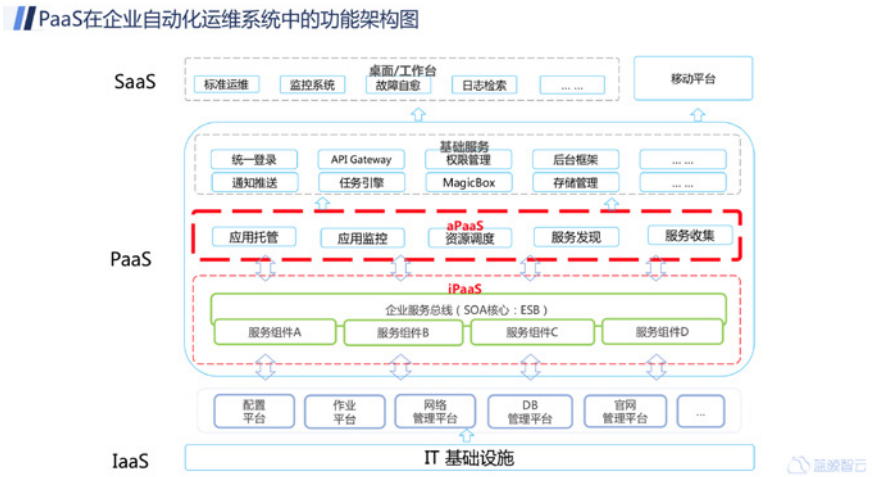
一些业界的经验可参考：

京东如何从OpenStack迁移至Kubernetes

你知道吗？传统企业已经在用最新互联网架构了

运维也需要PaaS

腾讯互娱的运维团队，需要为公司的在线游戏提供运维能力，这可能是中国挑战最大要求最高的运维服务，因此他们有数百人的研发团队，他们的做法可以很大程度上代表运维的发展方向，而不断思考和迭代的结果就是自研了一套PaaS平台蓝鲸。蓝鲸本身不使用Docker、Kubernetes等，完全自研，但我们可以看到，运维的发展方向就是PaaS。



PaaS本身与DevOps的理念完全契合，它改变了传统运维的职责，让他们变成运维开发，为企业研发运维工具乃至是PaaS平台。而对于没有蓝鲸团队开发能力的人，容器和Kubernetes能为他们提供弯道超车的捷径。

京东金融的运维团队就采用了Kubernetes来搭建他们的PaaS平台。

PasS平台化将问题的关注点从基础资源上升到了应用层面，目标是提供一个帮助开发人员运行、管理应用的平台，让使用者更关注运行的代码（业务逻辑）。

PaaS能解决的问题：

- 应用聚合：如开发需要一个Redis，直接启动一个Redis容器即可
- 服务发现、快速伸缩、状态管理等
- 服务监控、恢复、容灾
- 费用统计：提供计算资源信息汇总，针对不同项目收费
- 安全管控：不管什么平台，安全都非常重要，例如A应用可以访问B,B不允许访问A以及安全审计等。
- 快速部署。

随着Docker容器技术的出现，让我们有了更合适的工具建设PaaS平台，具备了基于应用构建服务的能力。在Docker容器调度框架上，我们又选择了Kubernetes平台。

——张龙，京东金融PE

为什么 PaaS 会成为云计算主流?

除了上面的这些，我们还可以看到，PaaS是SaaS服务发展到一定程度后必然会做的事情，这么做不仅可以满足客户更全面、定制化的需求，也让SaaS厂商可以向更多领域拓展。如果要举一个例子的话，大家想想微信和小程序就能理解。

而为什么Kubernetes会成为PaaS的选择，为什么PaaS会成为云计算的主流，是因为容器和Kubernetes是今日云原生概念的核心和基础。云计算诞生到现在有十来年了，但云时代的应用应该长什么样子，过去一直没有人能说清楚，直到容器诞生后，我们终于离想象中的云时代稍微近了一些。

通过了解软件工程的这三个本质，你会发现，我们上面所说的那些分布式的技术点是高度一致的，也就是下面这三个方面的能力。

- 分布式多层的系统架构。
- 服务化的能力供应。
- 自动化的运维能力。

只有做到了这些，我们才能够真正拥有云计算的威力。这就是所谓的Cloud Native。而这些目标都完美地体现在 PaaS 平台上。前面讲述的分布式系统关键技术和软件工程的本质，都可以在 PaaS 平台上得到完全体现。

——陈皓 《洞悉PaaS平台的本质》

云计算的未来

过去几年云计算的发展令人眼花缭乱，想要预测它的未来无疑是极为困难的，但只要把握住Kubernetes这条主线，理解从虚拟化到容器再到两者融合的发展路线，在短期内我们还是能做一些预测。

这个问题（Kubernetes在五年后会变成怎样）很好。我希望，在接下来的五年中，我们对Kubernetes的讨论不比对Linux内核的讨论多。它真的应该成为所有工作的基础。如果我们接下来的行为正确，我认为，有些事情就会成真。

大多数开源和ISV（软件供应商）的安装指令都是始于“选择一个经过认证的Kubernetes集群”。第2步将是“运行这个kubectl命令”。Kubernetes将让第三方软件不再忧虑开发针对无数平台的版本，让那些供应商更容易提供云提供商托管服务之外的方案。在许多情况下，使用云服务并没什么不对，但是，你应该从你自己的基础设施上也能获得类似的体验。

我相信，对于开发流程，我们将从封闭的PaaS服务，转向企业可以使用一流组件组装类似PaaS功能。其中，有些可能是领域专属的，只在一个特定的行业里应用。企业能够快速组装一个完整的解决方案，提供从代码到有强大防护的生产环境的简单路径，也提供在需要时“打破玻璃”运行自定义功能的能力。

——Craig McLuckie，Kubernetes创始人

Kubernetes已经胜利，但基于Kubernetes的各类组件、工作流并不成熟，就像Kubernetes创始人McLuckie所说的，Kubernetes需要成为讨论的“背景”，我们讨论的将是基于容器编排的各种创新和应用，比如Service Mesh。

在我看来，在三到五年之后， Kubernetes 会成为服务器端的标准环境， 就像现在的Linux，而 Service Mesh 就是运行在 Kubernetes 上的分布式应用的动态链接器，届时开发一个分布式应用将会像开发单机程序一样简单，业界在分布式操作系统上长达三十多年的努力将以这种方式告一段落。

——宋潇男，普元信息云计算架构师（Source）

剑指 Kubernetes 揭秘腾讯云的 PaaS 技术选型策略

作者 江柳

前言

Kubernetes 很火，一大批互联网公司早已领先一步，搭建起专有的PaaS平台，传统企业们看到的 Kubernetes的趋势，亦不甘落后，在试水的道上一路狂奔……

虽然，Kubernetes很火，并不代表可以“上手即用”，基于Kubernetes的容器编排也不是简单的“拿来主义”。在容器圈，除了Kubernetes，还存在着 Mesos、Swarm等分属不同阵营的容器集群管理工具，以及基于这些工具的多个容器云提供商。

腾讯云在 2016年底决定开发容器产品，随后组建容器技术团队并进行技术选型，通过对不同编排工具的分析对比，最终选择 Kubernetes 作为容器编排引擎，并且迅速在 2017 年初推出容器解决方案 CCS，为用户提供托管 Kubernetes 的一站式服务。

随着业务量的增加，腾讯云容器团队基于 Kubernetes，不停的增加和完善容器监控、日志处理、容器 Registry 等关键特性，来保障用户业务的平稳运行。在整个 2017 年的运营过程中，管理了数千个集群、数十万容器，以及业务高峰期的成倍扩容。

那么，在 16年底，Kubernetes并未大热时，腾讯云为何偏偏在诸多主流的编排引擎当中选择 Kubernetes？腾讯云 Kubernetes架构和资源调度原理是什么样的？在用户托管服务中，又为何采用 Kubernetes来托管用户 Kubernetes集群的 Master？作为国内最大基于 Kubernetes的容器服务提供商，腾讯云在 Kubernetes上还有哪些应用实践…… 本文将带着这些疑问，为你一一揭开腾讯云基于 Kubernetes的 PaaS平台神秘面纱。

为何选用 Kubernetes？

容器技术无疑是近几年最热门的技术之一，很多公司或者行业已经把容器作为自己的测试环境以及正式环境，正式上跑一些业务。腾讯紧随外界技术发展的潮流，在三年之前，腾讯云的基础平台部门开始容器方面的技术实践，经过三年时间的积累，腾讯云目前已经有很多业务跑在容器平台上。

在最开始的容器产品技术选型阶段，腾讯云也曾对比过 Kubernetes、Docker Swarm、 Mesos：

- Kubernetes的核心是如何解决自动部署，扩展和管理容器化（containerized）应用程序。它支持资源调度、服务发现、服务编排、资源逻辑隔离、服务自愈、安全配置管理等。
- Mesos是一个分布式内核，核心理念是数据中心操作系统（DCOS），为了解决 IaaS层的网络、计算和存储问题，

所以 Mesos的核心是解决物理资源层的问题。它同时支持Marathon、 Kubernetes 和 Swarm 等多种框架，Mesosphere 也是 Kubernetes 生态的一员。

- Swarm：从 Docker1.12版本开始，Swarm随 Docker一起默认安装发布，也由于随 Docker引擎一起发布，无需额外安装，配置简单。它支持服务注册、服务发现，内置 Overlay Network以及Load Balancer。Swarm是与 Docker CLI非常类似的操作命令，对熟悉 Docker的人非常容易上手学习。

综上来看，每一种工具都有自己的核心理念。

但是，腾讯云最终选择了 Kubernetes， 现在看来这个选择无比正确。 在技术选型上，除了编排引擎自身的核心特性，腾讯云也主要从以下几个方面进行了评估。

模块	Kubernetes（最终选择）	Swarm	Mesos
社区	23522 stars, 8300 forks 1216 contributors **远远领先与Swarm 和Mesos**	4580 stars, 952 forks. 164 total contributors	3126 stars,,1265 forks. 247 contributors
能力	**整体能力强，已大量运用于生产环境** 包括集群管理、服务管理、任务管理、容器管理、服务发现、安全等基本覆盖了容器所需的所有能力	能力在持续扩展中 能力比Mesos强，比Kubernetes弱	仅提供调度能力
易用性	功能较多，有一定门槛	**易用性比kubernetes好**	调度能力，易用性较弱
核心亮点	**功能强大、社区活跃，可发展性好**	**兼容docker标准API**	**轻量，只涉及调度**

注：表格里的材料是当初选型时调研的情况，上述编排工具现在的特性已经有了新变化。

从整个行业来看，业界的技术架构正在大规模向微服务迁移，容器技术天生就是部署微服务的最佳方式，腾讯云拥有海量的业务架构，而 Kubernetes 的使用让部署大规模的微服务更加容易，这也是腾讯云选

择 Kubernetes的主要原因，另外腾讯云选择 Kubernetes还考虑了其它优势，如：

- 出身名门 Google，其开发和设计受到了 Google著名的 Borg系统的影响；
- GitHub上关注 Kubernetes项目和提交代码的开发者非常多，社区活跃，如果遇到问题，通过社区咨询和解决 问题速度也会比较快。
- Kubernetes可以很好的支持有状态的服务。

Kubernetes 是近 3 年来社区热度最高的项目，Linux 基金会也成立了 CNCF来加强社区运作，不仅对 Kubernetes 进行管理，还对相关项目进行有序的运作，保证了整个技术栈稳定和蓬勃的发展。Kubernetes技术相比Docker swarm和Mesos 学习难度更高，腾讯云为此提供了免费容器实验室，帮助开发者快速入门学习kubernetes，感性趣的读者可以点击了解详情。

腾讯云目前有哪些容器解决方案?

腾讯云 PaaS平台发展至今已有了丰富的容器解决方案， 包括但不限于：腾讯云 Kubernetes托管服务、单容器实例服务、TencentHub服务、Flow Engine服务、CCI持续集成服务等，以及基于 Kuberentes 的 TensorFlow 和 Spark 解决方案，帮助用户在使用容器解决方案的同时进行数据计算和模型训练。

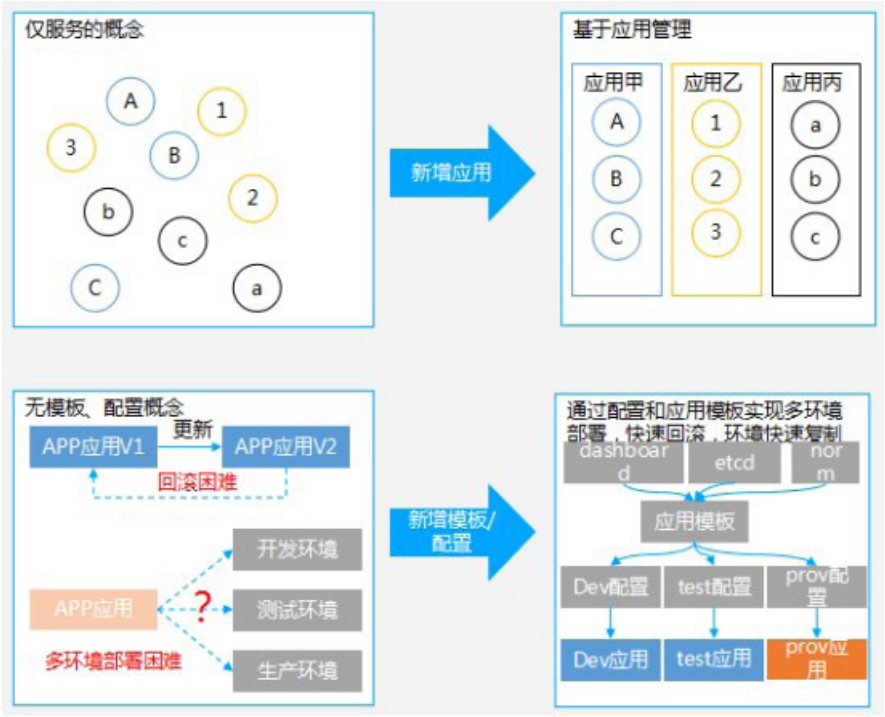
- 腾讯云 Kubernetes服务：基于原生 kubernetes 提供以容器为核心的、高度可扩展的高性能容器管理服务。腾讯云容器服务完全兼容原生 kubernetes API ，扩展了腾讯云的 CBS、CLB 等 kubernetes 插件，为容器化的应用提供高效部署、资源调度、服务发现和动态伸缩等一系列完整功能，解决用户开发、测试及运维过程的环境一致性问题，提高大规模容器集群管理的便捷性，帮助用户降低成本，提高效率。
- 单容器实例服务：基于 Clear Linux 技术提供的单容器实例解决方案，通过单容器实例服务享受容器的便捷、虚拟机级别的隔离， 像使用虚拟机一样如丝般顺滑的来使用容器，非常试用于通过容器来进行批量计算、通过容器来快速部署个人站点。不依赖VM的容器在配置和使用时间上更灵活，提供了一种更低成本的计算资源。目前，但容器实例服务正在内测阶段。
- TencentHub服务：在云原生应用盛行的今天，仅仅提供 Git 服务或 Container 镜像存储的 Hub 已经无法满足用户在日常开发和运维过程中的种种需求。构造支持多种文件格式、容器镜像格式、编排方案的云原生 Hub 能更好的降低用户的架构向云原生迁移。 TencentHub不仅仅提供一个私有镜像的存储，同时提供了Helm包、二进制文件等文件存储。
- DevOps 产品：DevOps 任务容器化、资源及执行全托管至Kubernetes、提供给用户一个全生命周期的 Workflow，完成容器构建、测试和部署等 DevOps 流程，支持用户通过界面拖拽定义 Workflow ，支持 Workflow 定义文件导入等功能，和腾讯云已有基础服务和 DevOps 服务打通。核心引擎开源到容器社区，组件库作为配套功能同时开源。

腾讯云是如何基于 K8S 做容器集群的管理?

腾讯云容器服务应用编排

Kubernetes原生的方案中，基于服务粒度对系统组件进行管理，支持服务注册发现和路由管理。对于一个服务提供多种不同的资源描述类型，比较常用的有 Deployment、Job、CronJob、Stateful、Daemonset。

腾讯云容器服务参考社区 Helm的实现形式，在容器服务中实现了完整的应用编排管理功能。腾讯云容器服务编排服务主要分为配置管理，应用模板管理，基于应用的服务组管理几个主要部分。



配置管理

配置管理是指将应用中常变的值以变量的形式替代，配置项支持多版本，方便用户进行更新和回滚应用。关于配置管理的实现，腾讯云支持三种实现方式：

- Helm的模板文件支持变量渲染
- Kubernetes的 ConfigMap中环境变量方式
- Kubernetes的 ConfigMap通过 Volume方式进行数据填充

其主要作用包括：

1. 通过提取出多个环境中不同的部分，支持同一套系统在多个环境中部署。
2. 提取出服务中经常变更的部分，实现服务的灵活变更。并且通过配置文件的版本管理，可以很好的对变更进行追溯和回滚。
3. 通过配置项，可以隐含的实现多个服务直接依赖关系的管理。例如：服务 A需要访问服务 B的，一般情况下依赖于服务 B的名称。这种情况下，将服务 B的名称提取成为一个配置项，而将配置项传递给服务 A。这样在不同环境中，服务 B名称的改变，服务 A能自动感知，不需要单独修改服务 A的参数。

应用模板

应用模板包括多个服务的定义加一个默认配置，通过应用模板 +配置项的组合，方便用户部署相同应用的不同环境。主要用于描述一个或多个服务的定义，服务的定义支持原生的 Yaml语法，但可以通过 GoTemplate 的方式定义对应的变量。其主要作用为：

- 实现应用的快速克隆。由于应用的相关信息已经通过对应的Template文件进行了描述，复制应用的过程只需要针对性的修改应用的配置，其他结构信息不需要进行改变。
- 应用的多环境部署。在多个环境中，实现应用的部署，也不需要关系每个服务具体的部署信息，只需要在不同环境下修改环境对应的配置，即可以通过应用模板实现在新环境应用的快速部署。
- 更高阶的功能，通过应用市场可以下载通用的模板，快速的部署应用。例如：在 Helm(Charts)的应用市场 https://kubeapps.com/，已经打包好了 100+应用的模板文件。直接下载对应的应用模板就可以实现应用的部署。

应用

应用包括描述多个服务以及这些服务间的相互调用依赖关系 ，方便用户管理多个服务。应用可以理解为多个服务的组合，多个服务会统一进行展示，服务支持按照应用进行搜索。多个服务的配置项，统一的通过同

一个配置进行管理。通过服务组的方式，管理多个服务。可以简化多个服务管理的复杂度。

应用中的服务支持单独编辑，部署和更新。同时服务支持差异化比较，方便用户查看两次修订之间的差异。在服务编辑时，自动提取出对应的变量，简化配置的过程。并支持服务回滚功能，支持服务回滚到上一个版本。

用 K8S 托管用户 K8S 集群的 Master

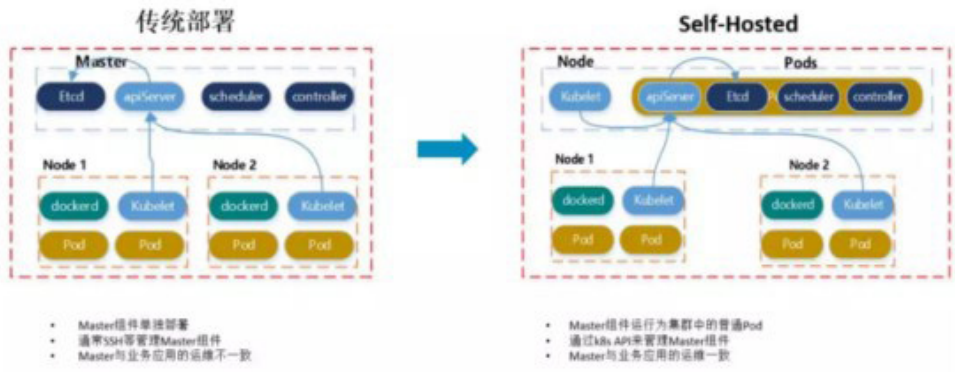
与其他云服务商不同的是，腾讯云为用户管理 Master 节点，最重要的挑战是为了保证用户 Kubernetes 集群的高可用，将用户 Kubernetes 集群的 Master 节点的进程以容器方式运行，发挥 Kubernetes 的优势提供稳定保障。

在这个过程中，随着托管集群数量的增加，Kubernetes 的 Master 节点的负载也越来越高，腾讯云提供的 Kubernetes 托管服务面临巨大的稳定性挑战：

- 多集群共享 ETCD面临性能瓶颈、运维困难、可用性较难持续保障
- 大量小集群产生 master服务器成本增加,可用性、性能较难保障
- 用 agent管理节点，随集群增加安装包、证书、配置越发难以维护，更新回滚困难
- 提供给用户的监控与后台使用的监控是完全不同的两套，监控对象与环境不统一，维护困难

Kubernetes为高效运维而生，为什么运维起来却这么难？腾讯云为用户提供了高效的 Kubernetes服务，为什么自己做不到？ 通过多次的技术方案碰撞，腾讯云最终选择用 Kubernetes来管理 Kubernetes的方案，让自己成为自己的第一个用户。

如下图，腾讯云将用户集群 Master 节点的组件全部使用容器的方式部署在一个 Kubernetes 集群里进行管理，最大限度的利用了容器的优势和 Kubernetes 本身的能力，通过这样的部署方案：



- 用户集群 Master不再独占云主机，解决成本问题
- 健康检查即可实现基本 HA，多副本可提供更高可用性
- 单个用户集群 Master相关数据放在同一个 namespace下
- 无需额外 db存储用户集群版本信息
- 升级时备份 namespace下数据即可快速回滚
- 无需开发、维护额外的 agent

腾讯云 Kubernetes 高可用托管架构解析

用 K8S管理 K8S方案说起来简单，但将用户的所有 Master 节点的组件全部使用容器的方式部署在一个 Kubernetes 集群里，其实现过程也并不容易。具体到落地实践上，腾讯云是如何做的呢？下图为腾讯云的Kubernetes高可用托管架构。

托管服务中将整个平台分为控制面和数据面两层。控制面节点中驻留的是用户 Kubernetes 集群的 Master 相关容器和腾讯云容器服务的服务管理服务节点；数据面的节点是客户运行业务的节点。

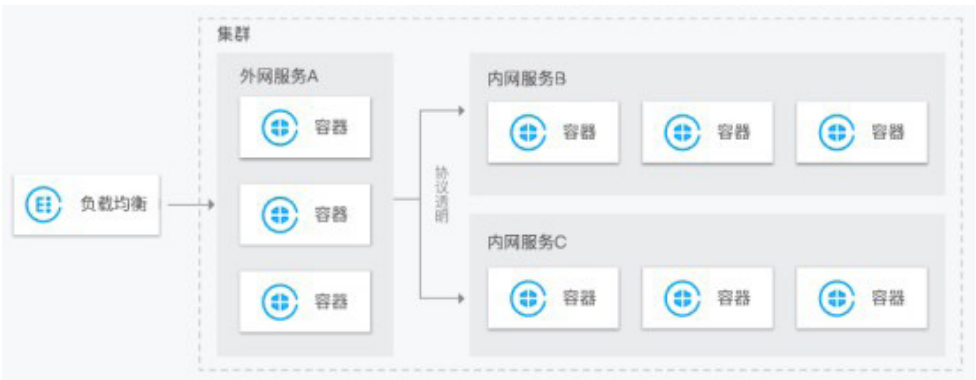
服务管理节点是由一组高可用的 Kubernetes Master 节点组成，同时包括一组高可用的 etcd 集群。服务管理节点外的其它机器都属于该 Kubernetes 的 Node 节点。当用户在腾讯云的控制台申请一个 Kubernetes 集群时，在这些 Node 节点上把 etcd/api-server/controller/scheduler 等 Kuberneter Master 必须组件以 Kubernetes Controller 的方式部署在集群内，并且通过 ingress 暴露到 VPC 。用户 Node 节点在初始化时指定要加入的 Kubernetes 集群。

采用以上架构方案，当用户集群负载升高的时候，可以快速启动更多的容器应对访问压力。同时无需开发额外的 agent 管理 Master 节点，减少管理环节有效的降低运维难度。腾讯云可以通过容器的滚动升级，快速为用户升级到最新的 Kubernetes 版本，享用最新的特性。可以方便的为客户部署多种版本的 Kubernetes ，甚至是用户的定制版本。

如何用腾讯云容器解决方案部署微服务?

腾讯云 CCS解决方案介绍

微服务架构适用于构建复杂的应用，腾讯云 CCS容器解决方案将单体式应用从不同纬度拆分成多个微服务，每个微服务的内容使用一个 Docker 镜像管理。在功能不变的情况，应用拆分成了多个可管理的服



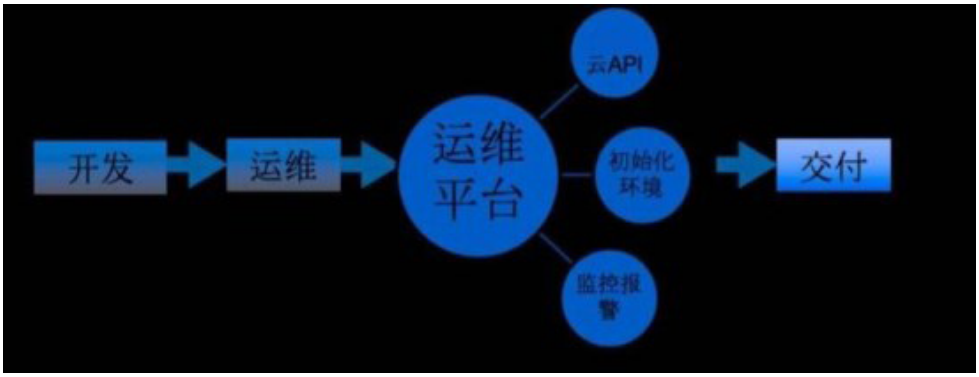
务，使得每个单体的服务更容易理解、开发和维护。

不同的微服务也可以由不同的团队来开发，开发团队可自由选择开发技术和程序语言等，每个服务又可独立部署、独立扩展。例如 Web 应用程序可以分割成一组更简单的外部服务及多组内部服务。

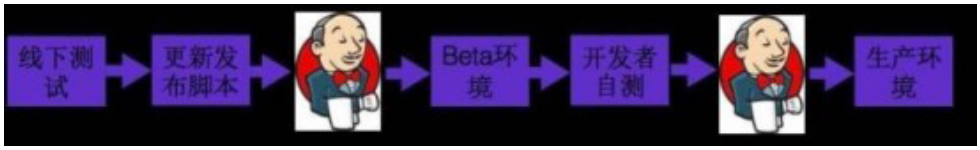
小红书基于腾讯云的微服务改造实践

小红书是一家发展非常快速的公司，技术团队在急剧增加的同时技术栈也在不断变迁。以前小红书的研发团队采用的是纯 Python 的技术环境，随着业务的发展，不同的团队在做 Java、Go、Node等不同方向尝试，同时在每年电商大促时，扩容在现有模式也很耗时耗力。为了为了支撑业务的发展， 小红书利用腾讯云的 CCS解决方案对业务系统进行微服务化改造。

在微服务改造的过程中，小红书基于腾讯云容器服务搭配 Jenkins、GitLab、Prometheus 和 Spinnaker等开源组件，以最小的投入，最低的开发量快速的实现容器化微服务架构。



上图是小红书之前的应用上线的过程，开发向运维提需求，需要多少台服务器，运维依据需求去做初始化并交付给开发。现在小红书有一个运维平台，所有服务器的部署都是由这个平台来完成的，平台调用腾讯云 API 生成服务器，做环境初始化，配置监控和报警，交付给开发的是一个标准化好的服务器。



如上图，开发者拿到服务器准备线上发布时用 Jenkins 触发脚本的方式：用 Jenkins 的脚本做测试，执行代码推送。当需要新加一台服务器或者下线一台服务器，要去修改这个发布脚本。 发布流程大概为：Jenkins 脚本先往 beta 环境发，开发者在 beta 环境里做自测，自测环境没有问题就全量发。

经过架构的系列微服务改造实践，小红书的资源利用率提高了100%， 计算资源减少将近 1倍，稳定支撑了双 11等大型促销活动。除了应用于微服务架构改造，CCS解决方案也能够根据业务运行情况，提供集群和服务两个层级的弹性伸缩能力，并为持续集成与持续交付提供的高效的 DevOps 环境。

未来规划

在 PaaS平台的未来规划上，腾讯云将会提供更多的容器基础解决方案，支持包括 Docker、OCI、Kata 等容器引擎和镜像格式。同时推出更多配套的 DevOps 服务，为用户提供完整的 PaaS + DevOps 开发部署环境。

据悉，腾讯云单容器实例解决方案 CIS计划在 5 月底发布公测，用户可以通过 CIS从容器 Image 当中快速创建容器实例。同时提供 DevOps 编排引擎的 Tencent Hub 的一部分已经上线，计划将在 5 月底正式发布公测，而在黑石上部署的 CCS 解决方案也即将在 4 月初上线。

华为云的 Kubernetes 实践之路

作者 华为云应用服务技术团队

华为与Kubernetes的渊源颇深，早在Kubernetes刚开源的时候就以社区创始成员及白金会员的身份加入其中。目前拥有1个Steering Committee席位和5个Maintainer席位。

1.1 华为自身基于 Kubernetes 的实践

加入初期，作为全球最大的电信设备制造商之一，华为内部IT运维着遍布全球的八个数据中心，在100K + VM中运行800多个应用程序，使用虚拟机封装应用程序，但每次启动虚拟机都花费了大量的时间，这给管理及部署基于虚机应用程序的高成本和低效率带来了严峻的挑战。因此华为决

定利用Kubenetes技术对自身IT系统进行容器化改造。

与此同时，华为通过参与和贡献 Kubernetes项目，为自身带来了在规划、网络、多集群联合、应用支持、安全、可扩展性和政策执行等方面的良好设计、代码和文档管理，以及在服务治理方面的收益。通过自身的容器化改造实践，在受益的同时又将自身遇到的实际问题不断的贡献给社区，与社区成员一同推动Kubernetes的发展。

比如，在华为内部IT系统的实践历程中，业务的全球化属性给平台带来了混合云、跨地域、多DC部署方面的需求，这与社区发展多集群联邦的理念不谋而合。因此，华为在集群联邦项目成立之初就积极参与其中，主导了架构设计以及联邦级别的无状态应用、短任务支持、集群间策略调度、应用跨集群自动伸缩等关键特性开发。目前集群联邦已在社区正式孵化为独立子项目。

另一个例子是早期的K8S并不支持亲和反亲和等高级调度策略，使得大型传统应用改造上云十分困难。在华为公司对应用做微服务拆分和容器化改造的过程中，最典型的问题就是拆分后的组件间如何高效地通信。在传统方案中，往往有多个组件的业务进程部署在同个虚机上，组件间交互可以通过进程间通信来实现。应用改造后，组件变成了k8S中的Pod，被相对独立地调度和拉起，通过容器网络互相通信。当一个复杂应用的各个组件十分分散时，网络通信的时延会大幅增加。一方面，针对这个问题，华为在k8S社区的主导实现了节点亲和反亲和调度、应用间亲和反亲和调度、Taints tolerations等高级调度机制。通过给Pod配置高级调度策略干预应用组件的分布，配合容器网络在路由策略上的优化，访问时延问题得到了显著的改善。

另一方面，在集群规模和性能方面，华为也做了很多探索与实践。面对大规模场景下海量Service的管理性能问题，华为设计实现并向社区贡献了使用IPVS管理service路由规则的方案，将k8S对service的管理规模从上千提升到了数万的级别。

1.2 华为云应用服务与 Kubernetes

华为云应用服务产品均围绕着“容器”为中心构建，致力于帮助客户容器化的应用在云上高效地开发、交付与运维，并保障应用运行时的高性能、高可靠、高弹性。目前，华为云应用服务产品以基于K8S的华为云容器引擎（CCE）为核心，协同补齐了完整的应用开发、交付与运维流程，为客户提供完整的一站式云上应用生命周期管理方案。

华为云应用服务大体上可以分为三大类。

第一类围绕着Kubernetes核心功能，也就是容器编排与调度，与下层的基础设施层包括计算、网络、存储，以及水平的权限控制、网络防护、镜像仓库等服务进行整合形成一个容器化基础设施平台，并向上对接到集群管理、多DC/AZ、多区域管理实现云上的水平弹性。

通常大家所提到的“容器服务”或“容器云”大部分都是指这一类服务。华为云所提供的云容器引擎（CCE）、云容器实例（CCI）归属于此类。两者均基于Kubernetes构建，但技术路线偏重点有所区分。

CCE的服务形态是用户专属的Kubernetes集群（Kubernetes as a Service），用户能够控制整个Kubernetes集群的资源与应用，并且可以调用完整的Kubernetes API，以及安装各类Addon以及自定义扩展比如调度器、工作负载控制器等；而CCI的服务形态是无服务器容器（Serverless Container），用户无需感知Kubernetes集群，通常只需要调用Kubernetes Workload API进行应用的管理即可，而把资源全部交由华为云进行自动调度与管理。

因此，通常CCE适合业务半托管的场景，即用户自身有一定运维能力，且业务场景需要用户手动做一些管控比如资源规划、弹性伸缩，甚至自定义一些平台特性以适配业务；相对而言CCI适合业务全托管的场景，即用户只需关注以容器形态所交付的应用本身，无需关注资源管控，甚至无需关注Kubernetes的相关原理。

第二类服务围绕着Kubernetes标准化接口以及结合具体场景的最佳实

践来构建完整的应用开发、交付与运维流程，实现云上的应用全生命周期管理。

华为云在开发阶段提供微服务开发框架帮助用户在产品开发中落地微服务架构实践，在交付阶段提供“从代码到容器镜像”的自动镜像构建服务，支持一键式部署到Kubernetes平台之上，实现持续交付，而最终业务上线运行之后的运维阶段除了基础的容器监控、日志、告警系统之外，同时提供了微服务治理引擎，以及应用性能管理用于故障在线辅助与自动定位。

具体举例而言。

- 华为云微服务引擎（CSE）提供了具备升降级、容错、熔断等完整服务治理能力的微服务框架，兼容Spring Cloud、Dubbo等开源接口，并与CCE深度整合，支持ServiceMesh，未来计划进一步与CNCF基金会各微服务相关项目，尤其是Istio生态相结合，提供最适合在K8S之上运行业务所使用的微服务开发框架
- 华为云应用编排服务（AOS）提供了以应用为中心的高层编排引擎，能够将K8S上运行的各种工作负载、各类资源对象整合管理，并提供了完善的版本与生命周期管理机制，便于客户以更高层的“应用”为对象进行日常交付与运维管理
- 华为云应用性能管理（APM）提供了丰富的各类运维工具，除了基础的监控、日志与告警，进一步面向故障定位与分析场景提供了应用全局性能拓扑展示与调用链跟踪等高级特性，使得运维人员能够及时了解应用健康状态并进行相关处理。

第三类则是直接在Kubernetes之上身体力行地构建一些典型服务化应用，针对某些业务场景提供更易用、更高效的服务，使得客户更聚焦自身业务逻辑。

比如：

- 以分布式数据库（DDM）、分布式缓存（DCS）、分布式消息（DMS）为代表的云化中间件服务，供应用业务逻辑所调用，辅助

客户应用的容器化、无状态化、微服务化开发或改造；

- 以CCE/CCI为基础设施层所构建的Serverless Computing（FunctionStage）服务，面向Event-Driven的典型业务流场景简化应用代码逻辑，并基于容器热启动、各类主流语言运行时的快速启停优化，实现更高效、更低成本的实时计算；
- 区块链服务BCS则提供了主流的Hyperledger开源框架，并基于Kubernetes的高性能实现3分钟一键上链，2000+TPS的并发区块处理能力，可满足联盟链与私有链的各类业务场景诉求，使客户免运维地使用区块链构建自有业务框架。

1.3 未来：紧随社区版本 持续优势创新

纵观华为在Kubernetes上的创新可以总结为优势创新、场景创新、技术创新三个层面，优势创新是围绕华为固有的自身强势领域如网络、硬件进行与容器技术的结合运用。场景创新则是聚焦在不同领域的客户需求如游戏、电商、AI等，基于客户的计算需求进行解决方案的适配。技术创新，以无服务器容器为例，在Serverless的云服务趋势下，华为云提供更加便捷，更加全新理念的容器服务方式。

目前看来，容器服务并没有统一的服务标准，并没有说哪一种创新可以一招解决所有企业云上容器化的痛点，这需要根据客户的业务场景进行量身匹配，而华为云的全栈容器服务的实践案例也充分说明了这一点。众多不同的容器服务在上线不久已应用在众多不同领域，裸金属容器已成功运用在一部分游戏客户中，帮助其进行测试环境，及高峰时间的流量应对。Windows容器成功运用在传统IT系统的容器化改造，而无服务器容器则可以帮助更多缺乏Kubernetes技术投入的公司快速上手享受容器化带来的益处。

这也就是华为云对于Kubernetes的一些探索和思考，未来还会有更多基于容器的创新，一切才刚刚开始。

京东从OpenStack切换到Kubernetes的经验之谈

作者 鲍永成

背景介绍

2016年底，京东新一代容器引擎平台JDOS2.0上线，京东从OpenStack切换到Kubernetes。到目前为止，JDOS2.0集群2w+Pod稳定运行，业务按IDC分布分批迁移到新平台，目前已迁移20%，计划Q2全部切换到Kubernetes上，业务研发人员逐渐适应从基于自动部署上线切换到以镜像为中心的上线方式。JDOS2.0统一提供京东业务，大数据实时离线，机器学习（GPU）计算集群。从OpenStack切换到Kubernetes，这中间又有哪些经验值得借鉴呢？

本文将为读者介绍京东商城研发基础平台部如何从0到JDOS1.0再到

JDOS2.0的发展历程和经验总结，主要包括：

- 如何找准痛点作为基础平台系统业务切入点；
- 如何一边实践一边保持技术视野；
- 如何运维大规模容器平台；
- 如何把容器技术与软件定义数据中心结合。

集群建设历史

物理机时代（2004-2014）

在2014年之前，公司的应用直接部署在物理机上。在物理机时代，应用上线从申请资源到最终分配物理机时间平均为一周。应用混合部署在一起，没有隔离的应用混部难免互相影响。为减少负面影响，在混部的比例平均每台物理机低于9个不同应用的Tomcat实例，因此造成了物理机资源浪费严重，而且调度极不灵活。物理机失效导致的应用实例迁移时间以小时计，自动化的弹性伸缩也难于实现。为提升应用部署效率，公司开发了诸如编译打包、自动部署、日志收集、资源监控等多个配套工具系统。

容器化时代（2014-2016）

2014年第三季度，公司首席架构师刘海锋带领基础平台团队对于集群建设进行重新设计规划，Docker容器是主要的选型方案。当时Docker虽然已经逐渐兴起，但是功能略显单薄，而且缺乏生产环境，特别是大规模生产环境的实践。团队对于Docker进行了反复测试，特别是进行了大规模长时间的压力和稳定性测试。根据测试结果，对于Docker进行了定制开发，修复了Device Mapper导致crash、Linux内核等问题，并增加了外挂盘限速、容量管理、镜像构建层级合并等功能。

对于容器的集群管理，团队选择了OpenStack＋nova-docker的架构，用管理虚拟机的方式管理容器，并定义为京东第一代容器引擎平台JDOS1.0（JD DataCenter OS）。JDOS1.0的主要工作是实现了基础设施容器化，应用上线统一使用容器代替原来的物理机。

在应用的运维方面，兼用了之前的配套工具系统。研发上线申请计算资源由之前的一周缩短到分钟级，不管是1台容器还是1千台容器，在经过计算资源池化后可实现秒级供应。同时，应用容器之间的资源使用也得到了有效的隔离，平均部署应用密度提升3倍，物理机使用率提升3倍，带来极大的经济收益。

我们采用多IDC部署方式，使用统一的全局API开放对接到上线系统，支撑业务跨IDC部署。单个OpenStack集群最大是1万台物理计算节点，最小是4K台计算节点，第一代容器引擎平台成功地支撑了2015和2016年的618和双十一的促销活动。至2016年11月，已经有15W+的容器在稳定运行。

在完成的第一代容器引擎落地实践中，团队推动了业务从物理机上迁移到容器中来。在JDOS1.0中，我们使用的IaaS的方式，即使用管理虚拟机的方式来管理容器，因此应用的部署仍然严重依赖于物理机时代的编译打包、自动部署等工具系统。但是JDOS1.0的实践是非常有意义的，其意义在于完成了业务应用的容器化，将容器的网络、存储都逐渐磨合成熟，而这些都为我们后面基于1.0的经验，开发一个全新的应用容器引擎打下了坚实的基础。

新一代应用容器引擎（JDOS2.0）

1.0的痛点

JDOS1.0解决了应用容器化的问题，但是依然存在很多不足。

首先是编译打包、自动部署等工具脱胎于物理机时代，与容器的开箱即用理念格格不入，容器启动之后仍然需要配套工具系统为其分发配置、部署应用等等，应用启动的速度受到了制约。

其次，线上线下环境仍然存在不一致的情况，应用运行的操作环境，依赖的软件栈在线下自测时仍然需要进行单独搭建。线上线下环境不一致也造成了一些线上问题难于在线下复现，更无法达到镜像的“一次构建，

功能	选型
容器工具	Docker
容器网络	Cane
容器引擎	Kubernetes
镜像中心	Harbor
持续集成工具	Jenkins
日志管理	Logstash + Elastic Search
监控管理	Prometheus

图 1

随处运行”的理想状态。

再次，容器的体量太重，应用需要依赖工具系统进行部署，导致业务的迁移仍然需要工具系统人工运维去实现，难以在通用的平台层实现灵活的扩容缩容与高可用。

另外，容器的调度方式较为单一，只能简单根据物理机剩余资源是否满足要求来进行筛选调度，在提升应用的性能和平台的使用率方面存在天花板，无法做更进一步提升。

平台架构

鉴于以上不足，在当JDOS1.0从一千、两千的容器规模，逐渐增长到六万、十万的规模时，我们就已经启动了新一代容器引擎平台(JDOS 2.0)研发。JDOS 2.0的目标不仅仅是一个基础设施的管理平台，更是一个直面应用的容器引擎。JDOS 2.0在原1.0的基础上，围绕Kubernetes，整合了JDOS 1.0的存储、网络，打通了从源码到镜像，再到上线部署的CI/CD全

流程，提供从日志、监控、排障、终端、编排等一站式的功能。JDOS 2.0的平台架构如图1所示。

在JDOS 2.0中，我们定义了系统与应用两个级别。一个系统包含若干个应用，一个应用包含若干个提供相同服务的容器实例。一般来说，一个大的部门可以申请一个或者多个系统，系统级别直接对应于Kubernetes中的namespace，同一个系统下的所有容器实例会在同一个Kubernetes的namespace中。应用不仅仅提供了容器实例数量的管理，还包括版本管理、域名解析、负载均衡、配置文件等服务。

不仅仅是公司各个业务的应用，大部分的JDOS 2.0组件(Gitlab/Jenkins/Harbor/Logstash/Elastic Search/Prometheus)也实现了容器化，在Kubernetes平台上进行部署。

开发者一站式解决方案

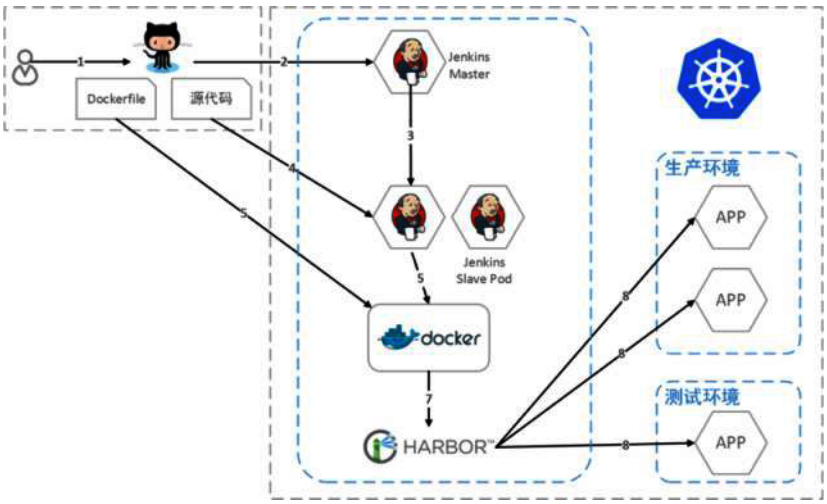


图 2

JDOS 2.0实现了以镜像为核心的持续集成和持续部署（见图2）。

- 开发者提交代码到源码管理库
- 触发Jenkins Master生成构建任务
- Jenkins Master使用Kubernetes生成Jenkins Slave Pod。

- Jenkins Slave拉取源码进行编译打包
- 将打包好的文件和Dockerfile发送到构建节点
- 在构建节点中构建生成镜像
- 将镜像推送到镜像中心Harbor
- 根据需要在不同环境生产/更新应用容器

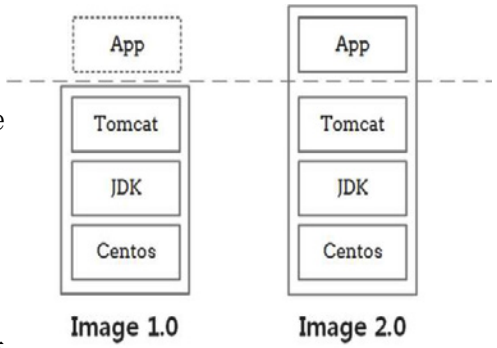


图 3

在JDOS 1.0，容器的镜像主要包含了操作系统和应用的运行时软件栈。APP的部署仍然依赖于以往运维的自动部署等工具。在2.0中，我们将应用的部署在镜像的构建过程中完成，镜像包含了APP在内的完整软件栈，真正实现了开箱即用（见图3)。

网络与外部服务负载均衡

JDOS 2.0继承了JDOS 1.0的方案，采用OpenStack-Neutron的VLAN模式，该方案实现了容器之间的高效通信，非常适合公司内部的集群环境。每个Pod占用Neutron中的一个port，拥有独立的IP。基于CNI标准，我们开发了新的项目Cane，用于将Kubelet和Neutron集成起来。

同时，Cane负责Kubernetes中service中的LoadBalancer的创建。当有LoadBalancer类型的service创建/删除/修改时，Cane将对应的调用Neutron中创建/删除/修改LBaaS的服务接口，从而实现外部服务负载均衡的管理。另外，Cane项目中的Hades(https://github.com/ipdcode/hades京东开源在GitHub上)组件为容器提供了内部的DNS解析服务。

灵活调度

JDOS 2.0接入了包括大数据、Web应用、深度学习等多种类型的应用，并为每种应用根据类型采用了不同的资源限制方式，并打上了Kubernetes的不同标签。基于多样的标签，我们实现了更为多样和灵活的调度方式，并在部分IDC实验性地混合部署了在线任务和离线任务。相较

于1.0，整体资源利用率提升了约30%（见图4）。

图 4

推广与展望

有了1.0的大规模稳定运营作为基础，业务对于使用容器已经给予了相当的信任和支持，但是平台化的容器和基础设施化的容器对于应用的要求也不尽相同。比如，平台化的应用容器IP并不是固定的，因为当一个容器失效，平台会自动启动另一个容器来替代，新的容器IP可能与原IP不同。这就要求服务发现不能再以容器IP作为主要标识，而是需要采用域名，负载均衡或者服务自注册等方式。

因此，在JDOS2.0推广过程中，我们也推动了业务方主要关注应用服务，减少对单个容器等细节的操作，以此自研了全新智能域名解析服务和基于DPDK高性能负载均衡服务，与Kubernetes有效地配合支持。

近两年，随着大数据、人工智能等研发规模的扩大，消耗的计算资源也随之增大。因此，我们将大数据、深度学习等离线计算服务也迁移进入JDOS2.0。目前是主要采用单独划分区域的方式，各自的服务仍然使用相对独立的计算资源，但是已经纳入JDOS2.0平台进行统一管理，并通过机器学习方法，提升计算资源使用效率。

灵活的标签给予了集群调度无限的可能。未来我们将丰富调度算法，

并配以节能的相关技术，提高集群整体的ROI，从而为打造一个低能耗、高性能的绿色数据中心打下基础。

回望与总结

Kubernetes方案与OpenStack方案相比，架构更为简洁。OpenStack整体运营成本较高，因为牵涉多个项目，每个项目各自有多个不同的组件，组件之间通过RPC（一般使用MQ）进行通讯。为提高可用性和性能，还需要考虑各个组件的扩展和备份等。这些都加剧了整体方案的复杂性，问题的排查和定位难度也相应提升，对于运维人员的要求也相应提高。

与之相比，Kubernetes的组件较少，功能清晰。其核心理念（对于资源和任务的理解）、灵活的设计（标签）和声明式的API是对Google多年来Borg系统的最好总结，而其提供的丰富的功能，使得我们可以投入更多精力在平台的整个生态上，比如网络性能的提升、容器的精准调度上，而不是平台本身。尤其是，副本控制的功能受到了业务线上应用运维工程师的追捧，应用的扩容缩容和高可用实现了秒级完成。JDOS 2.0目前已经接入了约20%的应用，部署有2个集群，目前日常运行的容器有20000个，仍在逐步推广中。

真诚感谢Kubernetes社区和相关开源项目的贡献者，目前京东已经加入CNCF组织，并在社区排名达到TOP30。

作者简介

鲍永成，京东基础平台部技术总监，带领基础平台部集群技术团队从2014年上线京东容器引擎平台JDOS1.0到现在的JDOS2.0，作为坚实的统一计算运行平台承载京东全部业务稳定运行。目前主要工作方向是JDOS2.0研发和京东第一代软件定义数据中心建设。

微软Azure PaaS发展之路

作者 张婵

云计算通常包括IaaS, SaaS和PaaS三个层面，相较于已成气候的IaaS和SaaS，最近几年云计算领域的集中发力点在PaaS层面。微软作为全球老牌IT巨头，也是PaaS供应商的领导者。微软的PaaS之路是怎么一步步发展起来的？对此我们专访了微软Azure资深架构师Steven，来了解微软Azure PaaS的发展之路。

微软早期 PaaS 服务

Azure是微软基于云计算的操作系统，自2008年开始发展，2010年正式推出，主要目标是为开发者提供一个平台，帮助开发可运行在云服务器、数据中心、Web和PC上的应用程序，使云计算的开发者能使用微软全

球数据中心的储存、计算能力和网络基础服务。

Azure平台上现已包括30余种服务，早期的PaaS服务包括：

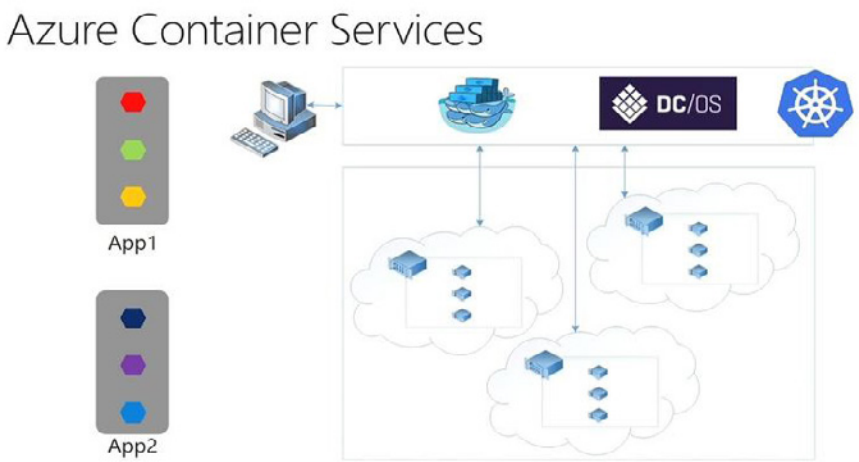
Azure Cloud Service：提供了抽象化的运算资源给云端应用程序使用，开发人员可以部署云端应用程序到Azure Cloud Service，以获取所需的执行环境与运算能力；

Azure Service Fabric：2016年正式GA发布，是基于微软资源管理框架与基础建设发展出的新型服务，提供标准的操作系统映像，开发人员可使用 Azure Service Fabric SDK开发微服务应用程序，可视为下一代的Azure Cloud Service。

随着容器技术的快速发展，微软也提供了重要的容器服务。

ACS：微软云端容器服务

2016年，微软推出了Azure Container Service (ACS)。ACS是一个容器托管环境，可用于容器的部署和管理，支持Docker Swarm, kubernetes, DC / OS等多种容器编排工具。虽然有了ACS全托管服务，但有些高级用户可能还希望做一些深度定制化的操作，获得完全的容器集群



控制以保证足够的灵活性（比如自定义master上的组件服务等）。这时用户可以使用开源的acs-engine来创建和管理自己的集群。acs-engine是 ACS 的核心部分，提供了部署和管理 Kubernetes，DC/OS 和Docker

Swarm集群的命令行工具。它通过将容器集群描述文件转化为一组ARM（Azure Resource Manager）模板来建立容器集群。

Steven介绍道，如果用户想要深度定制化自己的集群，比如加上虚拟网络的支持或一些高级组件，参数调整等等，就可以使用微软提供的acs-engine创建和管理自己的ARM部署模版，添加需要的脚本，参数，组件等定制化配置，然后使用微软提供的命令行工具实现自动化的部署。

微软与 Kubernetes

2015年CNCF基金会成立，微软加入其中成为其铂金会员，开始参与其中的项目向Kubernetes贡献代码。微软在CNCF里的贡献包括：

Draft，Azure团队的第一款开源容器管理工具，工具简化了所有在Kubernetes集群上运行的应用程序的开发工作，使用户能快速地进行容器化。

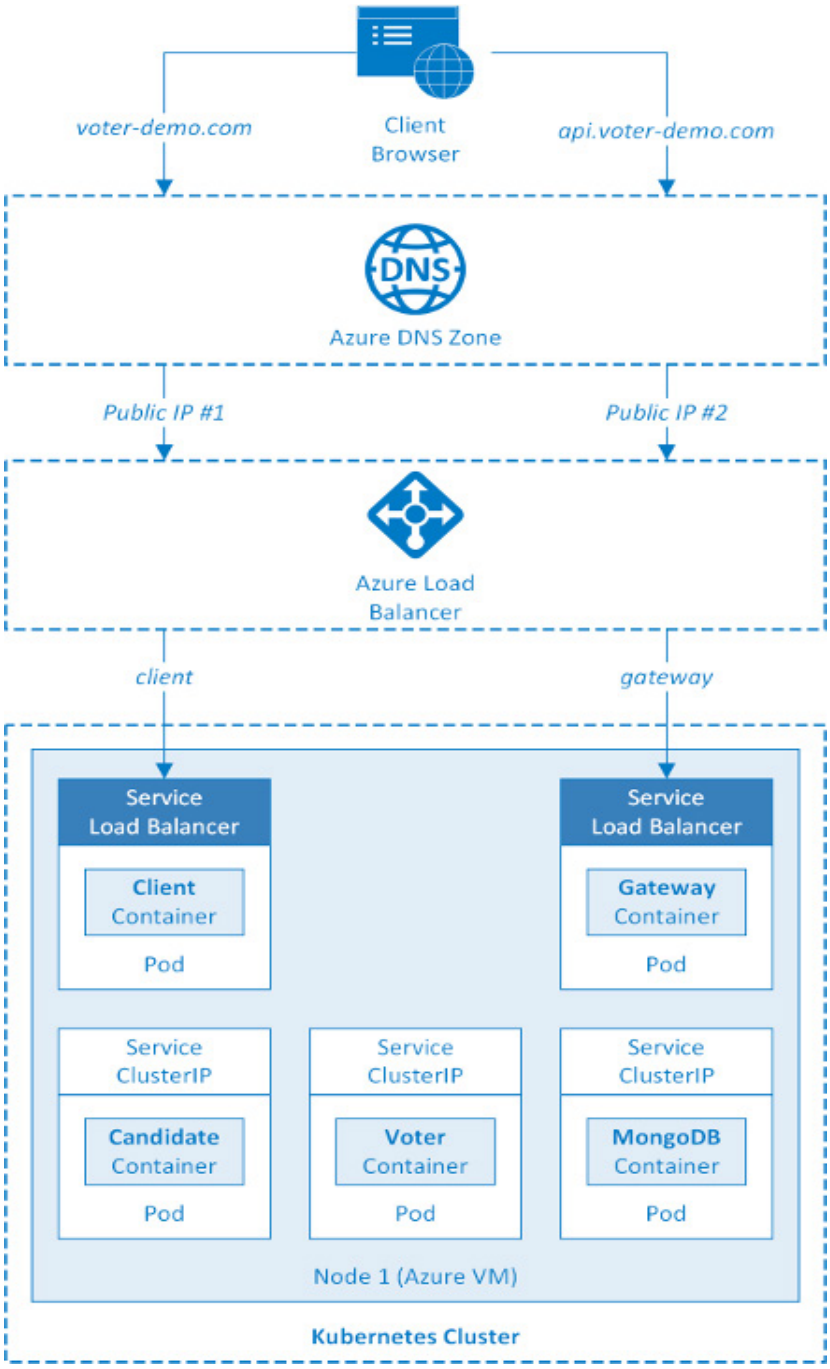
Helm，目前是Kubernetes服务编排领域的唯一开源子项目，做为Kubernetes应用的一个包管理工具，通过软件打包的形式，支持发布的版本管理和控制，很大程度上简化了Kubernetes应用部署和管理的复杂性。

除此之外还有OMS (Operations Management Suite)等监控项目。OMS是不开源的，可以用于监控Kubernetes，进行日志分析管理。

除了技术贡献，微软也对Kubernetes技术进行了人才投入，请了很多业界大牛来负责微软整个平台的容器产品，包括原Deis公司的创始人兼CTO Gabe Monroy，以及谷歌的前首席架构师Brendan Burns。Gabe Monroy是Docker和Kubernetes的早期贡献者，在容器技术方面有着丰富的经验；而Brendan Burns则是Kubernetes的首席工程师，Kubernetes容器编排的主要创始人之一。

AKS：主打 Kubernetes 的容器服务

2017年是Kubernetes快速发展的一年，Kubernetes作为开源容器编排标准正在日益崛起。微软也看到，每年基于Azure的部署Kubernetes的



需求量差不多以300%的速度在增长，所以就在Kubernetes的基础上新出了一个独立于ACS的服务，在2017年初推出了Azure Kubernetes Service (AKS)。

AKS是一个托管的Kubernetes服务 （目前还在预览阶段），可进行健

康监控和维护，支持自动升级和自动故障修复。AKS消除了用户管理和维护Kubernetes集群的负担，使用AKS时集群管理本身是免费的，Azure只收取容器底层的虚拟机费用。

另外，微软把AKS容器服务和自己的容器注册表服务（Azure Container Registry，私有容器镜像仓库，类似于Docker Hub），还有其他的PaaS服务做了无缝整合，用户可以把自己构建的容器放在容器注册表里面，然后来做灰度测试和CI/CD，做完之后通过命令行直接发布到生产环境。

Steven说AKS使用起来非常简单。比如在Mac上，AKS提供Azure Command Line的工具，只需三五行的命令就可以快速部署一个Kubernetes集群，并且在本机上进行管理。创建、伸缩、升级集群都可以在本机上进行，非常符合开发者的使用习惯。

Steven介绍到，不管是ACS还是AKS，微软在Azure平台上提供基于开源产品服务的一个最基本原则就是要保证这些服务和开源产品100%兼容。这也就意味着用户不需要重新学就可以把原来使用的工具，方法直接应用到微软提供的容器服务上面，微软只不过是把一些琐碎的工作做了自动化，在内部做了一些安全还有性能的优化，加固了整个容器平台。

随着技术的不断演进，Serverless（无服务器）架构现在被越来越多的提及。Serverless与PaaS的概念在某些方面有很多相似的地方，但这两者之间有一些微妙的差别：PaaS托管的是整个应用，而Serverless关注应用的某个碎片化的逻辑代码，从资源使用率和成本控制来说更具优势。针对Serverless，微软Azure又推出了Azure Functions。

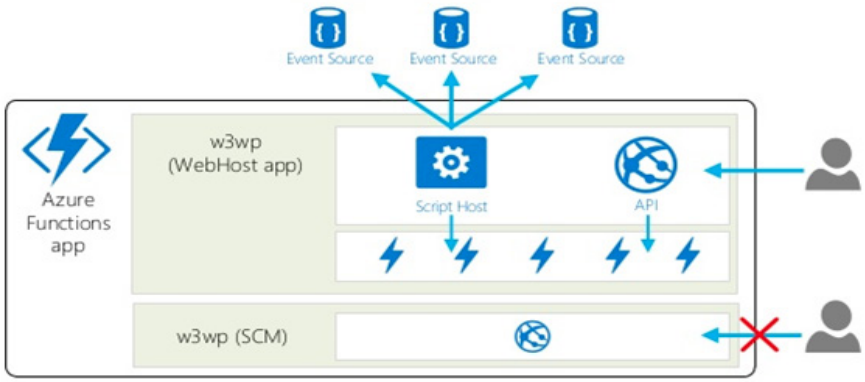
Azure Functions：微软 Serverless 平台

2016年底，微软正式发布了Azure Functions。Azure Functions为开发人员提供了一个事件驱动的无服务器计算平台，可以实现按需缩放，允许开发人员在不用接触与管理服务器的情况下，编写小型的处理程式以处理云端上的讯息或事件。

与传统的PaaS相比，使用Azure Functions无需关心运行时版本的问题。因为Serverless做发布的时候可以做定义或处理，能在非常短的时间内发布到云端，之后的事情都由后台进程来帮助做处理。

此外，serverless本身倾向于把服务封装成无状态的，其运行方式是按需运行，仅在设定的事件触发器上有事件产生时才运行，运行也是受调度平台的控制。

Azure Functions现在已经和微软的一些容器服务还有物联网的边缘计算相结合了。未来Azure Functions会应用在包括实时响应在内的更广泛的服务处理中，也将支持更多的开发语言。现在Azure Functions能支持Java，C#，Node.js，以后Python，Ruby等语言可能也会得到支持。



Azure PaaS 服务的未来

基于Kubernetes的PaaS在Azure内部受到高度重视，有社区大牛的掌舵，我们相信Azure将打造一流的PaaS服务。在对未来的看法上，Steven认为，容器化将是未来的主流，而微软的Azure将在PaaS的配套服务上投入更多精力。

Steven说，你看过去几十年基础架构的演变，从大型机到小型机，到X86架构，然后是虚拟化，容器化到Serverless计算，成本越来越低，部署速度越来越快，scalability也越来越强。现在很多大型企业的应用，包括Azure自身的一些服务，都已经转向了容器化，也就是

Docker+Kubernetes组合，这是未来的一个大趋势，会用得越来越广泛。

“现在可能70%的应用还在VM上面，已经有约30%慢慢转向了容器化的方式在容器里面运行。随着时间的推移，慢慢这个数据可能会变成70%以上都是容器化的Docker+Kubernetes的方式，30%是在VM里面。”Steven说道。虽然现在也有物理机，但是现在VM已经变成了主流；在未来，VM也会使用，但是容器化将是主流。

在转向容器化的过程中，Kubernetes作为容器编排事实标准，未来在它之上做的更多的事情就是把现在它所涉及到的存储，网络，服务的交互，发现，架构等各个方面做到高效的统一，减少不同版本不同语言之间的异构化差异，让部署变得更容易。有了一整套规范之后，就可以开发更规范的插件，大家就能更容易地切入到这个方向上来。例如微软自己也为Azure上的容器服务开发了针对Kubernetes和Docker Swarm的网络插件，让用户在使用Azure容器服务时可以利用到Azure SDN功能简化容器网络管理。

当所有的规范都形成后，开发人员上手的难度降下来，就会如Steven上面所说，70%以上的应用都会容器化。

未来，微软会一直和开源产品保持一致，跟开源社区保持紧密合作，在Azure上面为这种趋势提供最好的技术架构平台。现在微软大部分的服务都已经转向了微服务的架构。微软选择Kubernetes作为长期的容器平台发展方式，构建了很多周边的项目，比如ACR，Draft，Helm，FaaS，在AKS上提供这样的服务给用户使用，这样用户就不用担心被锁定，也不用担心灵活性和开放性。

作者简介

连冠华（Steven Lian），现任微软Azure资深架构师，主要负责为制造业、汽车、金融及互联网等行业大客户提供大规模云应用、分布式计算、车联网、开源DevOps、人工智能及大数据等架构设计和咨询服务。

你知道吗？传统企业已经在用最新互联网架构了

作者 李亚琼

“现在的 PaaS 已经跟传统 PaaS 完全不一样了；甚至可以说，有了容器 PaaS，很多场景下都不需要 IaaS。”企业私有云提供商博云 CTO 李亚琼博士对此一点都不惊讶。日常工作中他接触了太多对互联网业务需求强烈的传统企业，以一家汽车制造企业、一家能源电力企业为例，采用的都是“物理机 + 容器”的方式来部署私有云，上云的业务既有内部管理系统（OA），也有核心业务系统（支付网关）。可以说这些客户倒逼云计算供应商提供出 Docker+Kubernetes 的容器 PaaS 解决方案。

李亚琼博士的这番话的背景，是 Kubernetes 成为事实标准，在相关领域攻城略地，而基于 Docker+Kubernetes 的集群开始成为私有云的标准解法，不但互联网公司纷纷从 OpenStack 迁移到 Kubernetes，连博云

这样的私有云提供商也将 Kubernetes 的 PaaS 作为他们的主打产品。

容器 PaaS 是风口吗?

有关传统 PaaS 与容器 PaaS 的对比，业界有非常多的观点。

PaaS 一度是云计算的宠儿，从这个概念诞生开始，就有人认为它才是云计算的未来：使用 PaaS 就像用水和电一样，无关系统和环境，无需运维。大量的公有云厂商都瞄准了这个方向推出产品，甚至很多公司最开始推出的云计算产品就是 PaaS，最典型的就是新浪云 SAE。

但人们很快发现，传统 PaaS 的局限性太大，受限的运行环境、被阉割的 API，彼时开发 PaaS 上的应用就意味着和这个 PaaS 强绑定，很难迁移。甚至连开发都很有问题，因为你的开发环境和运行环境差别太大。因此 PaaS 逐渐偃旗息鼓，IaaS 作为更务实的选择，成为公有云的宠儿。

但这个情况被 Docker 以及之后的 Kubernetes 所改变了。

李亚琼博士认为，以前的 PaaS 面对的是开发，但其实无法满足开发者个性化的需求；容器 PaaS 关注的是部署和分发，不去干涉应用的运行时，反而给了开发更大的空间。

在传统 PaaS 平台结构中，更多是做一个应用沙盒，封装了应用正常运行所需的运行环境和系统，这类 PaaS 平台就如同一个“黑盒”，应用完全脱离了用户的控制，进入了完全被托管的状态，使得开发人员和运维人员对应用和应用运行时的掌控力变弱；另外传统 PaaS 通常在应用架构选择、支持的环境服务等方面有较强限制，导致此类云平台层次结构运力不足。

随着容器的兴起，传统 PaaS 一方面向更高级的 Serverless 转变，另一方面又分裂出 iPaaS，也就是容器 PaaS，可做应用层的封装调度、部署打包、开发扩容，很多互联网厂商和红帽这样的传统厂商都在转向以容器为核心的 PaaS。

对于客户来说，通过容器 PaaS 可以更加快速的实现业务开发、集成

和交付上线；另外它还具有无绑定、可拓展的特点。

当然，用户在选择哪些业务采用容器 PaaS 的时候更多的是考虑业务本身的流量是否具有突然性增长，而和具体的业务领域关系不大。不过，大部分用户上云也是遵循着优先部署对弹性能力要求比较高的业务、其他业务逐步迁移这样一个策略。

PaaS 会取代 IaaS 吗?

在企业私有云建设中，曾经一度进化出了 IaaS+ 云平台的分层结构，但 IaaS 层不具备贴近应用的资源调度策略。基于容器的 iPaaS 在部署和分发上更方便，更多的代码、应用、服务能被复用，而 IaaS 不贴近业务，交付慢。

另外伴随 PaaS 与 DevOps 结合为企业做微服务化改造，真正让企业的系统和应用实现了横向扩展、弹性伸缩。相对而言，IaaS 一般只能做原样迁移，不改造架构，企业上云前遇到的问题在上云后还会遇到。

此外李亚琼博士还补充：“在企业公有云上，也有 PaaS 能力向 IaaS 层渗透的趋势，这就是行业专有云。”我们发现在计世资讯发布的《2016-2017 年中国金融云市场现状与发展趋势研究报告》中显示，一些大型金融企业牵头，在自身搭建金融私有云的同时将冗余的资源提供给特定的、有需求的、受限于资金、技术能力等方面的中小型金融企业，最终形成专供金融行业企业使用的金融专有云模式。

为什么传统企业在上云上不再传统?

Kubernetes 作为 Google 开源的项目，其面向互联网应用的基因是渗透到整个平台的设计理念里的，这一点与传统企业的业务互联网化需求是非常匹配的。其他的调度框架、更多的特性会聚焦于如何实现资源调度，当然这也很重要；而 Kubernetes 的特性是围绕互联网应用架构去设计开发。

这是它吸引博云这样的 PaaS 厂商投身其中的重要原因。李亚琼博士

介绍，博云从 2015 年开始做 PaaS 平台研发，接触 Kubernetes 以后，直观判断这才是未来容器平台的核心和方向，事实也证实了博云的判断是准确的。

无论是 Docker、Kubernetes，还是 DevOps、微服务，李亚琼博士以一个传统客户选择这些互联网架构的上云历程，来诠释个中原因。

“我们在能源行业有个客户，在去年部署容器 PaaS 和 DevOps 平台。其中，集成了禅道（项目和文档管理）、Gitlab（代码管理、Issue 跟踪）、Jenkins（持续集成）、容器平台（持续部署、升级），我们帮客户实现整个 DevOps 工具链的集成和自动化 Pipeline 构建，现在客户从开发到测试环境部署实现了一键式。当然，根据客户的安全管理规范，上生产环境还是要经过内部审批后才能实现部署发布，不过在内部的镜像库、部署文件同步等方面我们也帮客户完成了平台搭建，只要流程审批通过也能实现一键式生成环境发布，效果非常不错。”

选择容器 PaaS 需要对原有的业务重构，这是用户在采用容器 PaaS 时不利的方面，但传统企业只要真正拥抱互联网，就必须去使用与互联网相适应的新技术。而容器 PaaS 在改造传统业务时，可以较好的实现逐步过渡，分期上线，也可以打消传统企业的疑虑。

以博云的 PaaS 平台架构为例：

这个架构中核心是从下向上四个层次：微服务运行时层、服务治理层、服务编排与协同层和场景应用层。其中，微服务运行时层也就是基于 Docker 和 Kubernetes 研发的容器 PaaS 层，聚焦与利用容器构建应用运行环境；服务治理层是围绕微服务间的调用及服务治理构建的平台软件，这一块博云也已经实现了在券商行业的落地实施；服务编排与协同层是正在研究的领域；最后就是客户的业务层，在这一层博云也会通过一些平台级的中间件对客户应用提供支持。

据了解，目前博云的 PaaS 产品 BeyondBOC 1.7 版已经可以支持区块链应用部署和微服务集成，比如支持了京东的商用中间件集成，其四级租户体系满足多场景需求，具备多数据中心管理和应用运维能力。在使用 BeyondBOC 的用户中，有 70% 以上已经使用在生产环境中，既包括很多金融行业用户，也包括新华社和某大型国有石油企业。

PaaS 会成为云计算主流吗?

从去年以来，云原生理念被越来越多的人所接受，以 Kubernetes 为核心的云原生容器基金会 CNCF 也迎来众多企业的支持。基于容器和 Kubernetes 的 PaaS、微服务、Serverless 等一起构成了云原生应用的基础设施和架构，只要云原生的理念不断普及，容器 PaaS 就会取代过去以虚拟化为核心的 IaaS，成为云计算的主流。

采访后记

2017 年，Kubernetes 成为容器编排事实标准，对云计算的底层架构有着深远影响。特别是在过去不受重视的 PaaS 层，有了 k8s 加持之后能力大大加强，开始逐渐发威。而在云计算的实际使用中人们发现，SaaS 和 IaaS 都在发展 PaaS 以满足客户快速开发的需求。这也是我们策划本次选题的原因所在，希望探讨 PaaS 的技术演变以及发展趋势。

在此次与博云 CTO 李亚琼博士的沟通中，我们发现他们很多传统企业客户都在选择 Docker、Kubernetes 这样的互联网架构，真的是在全面

拥抱互联网。博云也从他们业务经验中，解答了我们有关 PaaS 会取代 IaaS 吗？为什么 PaaS 重新流行？PaaS 会成为云计算主流吗？等诸多疑问。

“所谓云计算，其实是能力即服务，将能力提供出来，PaaS 能更好地抽象并提供能力。”这可能也是博云不希望以任何技术标签来定义自己的原因，他们更加看重的是是否能够根据客户应用需求和场景，具备整合技术的能力。“技术会被迭代、淘汰，而客户需求永远不会过时，客户会教你做出什么样的产品。”

作者简介

李亚琼，博云 CTO。毕业于中国科学院计算技术研究所，获计算机系统结构工学博士学位。加入博云前，先后在华为、曙光进行云计算相关产品研发。李亚琼博士长期从事计算机体系结构、操作系统、虚拟化和容错计算等方面研究，特别是在虚拟化环境下的资源建模与事件分发技术、资源调度与任务管理技术、安全增强与可信环境构建技术等方向进行了大量技术研究与产品开发工作。先后参与国家高技术研究发展“863”计划项目 5 项、“核高基”重大专项 1 项、国家自然科学基金委项目 1 项、国家发改委支持项目 1 项，涵盖高性能计算机系统、高端容错计算机、安全可信、云计算等基础平台及学科前沿研究方向。

版权声明

InfoQ 中文站出品

架构师特刊：Kubernetes PaaS 冲击波

出版：北京极客邦科技有限公司
北京市朝阳区来广营容和路叶青大厦北园 5 层
欢迎共同参与 InfoQ 中文站的内容建设工作，包括原创投稿和翻译，请联系 editors@cn.infoq.com。
网 址：www.infoq.com.cn