03 | 预习篇 · 小鲸鱼大事记（三）：群雄并起

2018-08-30 张磊

03 | 预习篇 · 小鲸鱼大事记（三）：群雄并起
朗读人：张磊　10′41″ | 4.90M

你好，我是张磊。我今天分享的主题是：小鲸鱼大事记之群雄并起。

在上一篇文章中，我剖析了 Docker 项目迅速走红背后的技术与非技术原因，也介绍了 Docker 公司开启平台化战略的野心。可是，Docker 公司为什么在 Docker 项目已经取得巨大成功之后，却执意要重新走回那条已经让无数先驱们尘沙折戟的 PaaS 之路呢？

实际上，Docker 项目一日千里的发展势头，一直伴随着公司管理层和股东们的阵阵担忧。他们心里明白，虽然 Docker 项目备受追捧，但用户们最终要部署的，还是他们的网站、服务、数据库，甚至是云计算业务。

这就意味着，只有那些能够为用户提供平台层能力的工具，才会真正成为开发者们关心和愿意付费的产品。而 Docker 项目这样一个只能用来创建和启停容器的小工具，最终只能充当这些平台项目的“幕后英雄”。

而谈到 Docker 项目的定位问题，就不得不说说 Docker 公司的老朋友和老对手 CoreOS 了。

CoreOS 是一个基础设施领域创业公司。 它的核心产品是一个定制化的操作系统，用户可以按照分布式集群的方式，管理所有安装了这个操作系统的节点。从而，用户在集群里部署和管理应用就像使用单机一样方便了。

Docker 项目发布后，CoreOS 公司很快就认识到可以把“容器”的概念无缝集成到自己的这套方案中，从而为用户提供更高层次的 PaaS 能力。所以，CoreOS 很早就成了 Docker 项目的贡献者，并在短时间内成为了 Docker 项目中第二重要的力量。

然而，这段短暂的蜜月期到 2014 年底就草草结束了。CoreOS 公司以强烈的措辞宣布与 Docker 公司停止合作，并直接推出了自己研制的 Rocket（后来叫 rkt）容器。

这次决裂的根本原因，正是源于 Docker 公司对 Docker 项目定位的不满足。Docker 公司解决这种不满足的方法就是，让 Docker 项目提供更多的平台层能力，即向 PaaS 项目进化。而这，显然与 CoreOS 公司的核心产品和战略发生了严重冲突。

也就是说，Docker 公司在 2014 年就已经定好了平台化的发展方向，并且绝对不会跟 CoreOS 在平台层面开展任何合作。这样看来，Docker 公司在 2014 年 12 月的 DockerCon 上发布 Swarm 的举动，也就一点都不突然了。

相较于 CoreOS 是依托于一系列开源项目（比如 Container Linux 操作系统、Fleet 作业调度工具、systemd 进程管理和 rkt 容器），一层层搭建起来的平台产品，Swarm 项目则是以一个完整的整体来对外提供集群管理功能。而 Swarm 的最大亮点，则是它完全使用 Docker 项目原本的容器管理 API 来完成集群管理，比如：

- 单机 Docker 项目：

```
$ docker run " 我的容器
```

- 多机 Docker 项目：

```
$ docker run -H " 我的 Swarm 集群 API 地址 " " 我的容器 "
```

所以在部署了 Swarm 的多机环境下，用户只需要使用原先的 Docker 指令创建一个容器，这个请求就会被 Swarm 拦截下来处理，然后通过具体的调度算法找到一个合适的 Docker Daemon 运行起来。

这个操作方式简洁明了，对于已经了解过 Docker 命令行的开发者们也很容易掌握。所以，这样一个“原生”的 Docker 容器集群管理项目一经发布，就受到了已有 Docker 用户群的热捧。而

相比之下，CoreOS 的解决方案就显得非常另类，更不用说用户还要去接受完全让人摸不着头脑、新造的容器项目 rkt 了。

当然，Swarm 项目只是 Docker 公司重新定义“PaaS”的关键一环而已。在 2014 年到 2015 年这段时间里，Docker 项目的迅速走红催生出了一个非常繁荣的“Docker 生态”。在这个生态里，围绕着 Docker 在各个层次进行集成和创新的项目层出不穷。

而此时已经大红大紫到“不差钱”的Docker 公司，开始及时地借助这波浪潮通过并购来完善自己的平台层能力。其中一个最成功的案例，莫过于对 Fig 项目的收购。

要知道，Fig 项目基本上只是靠两个人全职开发和维护的，可它却是当时 GitHub 上热度堪比 Docker 项目的明星。

Fig 项目之所以受欢迎，在于它在开发者面前第一次提出了“容器编排”（Container Orchestration）的概念。

其实，“编排”（Orchestration）在云计算行业里不算是新词汇，它主要是指用户如何通过某些工具或者配置来完成一组虚拟机以及关联资源的定义、配置、创建、删除等工作，然后由云计算平台按照这些指定的逻辑来完成的过程。

而容器时代，“编排”显然就是对 Docker 容器的一系列定义、配置和创建动作的管理。而 Fig 的工作实际上非常简单：假如现在用户需要部署的是应用容器 A、数据库容器 B、负载均衡容器 C，那么 Fig 就允许用户把 A、B、C 三个容器定义在一个配置文件中，并且可以指定它们之间的关联关系，比如容器 A 需要访问数据库容器 B。

接下来，你只需要执行一条非常简单的指令：

```
$ fig up
```

Fig 就会把这些容器的定义和配置交给 Docker API 按照访问逻辑依次创建，你的一系列容器就都启动了；而容器 A 与 B 之间的关联关系，也会交给 Docker 的 Link 功能通过写入 hosts 文件的方式进行配置。更重要的是，你还可以在 Fig 的配置文件里定义各种容器的副本个数等编排参数，再加上 Swarm 的集群管理能力，一个活脱脱的 PaaS 呼之欲出。

Fig 项目被收购后改名为 Compose，它成了 Docker 公司到目前为止第二大受欢迎的项目，一直到今天也依然被很多人使用。

当时的这个容器生态里，还有很多令人眼前一亮的开源项目或公司。比如，专门负责处理容器网络的 SocketPlane 项目（后来被 Docker 公司收购），专门负责处理容器存储的 Flocker 项目

（后来被 EMC 公司收购），专门给 Docker 集群做图形化管理界面和对外提供云服务的 Tutum 项目（后来被 Docker 公司收购）等等。

一时之间，整个后端和云计算领域的聪明才俊都汇集在了这个“小鲸鱼”的周围，为 Docker 生态的蓬勃发展献上了自己的智慧。

而除了这个异常繁荣的、围绕着 Docker 项目和公司的生态之外，还有一个势力在当时也是风头无两，这就是老牌集群管理项目 Mesos 和它背后的创业公司 Mesosphere。

Mesos 作为 Berkeley 主导的大数据套件之一，是大数据火热时最受欢迎的资源管理项目，也是跟 Yarn 项目杀得难舍难分的实力派选手。

不过，大数据所关注的计算密集型离线业务，其实并不像常规的 Web 服务那样适合用容器进行托管和扩容，也没有对应用打包的强烈需求，所以 Hadoop、Spark 等项目到现在也没在容器技术上投下更大的赌注；但是对于 Mesos 来说，天生的两层调度机制让它非常容易从大数据领域抽身，转而去支持受众更加广泛的 PaaS 业务。

在这种思路的指导下，Mesosphere 公司发布了一个名为 Marathon 的项目，而这个项目很快就成为了 Docker Swarm 的一个有力竞争对手。

虽然不能提供像 Swarm 那样的原生 Docker API，Mesos 社区却拥有一个独特的竞争力：超大规模集群的管理经验。

早在几年前，Mesos 就已经通过了万台节点的验证，2014 年之后又被广泛使用在 eBay 等大型互联网公司的生产环境中。而这次通过 Marathon 实现了诸如应用托管和负载均衡的 PaaS 功能之后，Mesos+Marathon 的组合实际上进化成了一个高度成熟的 PaaS 项目，同时还能很好地支持大数据业务。

所以，在这波容器化浪潮中，Mesosphere 公司不失时机地提出了一个名叫“DC/OS”（数据中心操作系统）的口号和产品，旨在使用户能够像管理一台机器那样管理一个万级别的物理机集群，并且使用 Docker 容器在这个集群里自由地部署应用。而这，对很多大型企业来说具有着非同寻常的吸引力。

这时，如果你再去审视当时的容器技术生态，就不难发现 CoreOS 公司竟然显得有些尴尬了。它的 rkt 容器完全打不开局面，Fleet 集群管理项目更是少有人问津，CoreOS 完全被 Docker 公司压制了。

而处境同样不容乐观的似乎还有 RedHat，作为 Docker 项目早期的重要贡献者，RedHat 也是因为对 Docker 公司平台化战略不满而愤愤退出。但此时，它竟只剩下 OpenShift 这个跟 Cloud Foundry 同时代的经典 PaaS 一张牌可以打，跟 Docker Swarm 和转型后的 Mesos 完全不在同一个“竞技水平”之上。

那么，事实果真如此吗？

2014 年注定是一个神奇的年份。就在这一年的 6 月，基础设施领域的翘楚 Google 公司突然发力，正式宣告了一个名叫 Kubernetes 项目的诞生。而这个项目，不仅挽救了当时的 CoreOS 和 RedHat，还如同当年 Docker 项目的横空出世一样，再一次改变了整个容器市场的格局。

总结

我分享了 Docker 公司平台化战略的来龙去脉，阐述了 Docker Swarm 项目发布的意义和它背后的设计思想，介绍了 Fig（后来的 Compose）项目如何成为了继 Docker 之后最受瞩目的新星。

同时，我也和你一起回顾了 2014~2015 年间如火如荼的容器化浪潮里群雄并起的繁荣姿态。在这次生态大爆发中，Docker 公司和 Mesosphere 公司，依托自身优势率先占据了有利位置。

但是，更强大的挑战者们，即将在不久后纷至沓来。

思考题

你所在团队有没有在 2014~2015 年 Docker 热潮中，推出过相关的容器产品或者项目？现在结局如何呢？

欢迎你给我留言。

版权归极客邦科技所有，未经许可不得转载

精选留言

fhchina 1

想要了解后续的标准化过程中，Docker分拆出的开源产品与标准的关系与历程，感觉又多又乱

2018-08-30

猿工匠 0

过瘾，期待更新😁

2018-08-30

欧文雪 0

占个沙发

2018-08-30

Backkom 0

docker生态维护的很好，这也是发展迅猛的重要原因，得开发者得天下，一如android的发展

2018-08-30

尖端科技 0

张老师，请教一下docker是否合适用来部署使用gpu这种资源的程序？docker部署的程序是适用于任何的程序部署？有没有啥限制？

2018-08-30

成都福哥 0

太少了，不过瘾～～一口气看完了三篇。期待第四篇～～

2018-08-30