容器生态圈的围剿，Docker已经“众叛亲离”？

2016-12-09 周晖 高效开发运维

关于作者

周晖，Pivotal大中华区云计算首席架构师，有着丰富的PaaS云实际建设经验，负责过国内某知名银行已经生产上线一年的PaaS云的架构设计和部分功能的实现，参与过某超算PaaS、某超市电商PaaS、某电力PaaS等项目的建设。

从Google K8s布道师的诘问说起

事情追溯到今年7月底，GoogleKubernetes布道师 Kelsey Hightower 和Docker的CTO Solomon Hykes在Twitter上发生了激烈的争论，争论的主题是要不要用RunC或其他容器来取代Docker引擎以及OCI的意义。

两人前段对话回合大致如下：

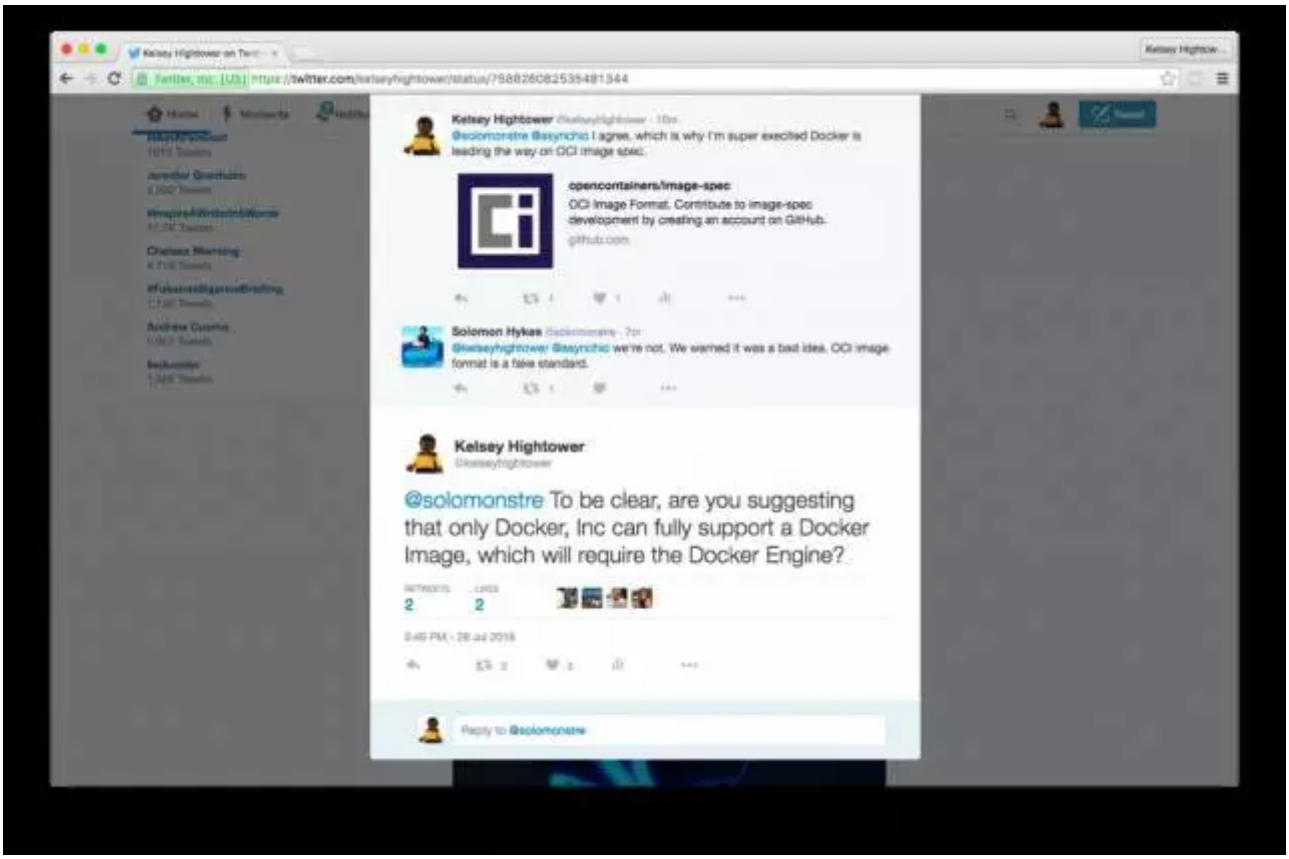
Kelsey：“很多平台都可以跑Docker镜像，已经不再需要DockerDaemon了。哪个会成功呢？”

Solomon：“其他平台跑Docker镜像是假的，其中只有90%能正常工作，其余10%则随时可能会出问题。而且Docker还在演进中。”“所以嘛，声称‘可以跑Docker镜像’的都是在撒谎。”

Kelsey：“好吧，那我们就没必要再提支持Docker了。我们实际支持的只是Docker的容器格式”，“Docker拥有创建和分发镜像的最佳工作流，而运行时，还是留给它的竞争者们吧。”

Solomon：“这些都是不完整的、不兼容的支持”，“他们也并不支持镜像格式，镜像的很多信息都会丢失”。

在强调完Docker的不兼容性之后，Solomon一怒说出“OCI就是个伪标准”。



此言一出，惊诧四方，因为OCI有50多家厂商参与，而且Docker还是重要贡献者，有说“标准不是一个人可以决定对错的”，有反讽的 “我相信工程师们乐于听到他们创始人说他们做的工作是假货”。

Kubernetes的老大Tim Hockin直接就说“不爽你就走，不要假装参与”。

Solomon不得不改口说“OCI标准的初衷是好的，只可惜扩展太早，我不认同就得走？”

有人随即反驳“只有你一个人说扩展的太早，这是明显的利益投射”。

Solomon还嘴硬“就因为我是唯一的不认同者，那我就必须离开？”

Kelsey随后则直点主题，“有两个问题在台面上，一是容器需不需要标准化，二是需不需要由Docker来领导这个标准化的工作？”。Kelsey再以嘲讽的口吻自问自答“如果是Docker来回答这些问题，对第一个肯定是说不，那干脆对第二个问题也同样说不吧，这可不是对和错的问题”。

因为事关OCI，Docker现在这么鄙视RunC，OCI不得点出Docker你也是参与者就别闹了，“Docker参与在OCI运行时和镜像规范，也参与了每周的开发会议”。

Kelsey乘胜追击，代表业界吐了个槽，“我一直相信Docker会给容器带来很多，但是我真的担心一个人想控制的太多”。Docker公司的控制欲早成为人们的吐槽点，什么都想

做，把整个生态圈其他的参与者逼到墙角去了。

看业界大佬撕逼，感觉和咋人民群众撕逼没太大差别。Solomon的撕逼技巧明显略逊一筹，有点像祥林嫂反复说“其他容器就是和Docker不兼容，包括RunC”，但是没说出个哪里不兼容的道道。

从表面上看，是容器标准之争；其实，这是因为整体生态环境存在着巨大的分歧，而分歧背后则涉及巨大的利益问题。

实则，容器生态发展遇到困难

我们来总结这次著名的撕逼事件的来龙去脉，这会成为容器界一个关键的历史转折事件。容器之争始实际上是很早之前就起源于Docker生态面临的问题。

第一难：开源项目Docker被注册成商标

Docker最早是Solomo所在公司dotCloud的内部项目，该项目于2013年开源发布，在获得社区的热烈响应之后，dotCloud公司就进入了圈地过程。圈地第一招就非常凶悍，把公司从dotCloud改名为Docker，然后注册Docker商标。

商标意味着当其他公司使用未经Docker社区许可的补丁、代码或软件包的时候时可能面临法律责任。这种情况下，一个公司可能因为补丁尚未经过Docker公司许可，抑或尚未被合并到项目中，而不能为客户提供技术支持。

这不只是法务上的圈地，对于很多采用Docker名字的，Docker公司给予了实际行动—口头警告（谁叫Docker是人家注册的名字呢，就好比weibo.com就是微博），包括你建一个群名不加修饰限定的叫Docker，或是你要写本书，不加修饰的叫DockerXXX，都容易受到Docker的警告。从法务上看这确实也属于侵权，就看Docker要不要告你。

这种情况下，第三方Docker生态厂商有两种选择：

- 被迫忍受Docker公司任何对该开源项目作出的决定，在问题出现的时候等待Docker公司的补丁；

- 另一种比较好的选择是与Docker达成某种协议，由Docker公司来保证提供稳定的发行版本。那Docker的生态公司就沦为Docker公司的代理商了。

第二难：Docker进军容器集群管理

Docker公司原来只在容器领域发展，Kubernetes/Mesos等做容器集群管理，属于CaaS（Container as a Service）：相互补充，互不竞争。

等Docker容器被广泛关注以后，Docker进入容器编排市场，收购了相关的技术以后推出Swarm的容器集群管理，从容器进入CaaS市场。2016年7月发布的Docker1.12把Swarm内置到Docker中去了，Docker Swarm作为容器集群管理软件，内置在Docker中，几乎就是Windows捆绑浏览器IE的模式，这就是用不平等的市场优势打击了Google的Kubernetes和Mesos等。这种做法和当年的Microsoft通过Windows捆绑IE来打击NetScape有异曲同工之妙。

Google也不是吃素的，所以7月底马上就是和Docker CTO口头开撕。

最关键问题在于：容器生态圈最重要的商业价值不在于容器技术本身，而在于容器集群管理。因为容器用于生产系统，进入企业运行环境就必须有容器的集群管理。（这也是为什么CaaS厂商没有自己做容器。）

Kubernetes和Mesos已经进入了这个商业领域并且取得了一定的优势。现在Docker进入这个领域，就直接和容器集群管理的公司竞争。但对于Docker公司而言，又必须进入这个领域，如果不进入这个领域，很难取得商业成功。Docker本身又经过多轮风投，风投给Docker带来了巨大的盈利压力：如果久久不能盈利，那风头的脸色就不会那么好看了。

可是，如果是公平竞争还好，比如Docker单独发布Swarm的Docker集群管理。但Docker直接把Swarm内置到Docker中去，安装部署Docker就带了Swarm，而Swarm的很多功能和Kubernetes和Mesos是重叠的，再用Kubernetes和Mesos的Docker不仅功能有大量重叠，而且带来了系统的复杂性和不稳定性。

那么，Docker都有哪些槽点呢？

1 向后兼容性问题

Docker被业界诟病最多的就是后向兼容性。Docker的新版本更关注的是革新性，而不是兼容性。但是，对于一个生产系统来说，后向兼容性至关重要：没有后向兼容性，意味着每次Docker升级都带来巨大的风险。

这也和Docker的企业文化有关，Docker更倾向于采纳新技术，实现突破性的功能。这是典型的初创公司的企业文化，不断的追加新功能，而不考虑企业级特性，更不考虑后向兼容的束缚。

这个好处是能在个人粉丝中取得共鸣，这也是Docker快速流行的一个重要原因。任何特征都可能是双刃剑：这种企业文化并不适合企业级，不考虑生产运行的可用性，对企业级应用来说可能是灾难性的。

那么业界人士对此是怎么评价的呢?

“Docker不断地破坏向后的兼容性”，RackN公司的CEO Rob Hirschfeld说。RackN公司的应用程序生命周期管理平台同时使用和提供了Docker组件，因此对后端的改动将会对其支持的客户的业务造成影响。“Docker将Docker Engine用作一个产品，而不是一个社区用来构建自有服务的组件”，Hirschfeld指责道。这种基于产品的策略意味着使用者被逼着在Docker发布新的革新特性或者合并新的组件（如Swarm）的时候去修复其破坏的向后兼容的问题。“尽管我们会使用这些发布的特性，然而这些改动会带来一系列与稳定性、版本、迁移和后端兼容相关的问题”。

Bob Wise，一名三星SDS云工程师，也同样呼吁Docker放慢其容器创新的脚步，该公司同样提供了基于云的容器相关支持服务。“如果你的团队在深度使用Kubernetes、Mesos或Cloud Foundry，你需要一个稳定、简单、无奇的容器实现方案，仅有最少的基本功能，由社区协商镜像的创建、命名和发布”，Wise的一篇博客中写道。“你需要使用每个都人都在使用的一个相同的、简单的、无奇的容器实现方案。作为一个社区，我们需要放缓对于基本构建组件的变更速度。唯有稳定性才能让构建其上的系统蓬勃发展。”

2 在企业级应用层面还有待提高

Docker虽然一直宣称ProductionReady，但是在实际的生产系统中同样受到不少诟病。

看看业界人士的说法：

Apcera技术产品经理Phillip Tribble在个人博客中，以一种外交辞令的口吻，让Docker不要再把其新推出的特性鼓吹成一个完工的企业级可用的产品。Tribble写道：“让互联网或者大会充斥着营销材料，宣扬各种令人振奋的新特性，但实际上却不如所说那样能用，不是一个明智的做法”，Apcera的商业模式一部分是基于提供Docker容器的支持。

Tribble的帖子引发了其他人在HackerNews上表达他们的Docker经历，包括一些新版本带来的让人伤心的bug。一位读者谈到一天中启动70,000到90,000个容器，约9%左右的会因为“各种Docker bug”遇到问题，这个比例最近的一次升级后下降到了4%。

但也有一些人在称赞Docker的稳定性，“我们一直在生产中使用Docker约3年了，还没有发现什么大的问题”，另一外读者评论说，“它将一些很炫的Linux内核特性用简洁的方式封装了起来”。

当然也有不少Docker的支持者认为Docker公司的软件是稳定的。同一个产品，在不同的客户有截然相反的反应说明有的用的好，有的用的碰到不少问题，在不同的环境下缺乏一致性，这也是企业生产系统的大忌。

3 捞过界了，越过了红帽操作系统界限

哪些功能应该由Docker来实现，哪些功能应该由底层操作系统或者技术栈中的其他组件来负责处理?

在今年的很多会议上，包括LinuxCon North America 2016，红帽工程师 Dan Walsh 说红帽陷入了困境：一方面客户越来越多的使用 systemd 来初始化Linux系统；而另一方面Docker似乎不愿使用systemd，取而代之使用Docker Daemon来提供初始化，服务激活，安全和容器日志的相关功能。

“在过去的三年里，我们一直试图使systemd和Docker更好的整合，而我却发现，两个领导人都非常强烈的坚持己见”，Walsh说，两个领导人指 Hykes和systemd的创造者-Lennart Poettering。

“所以，当机器的时候，谁来负责启动系统服务，是systemd还是Docker？”Walsh问到。一个拙劣的系统实现可能会导致systemd和Docker互相冲突。

红帽为自己的客户维护着自己的Docker版本分支，红帽分支中的补丁Docker有一天可能会合并，或者永远不。红帽也冒着巨大的风险，红帽的客户也冒着巨大的风险。所以红帽对Docker的支持其实远不如Ubuntu，因为Docker已经侵入到OS的领域了。

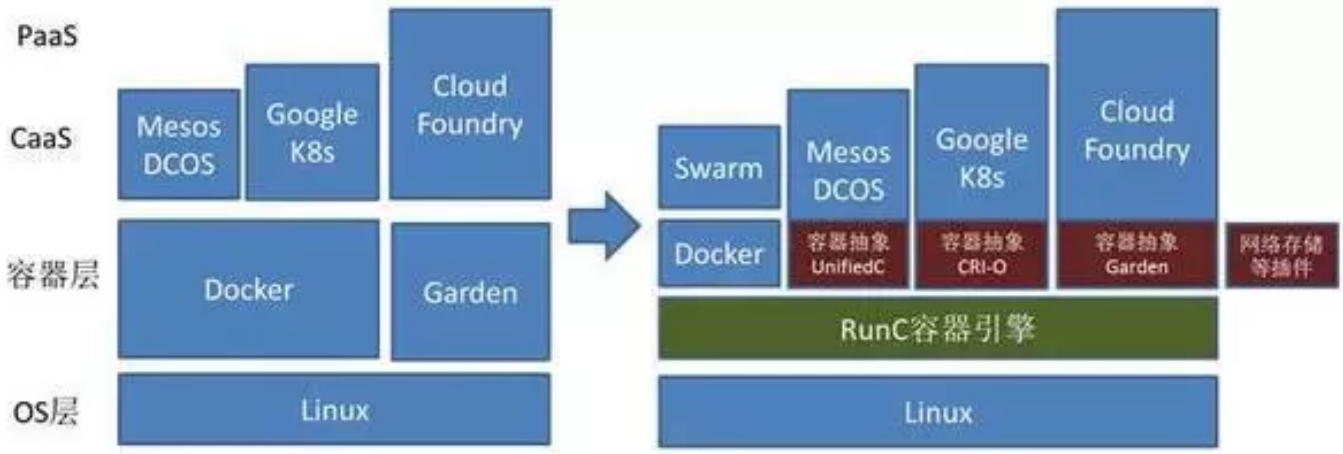
遗憾的是：即使是面对以上种种吐槽，Docker也不打算折中一下，甚至也不打算照顾这些意见，而是继续我行我素。

突围：拆解重组容器引擎，把RunC作核心

各方以RunC为核心重新构建容器生态圈，Docker容器被弱化

在对开源Docker分支进行了反复斟酌、放风声、试探和讨论之后，各方觉得杀伤力太大的方案。而重新回到了折中方案，以RunC为核心重新构建生态圈，并且通过插件来弱化容器在CaaS生态圈的重要性。

我们来看看生态圈的演进示意：



如上图，标识了容器生态圈或是CaaS的演进变化。

最早只有Docker和Garden两大主流容器，Mesos和Google都专注于CaaS，容器就全部采用Docker，CloudFoundry由于在Docker之前就推出了Warden(后升级到Garden)容器，CF采用自己的容器打造了PaaS平台，形成了一个和谐的生态。

在Docker捞过界了，并且确实有些不符合企业生产系统的因素，包括后向兼容性、商标问题、稳定性问题，于是各CaaS/PaaS生态厂商组建OCI联盟，打造RunC容器引擎，只需要一个简单的容器起停、管理等引擎，把Docker的容器功能一分为二，RunC作为一个简单明了的运行环境，降低复杂度，提升稳定性，适合生产系统。而对于Docker容器的其他功能，则在各自的容器抽象层，依据需要去实现，而且因为Docker本身集成了太多功能，不利于生产环境稳定性要求，各个容器抽象层都采用插件模式，维持容器的

简洁性，需要什么功能再插入容器，比如需要网络就可以插入网络插件，需要存储和卷访问，就插入存储和卷的插件。

目前的形势，就形成了Docker和各个CaaS/PaaS厂商在同一层面竞争，在CaaS/PaaS平台，Docker并没有什么优势，但是Docker想把其容器的广泛使用的优势在CaaS中延续，目前看来并不容易。容器的主要用户还是个人用户、开发者用户、运维用户，而CaaS是企业系统，二者目标客户不同、技术要求不同。

随着这个生态的演进，Docker容器会更多的用于开发、测试环境，而RunC在各个CaaS厂商的推动下会在生产环境得到广泛的应用。

K8s目前基本只支持RunC容器，对于Docker超出其容器抽象层之外的功能，一概不支持。同样，Mesos也通过其Unified Containerizer只支持RunC容器，目前还支持Docker，但是未来的规划是只支持UnifiedContainerizer。CF也通过Garden只支持RunC，不支持Docker超出RunC之外的功能。

RunC生态的快速发展

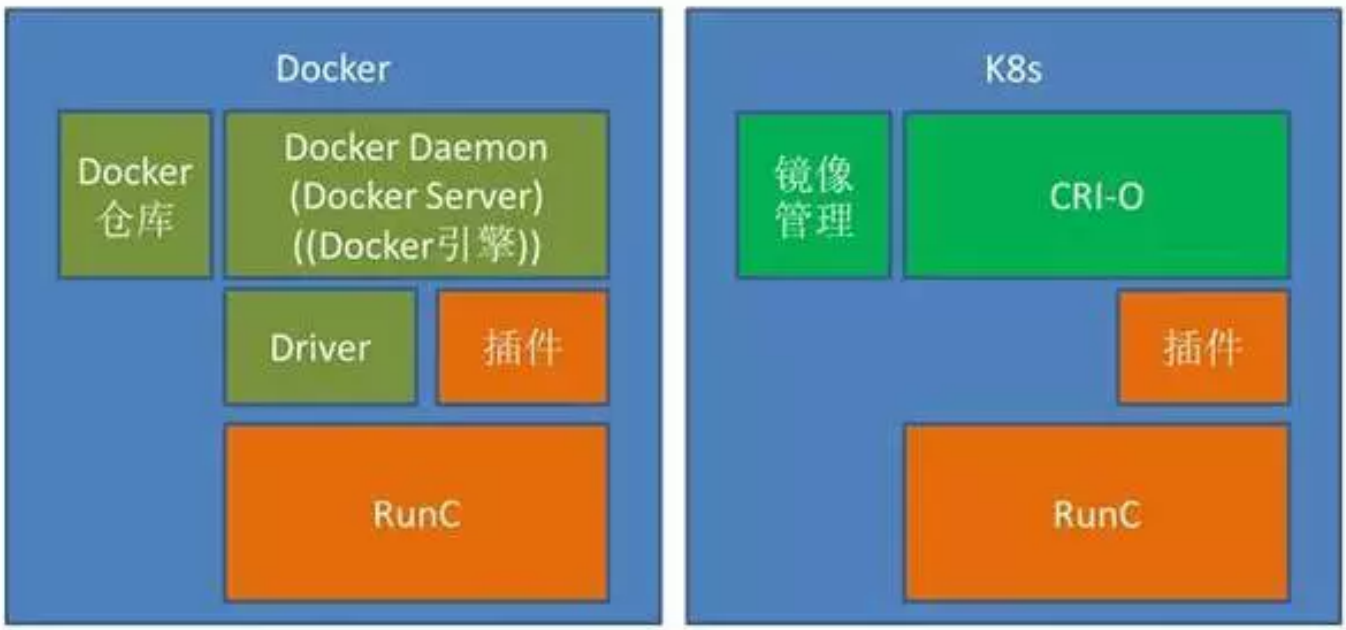
由于Docker的消极抵制，RunC的发展好像并不为人所知，但是RunC的发展还是很快的，RunC本身就简单，通过版本的持续的迭代更新，目前已经达到生产可用，而且主流的PaaS/CaaS纷纷采用。Docker也从1.11开始内置RunC容器运行时。

除了RunC本身的发展，RunC的生态圈也在快速发展，这个生态圈就脱离了的Docker。比如最近的Riddler，就是一个把Docker容器转换为RunC镜像。详见 https://github.com/jfrazelle/riddler。

详解CaaS界的几个方案

K8s通过CRI-O取代Docker容器管理引擎架构

和Cloud Foundry的架构模式类似，K8s也发展了CRI-O来取代Docker，架构图如下：



CRI-O是Google的Kelsey和Docker CTO所罗门论战之后的结果，论战之后，Google就提出一个设想，要让K8s调度的容器去Docker化，虽然他们一开始说的是要分支出一个

Docker的分支来做容器，但是后来考虑到这样做属于刺刀见红，杀伤力太大，所以在2016年6月先弄了一个OCID(OCI守护进程)，就是RunC的守护进程，和Docker Daemon有异曲同工之妙，该项目的维护人员此地无银三百两的说“这不是Docker的分支”。

由于OCID过于和DockerDaemon类似，随后Google又把这个项目重新命名为—CRI-O(ContainerRuntime Interface，容器运行时接口，O表示OCI，也就是RunC的运行时接口)，这也反映了Google的心态，一方面通过CRI对容器进行抽象，什么容器我都支持，另外加一个O，我重点支持OCI的RunC，显得不是那么白刀子进红刀子出，大家表面上还是和平共处，而且显得立意更高，通过容器抽象层进一步标准化容器，RunC只是标准化容器运行时，CRI把对容器的调用管理等也标准化，潜台词是DockerDaemon是非标准的、独家的。

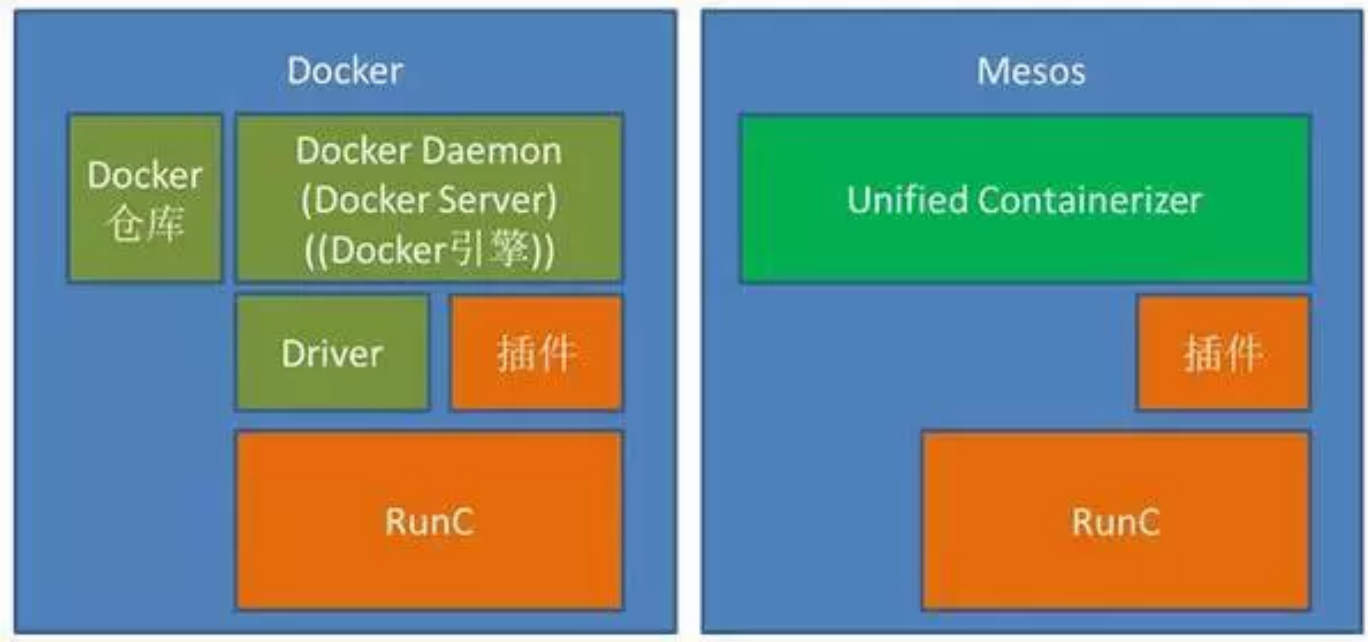
同时，Google也在向Mesos推销其CRI-O，希望Mesos也采用其CRI-O的架构。

CRI-O对容器运行时提供基本管理功能，同时Google的K8s提供镜像管理功能(Container/Images)，完全可以取代Docker的镜像仓库。K8s一方面支持容器插件技术，另一方面自己也制定实现一些容器插件，最典型的就是容器网络插件，自己定义并实现了CNM的容器网络插件。

因为K8s之前一直支持Docker，为了保持一定的兼容性，K8s继续支持Docker容器，但是不再支持Docker超出标准容器之外的特定功能，也就是把Docker的定位和RunC等同化，Docker做的再多功能也不用。

Mesos通过UnifiedContainer取代Docker容器管理引擎

和K8s类似，Mesos也不再只支持Docker容器，而且对容器进行了抽象，项目名字直接就叫”UnifiedContainerizer”—统一容器。目前还是支持 Docker 和Mesos Containerizer两种容器机制，未来就统一到”Unified Containerizer”。架构图如下：



Unified Containerizer也支持插件架构，但是和Docker的插件不是完全一样，设计的插件类型更丰富，包括三大类：

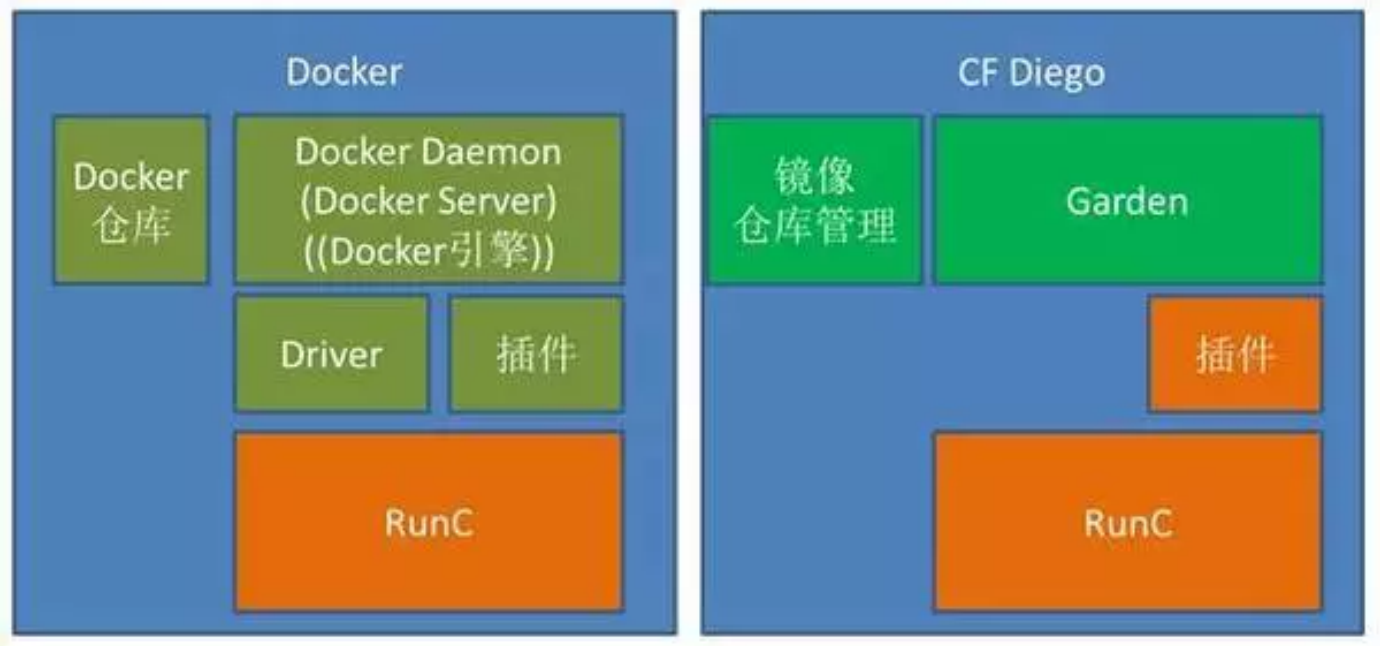
- 第一类是进程管理，支持容器之前的进程，这也是Mesos一贯的调度管理策略
- 第二类是隔离器: 在容器生命周期的各个阶段提供扩展接口，保护了Docker的几类插件，如网络、磁盘、文件系统、卷插件。
- 第三类是容器镜像管理，除了容器镜像，还将支持虚机镜像等。

Mesos的统一容器基本就包含了DockerDaemon、Docker仓库等功能。

当然，由于之前一直支持Docker容器，目前阶段Mesos还继续支持Docker，但是也有自己的Mesos Containerizer容器机制。

Cloud Foundry通过Garden取代Docker容器管理引擎

从RunC逐步成熟，CloudFoundry的容器引擎Garden就采用了RunC作为容器运行时，如下图：



Garden取代DockerDaemon(DockerDaemon有内有Docker Server，DockerServer内有Docker引擎)，直接调用RunC来生成容器运行环境，同时CFGarden也支持容器插件，容器插件是独立进程，在网络插件方面优先支持K8s的CMN插件标准。CF Diego有自己的镜像仓库管理，也可以从Docker仓库中获取Docker镜像部署。

不得不对Garden的设计多说几句，Garden包括之前的Warden，从一开始的设计就是容器抽象，使得可以支持不同的容器运行时，而且Garden做了三层抽象。所以Garden从一开始就支持.Net应用，不是通过Windows 2016的容器机制来实现，而是在.Net运行时模拟了一个容器的实现，所以Garden支持Windows的几乎所有版本的.Net应用。

K8s CRI-O和Mesos的UnifiedContainerizer都借鉴了Garden的容器抽象设计思路，所以Garden也是第一个支持RunC的CaaS/PaaS。

从这个架构可以看出，CloudFoundry的Garden基于RunC和容器插件，就替代了Docker的容器功能，共同的是RunC和容器插件，而Garden取代Docker Daemon的容器管理功能。

当然，Garden也支持直接部署Docker镜像。

那么，你还打算只用Docker吗？

Docker作为目前最热的容器开源项目，受到广泛的追捧。但是也要清醒地看到Docker和容器生态圈的种种争斗：Docker通过注册商标和在Docker中内嵌容器集群管理，挤压生态圈其他公司的生存空间，而受到生态圈联盟以RunC和相应的技术来制约Docker。

下面按照不同的使用情况，给出我的意见：

- 如果你是开发测试用Docker——那么基本不受影响、可以继续，这也是很多公司对Docker的定位。
- 如果你是生产系统采用Docker(包括Swarm)——你就要注意了，如果是你自己定制开发基于Docker/Swarm的CaaS，那问题也不大，出现漏洞或是定制可以自己打补丁；但是要意识到你的补丁不一定能合并到Docker的主干版本。
- 如果是你采用的是第三方定制——如果是基于Docker和Swarm的CaaS，你就一定要当心了，他们针对Docker做的定制要合并到Docker的后续版本有相当的难度：因为对于Docker的补丁定制合并，除了Docker公司其他公司几乎是没有什么控制力度的，还包括后向兼容性问题。

作为用户或是容器生态圈的创业公司，不能一棵树上吊死。

如果在容器层面只考虑Docker，而不考虑RunC，可能会和CaaS/PaaS生态圈的标准越来越远，未来和CaaS/PaaS的标准容器差异越来越大，主流的CaaS/PaaS厂商和技术，如K8s/Mesos/CloudFoundry均不再支持Docker容器超越RunC之外的功能，而只支持插件对RunC功能的扩展。

业界更普遍的定位是Docker用于开发测试环境，而RunC用于生产环境，所以对于要在生产环境采用容器技术的，一定要研究RunC。

很多容器创业公司是在Docker的风口成立的，由于Docker一家独大和Docker注册商标的法务问题，所以可能还没有在风口起飞。建议考虑在OCI/RunC的生态圈进行相关技术的发展，OCI/RunC的生态圈受到实力强大的几家公司的强力支持，如Google、CF基金会、Pivotal、Redhat、Mesos、CoreOS等。而且，RunC的生态圈还刚刚起步，还有很大的发展空间。

而且，**从技术创新角度而言，前瞻性判断非常重要：方向判断正确，一路辛苦是披荆斩棘；方向判断错误，一路辛苦也是前程堪忧。**

国内的公司对RunC的贡献度越来越高，特别是华为，可能是国内公司中对RunC贡献最大的。还有EasyStack、南大索芙特等的贡献，反倒是一些著名的Docker创业公司看不到对RunC的贡献。这一方面反应了华为、EasyStack技术眼光和对社区的贡献，另外也反映了为什么华为和EasyStack在商业上也更成功一些。

总结和展望

虽然大家都没有明说，但是Docker和CaaS生态圈在容器上的分裂已经是现在进行时了。前面提到的Twitter上的论战，在我看来也将是容器和CaaS生态圈重要转折事件。

让我们来看看目前正在发生的和在未来一年中很有可能发生的事情：

1. 目前Docker已经被科普，客户更关注的是CaaS提供出的价值而不是容器本身。

2. 一些不适合容器的Docker应用场景的案例会被证伪。在Docker和容器鲜为人知的时候，各种各样的Docker案例层出不穷，包括一些明显和常识有违的案例，比如交易系统采用Docker, 交易极严格的延时的要求不适合Docker。有的是故意混淆概念，交易本身不在Docker容器中，交易系统相关的一些模块在Docker中，为了突出宣传效果，说交易系统采用Docker。

3. CaaS业界开始行动：

- Cloud Foundry率先采用RunC作为容器运行时，而且刚刚做了一个25万个容器集群的测试，https://www.cloudfoundry.org/cloud-foundry-approaching-250000-containers/验证了PaaS+RunC的大规模集群的支持。
- K8s的CRI-O也会尽快发布，等CRI-O成熟以后，内置的容器运行时就应当是RunC，而不再是Docker了。
- Mesos的UnifiedContainerizer 也应当会在1年之内成熟，随后内置的容器应当也是RunC，而不再是Docker。

4. Docker创业公司分化。越来越多的容器创业公司会称其是K8s/Mesos初创公司，而非Docker创业公司。即强调自己是CaaS厂商，而非Docker厂商。而目前的局势对于一些纯Docker创业公司不是很乐观，因为毕竟比起CaaS领域，Docker没有优势。

5. 那么Docker公司自己而言呢？在容器失去了K8s/Mesos/CloudFoundry的支持之后，会更专注于Swarm，和CaaS的其他厂商的竞争将更直接。但是在我看来，Docker公司一贯的对企业生产环境特性的不在乎，Swarm很难对其他CaaS形成竞争优势。

6. RunC的生态圈将越来越丰富，第一个就是把Docker镜像转换为RunC标准镜像(这个已经有了)，其次就是各种各样的插件和RunC可以交互，中间还可以衍生出各种插件的功能，如即插即用(动态性)、自动发现之内的。

不如，我们相约一年以后，再来复盘。

参考内容

本文部分内容原创，不过含有一些引用参考，向下列文章的作者致敬。

http://www.dockerinfo.net/1959.html

http://dockone.io/article/1672

http://weibo.com/1901150895/DzV6cDEC1?from=page_1005051901150895_profile&wvr=6&mod=weibotime&type=comment

http://www.infoq.com/cn/articles/docker-standard-container-execution-engine-runc

https://segmentfault.com/a/1190000007157374

http://www.echojb.com/dotnet-other/2016/09/25/213618.html

http://www.wtoutiao.com/p/4fdLO2y.html

http://dockone.io/article/1770

http://dockone.io/article/776

http://www.kubernetes.org.cn/300.html

http://dockone.io/article/1327

http://mp.weixin.qq.com/s/NlyJtev6sVTCGSaGSmVtzg

注：本文整理自《容器，你还只用Docker吗？》上下篇。原文首发于DBAplus社群。已获得作者及首发公众号授权。