DaoliNet

为PaaS提供网络虚拟化

DaoliNet开源社区

2016年10月16日

内容提要

简化PaaS服务的需求（一堆）

已知解决方案（container cloud）及未解问题

DaoliNet网络虚拟化解决方案

下一步工作

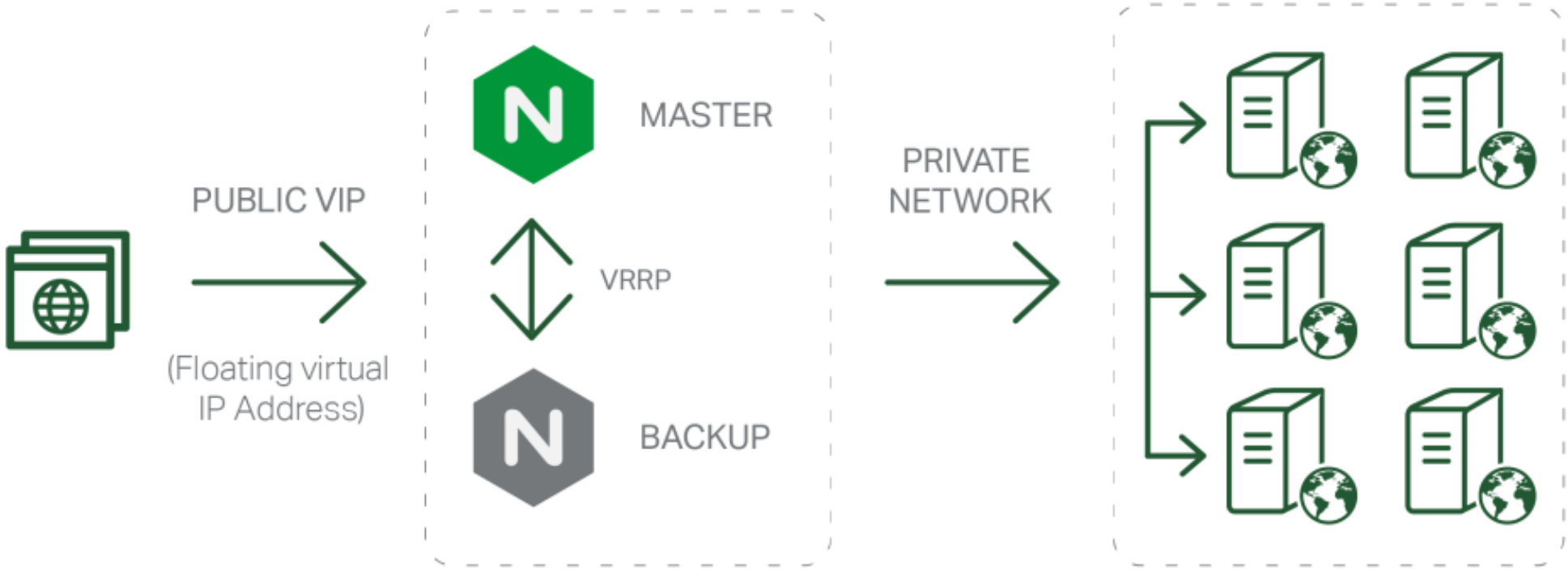
客户需求（举例）

客户：服务永远不挂！

PaaS提供商：一旦某服务挂了同样的备份服务瞬间就上，怎么样？

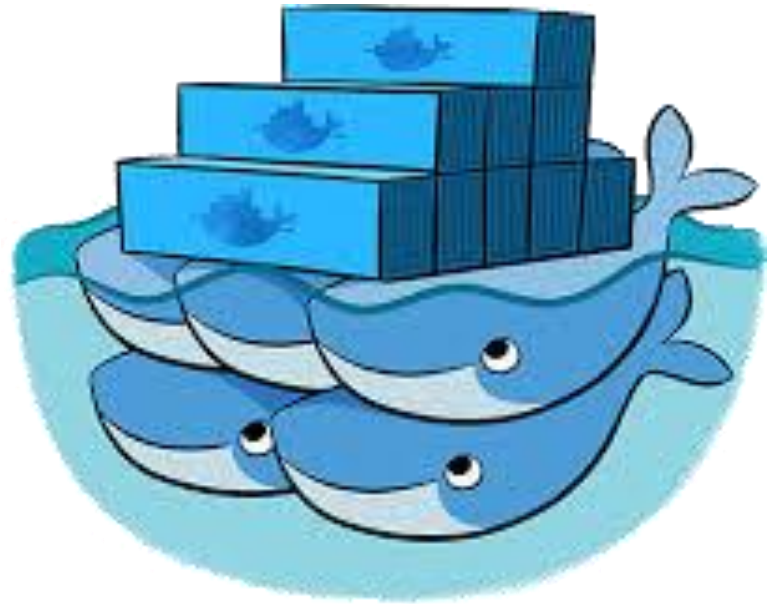
客户：别告诉我主、备，双活什么的，只要服务不挂就行！

PaaS提供商脑海中可能浮现的（举例）



非客户（PaaS）需求：DevOps工作流线型化

Container（sandbox）化的分布式应用DevOps，标准化装箱成服务，
通过名称（name）发现、组装和使用服务



Containers内部是SaaS应用程序编程的空间，
containers之间才是PaaS用武之地，箱体
是双方各自工作的边界，“互不干涉内政”

DevOps的整个生命周期中无需打开任何
container，尤其在ship, run过程中 *不要* 打开！

Container化的PaaS（微）服务举例

Apache、SQL、PHP、Nginx、SSL、IPSec，…

Container化使应用构建者可自定义服务，（微）服务就是名称而已

分布部署需求：Build, Ship, Run ANYWHERE

至于PaaS服务所用的资源是由什么物理设备提供的，这个不仅非常不重要，最好和DevOps的整个生命周期没有任何关系

资源位置无关的体现：如Docker Swarm会把containers随机分布部署到服务器集群上，Kubernetes也有类似分布式策略

只有这样才能简化DevOps流程，提高PaaS服务质量

微服务的连接需求

使用container对服务打包 + 不准打开container

=>

须用网络连接微服务containers

而且

网络资源须只与服务名称有关，与服务提供的物理位置无关

连接的动态需求

由于网络协议只能使用网络身份，如 (IP:Port)，通信，因此须将服务名称映射到网络身份

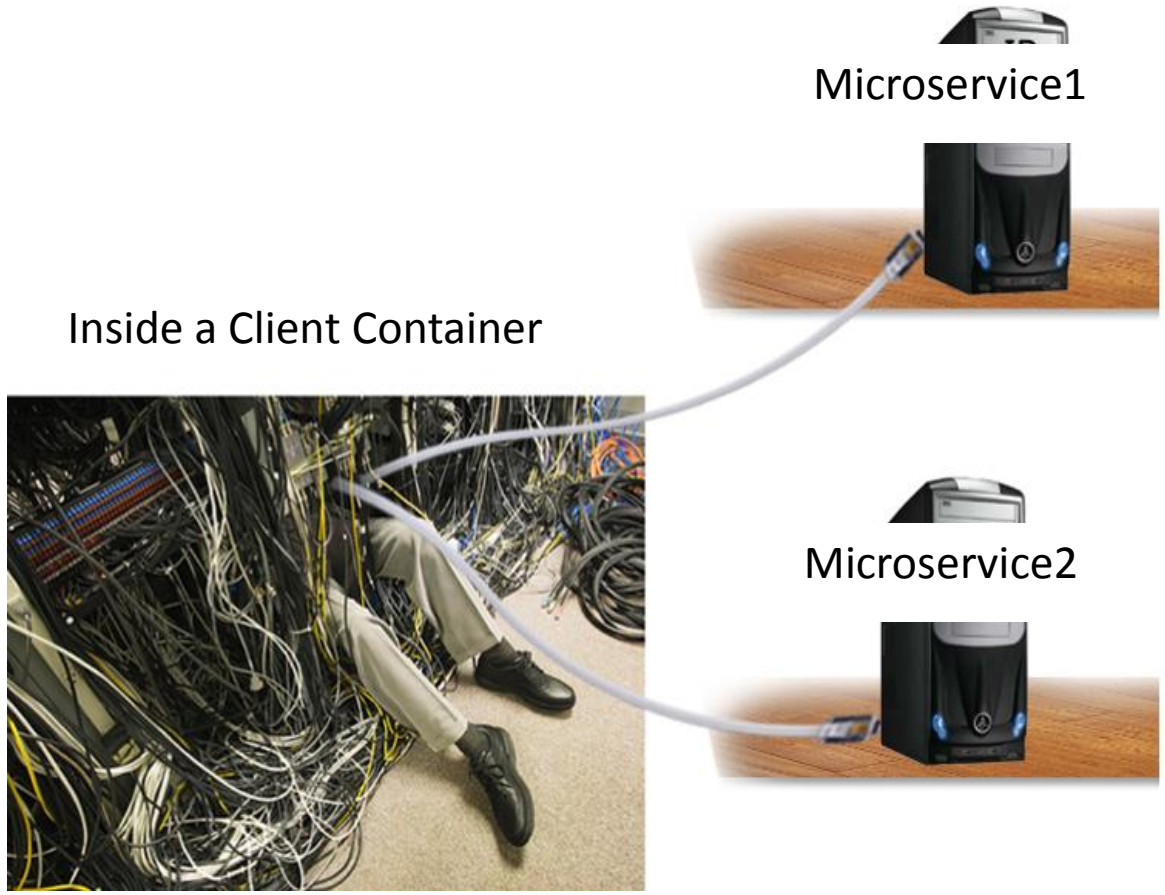
而“Run Anywhere”注定网络身份IP是动态变化的，所以映射须在runtime发生

已知解决方案

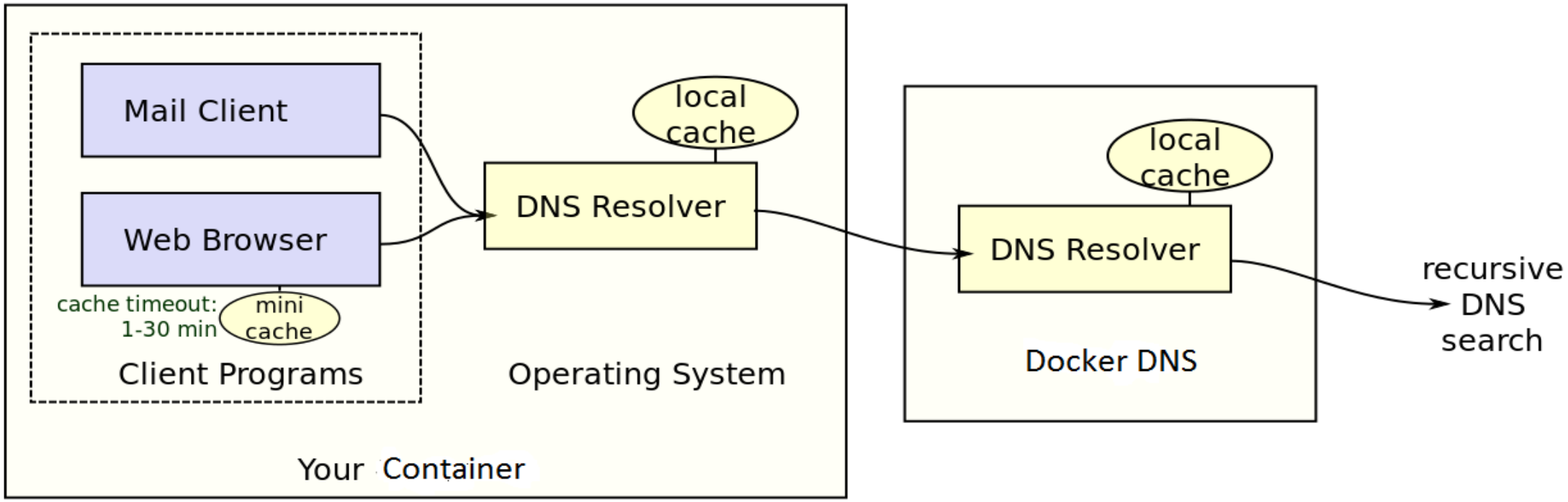
已知映射方法有Etcd，DNS，…，然而很遗憾已知方法似乎都要在runtime介入containers内部，造成“干涉内政”

什么内政？PaaS要求containers内部应用在runtime参与帮助路由

PaaS在runtime干涉内政使PaaS服务难度增加，服务质量下降 …

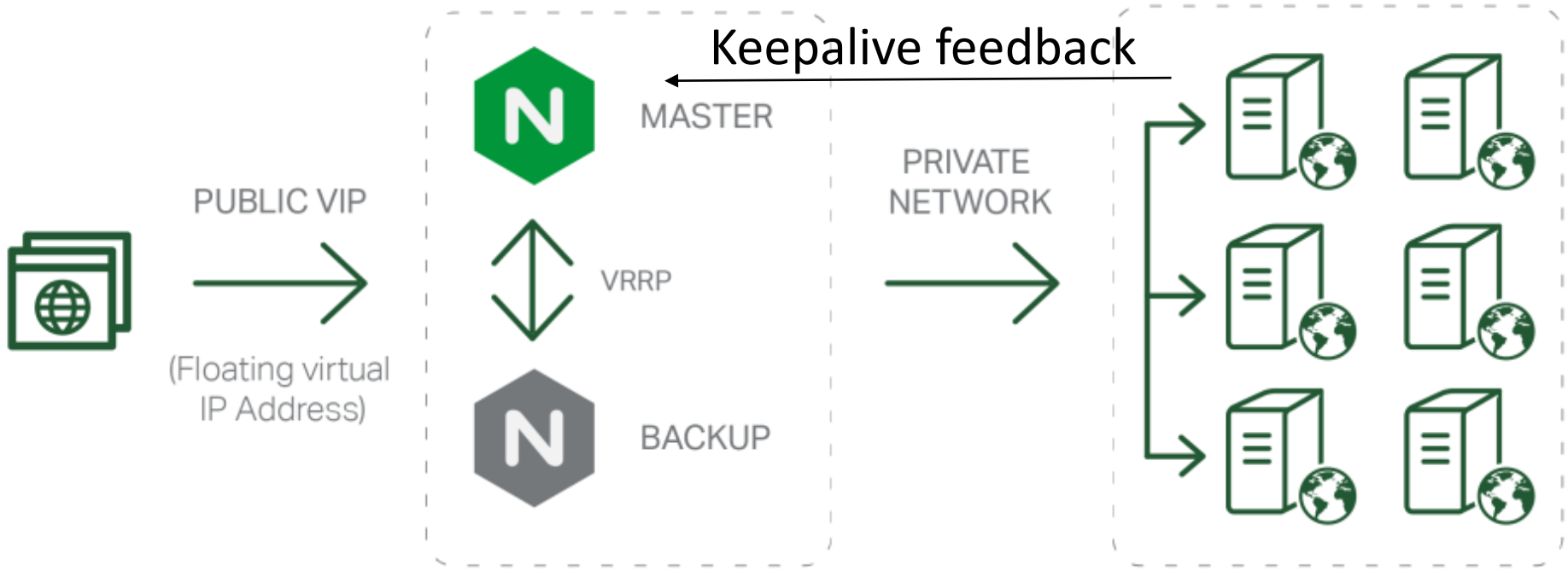


DNS：PaaS等待SaaS通知TTL超时



DNS是设计为人脑服务的，local cache更新速度不适用于机器之间，
若TTL超时省缺值设定太久，则service IP变化后还要重启client

而ETCD则要在client container内编程序了



回顾“服务不中断”客户需求：中间和右边的containers不具有build，ship，run属性，如用containers实现，则PaaS keepalive feedback逻辑要设计配置在中间containers内部，观察：不是client-server架构可解决的

如果PaaS服务不用打开container那该多好

假如从container内部看，IP也固定不变怎么样？

PaaS服务就不需要打开集装箱啦！

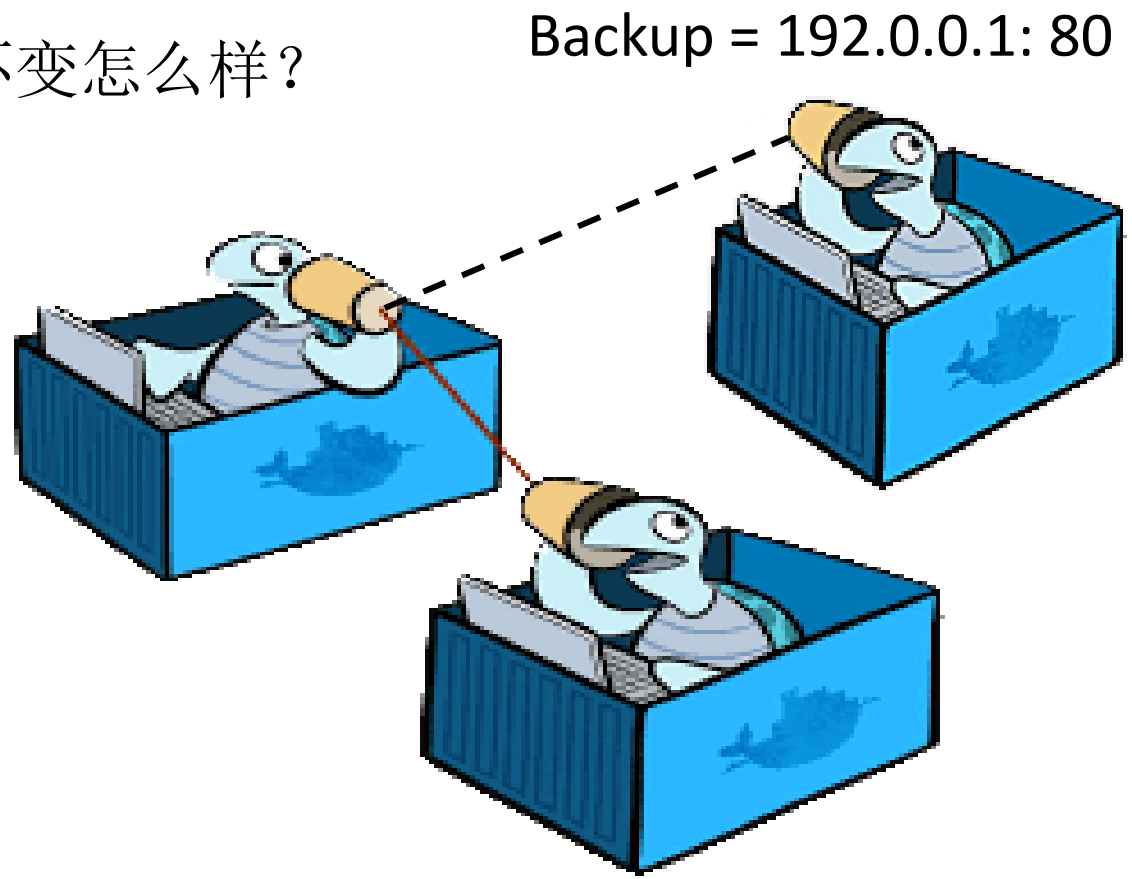
比如“服务不中断”需求：

当Master挂时，处于另一

物理位置的Backup接过服务

从container内部看 (IP:Port) 没变，

切换对container内部是seamless的



问题

IP固定不变？

Containers要动态分布部署在不同物理设备上，位置变了怎么会IP不变？

Port固定不变？

大家都抢着用80、443、22、25，…，而服务器的IP又是固定的，你也要IP:80，他也要IP:80，IP:Port冲突怎么路由？

网络虚拟化：无感知（abnostic）映射

Runtime对网络身份和服务名称的动态映射是不可避免的，但那是PaaS层工作，应该在containers外部进行

那样SaaS层就可在container内部看到恒定不变的网络身份，根本就不知道还存在动态映射

虚拟化的网络：PaaS、SaaS“互不干涉内政”，简化DevOps流程，提高PaaS服务质量

无感知映射原理

当container在一个物理服务器上launch时，PaaS orchestrator—如Docker Swarm, Kubernetes—可看到如下信息：

(ServerMAC, ServerIP, ServerPort; MAC, IP, Port)

其中哪怕 (IP, Port) 在应用的life-time固定不变，这个6-tuple含有足够信息量，具有全局唯一性，可以引导后继所需的正确路由

技术枝节

为确保传统路由的唯一性，Docker Swarm，Kubernetes都不允许创建具有相同IP地址的container

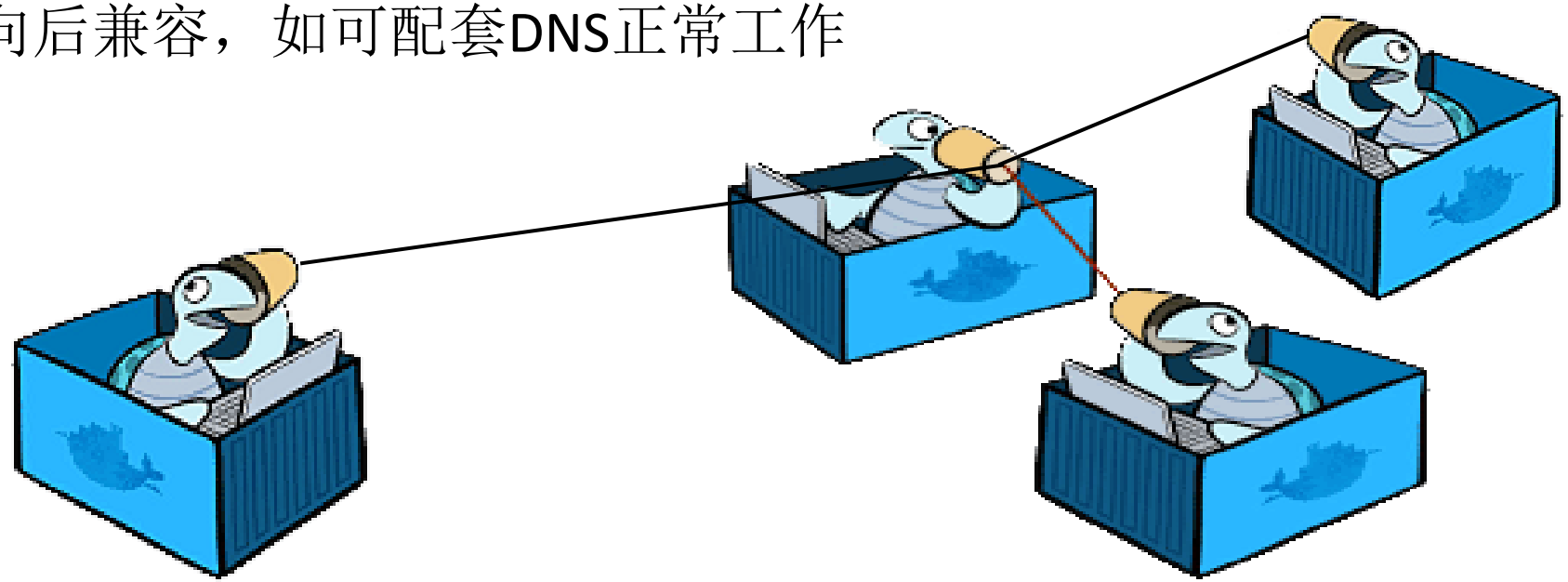
然而DaoliNet网络虚拟化路由的唯一性已经不再需要containers内部逻辑参与，而是因为cloud orchestrator在containers外部可见6-tupple sandbox化的网络信息

多亏这些cloud orchestration工具都是**开源**的，改了它们就是了

DAOLINET

性质：IP=虚拟服务器，Port=服务

- 表面看不同微服务的IP地址一样，containers却不一定在同一物理服务器上
- 虚拟IP地址“无限”多，无需Port映射解决Port冲突问题
- (IP:Port) 不变只是变的特例，所以虚拟化网络对现有PaaS网络技术向后兼容，如可配套DNS正常工作



应用

除了简化DevOps流程，DaoliNet网络虚拟化技术还有如下有趣应用：

1. 服务的动态横向扩展 (Scaleout) ： 对虚拟服务 (IP:Port) 的请求可被动态横向扩展路由到多个物理服务资源 (ServerMAC_i, ServerIP_i, MAC_i, IP, Port), for i = 1,2, 3, …
2. IPsec-Termination穿透网关防火墙：container的虚拟IP = 网关IP

DAOLINET

网络性能

DaoliNet属于3层非封包路由，网络性能与同样为3层非封包路由技术的Calico类似

实验结果

DaoliNet

[ID]	Interval	Transfer	Bandwidth
[1]	0.0-34.3 sec	2.00 GBytes	501 Mbits/sec
[2]	0.0-33.8 sec	2.00 GBytes	509 Mbits/sec
[3]	0.0-33.4 sec	2.00 GBytes	515 Mbits/sec
[4]	0.0-39.8 sec	2.00 GBytes	431 Mbits/sec
[5]	0.0-33.7 sec	2.00 GBytes	509 Mbits/sec
[6]	0.0-33.2 sec	2.00 GBytes	518 Mbits/sec
[7]	0.0-13.7 sec	2.00 GBytes	1.26 Gbits/sec
[8]	0.0-13.9 sec	2.00 GBytes	1.24 Gbits/sec
[9]	0.0-34.6 sec	2.00 GBytes	496 Mbits/sec
[10]	0.0-13.8 sec	2.00 GBytes	1.25 Gbits/sec

Calico

[ID]	Interval	Transfer	Bandwidth
[1]	0.0-20.8 sec	2.00 GBytes	826 Mbits/sec
[2]	0.0-33.5 sec	2.00 GBytes	513 Mbits/sec
[3]	0.0-26.8 sec	2.00 GBytes	642 Mbits/sec
[4]	0.0-21.4 sec	2.00 GBytes	802 Mbits/sec
[5]	0.0-40.8 sec	2.00 GBytes	421 Mbits/sec
[6]	0.0-13.7 sec	2.00 GBytes	1.25 Gbits/sec
[7]	0.0-73.9 sec	2.00 GBytes	233 Mbits/sec
[8]	0.0-14.0 sec	2.00 GBytes	1.23 Gbits/sec
[9]	0.0-35.0 sec	2.00 GBytes	491 Mbits/sec
[10]	0.0- 8.4 sec	2.00 GBytes	2.04 Gbits/sec

DAOLINET

下一步工作

PaaS IPAM （IP Address Management）新需求：

我们已知IaaS多租户实践要求Cloud Orchestrator允许启动具有相同逻辑IP地址的多个虚拟机

同样，PaaS的App Orchestrator也应该允许启动具有相同虚拟IP地址的多个containers，以满足简化DevOps，Scale-out等新需求

DaoliNet希望与开源社区同仁们共同努力一起做这个有意义的工作

DaoliNet 开放源代码， Apache 2.0 License

开源代码： github.com/daolinet/daolinet