[Calico在Docker中的搭建](http://www.cnblogs.com/tingfengainiaini/p/5010696.html)

**一，Multi-host网络需求**

开始之前推荐两篇文章

<http://xelatex.github.io/2015/11/15/Battlefield-Calico-Flannel-Weave-and-Docker-Overlay-Network/>

http://mp.weixin.qq.com/s?\_\_biz=MzAwMDU1MTE1OQ==&mid=400983139&idx=1&sn=f033e3dca32ca9f0b7c9779528523e7e&scene=1&srcid=1101jklWCo9jNFjdnUum85PG&from=singlemessage&isappinstalled=0#wechat\_redirect

Docker在1.9中libnetwork团队提供了multi-host网络功能，能完成overlay网络。但是网络功能需要linux主机的内核在3.16以上。对于使用centos或者mint之类的用户痛苦不堪，很多系统的内核还处于3.16以下。

随着SDN概念的到来、云平台的规模越来越大，Docker本身只能single host的问题严重限制了Docker的未来，虽然它相比VM有着数不尽的优势。

其实在docker1.9之前就可以通过一些方式实现multi-host功能，比如使用OpenVSwitch、Flannel、Calico、Weave等这些机制。

OVS和Flannel、Weave底层都是用了Vxlan的方式（包括Docker1.9本身也是使用了Vxlan的实现），都是能够完成Overlay的网络。

OVS和Flannel和Weave再细分就是就是实现的具体方式的不同了。OVS是比较成熟的技术，做的很多年功能也很强大， 但是配置复杂，对于大规模配置和项目迁移、扩容会有很大的麻烦。Flannel、Calico、Weave、Docker1.9具体的区别看我上面给出的这个github博客，博主很赞。

**二，Calico介绍**

Calico是一个纯3层协议，支持VM、Docker、Rocket、OpenStack、Kubernetes、或者直接在物理机上使用。官网上给出可以支持上万个主机、上百万的工作负载（container），由于它是纯三层协议，使用BGP协议（基于IP），更易于调试，支持IPv6，支持灵活的安全策略。



附上一个Calico官方给出的Calico与物理网络、OVS的性能对比：

<http://www.projectcalico.org/calico-dataplane-performance/>

觉得文章中有些“夸张了自己的效果”，我自己的测试结果与这个文章中的结果的差别有些大，虽然确实Calico在网络本身较好的情况下TCP有着绝对优势，但是它的UDP性能并不算很突出，在非ipip模式时能保持一点领先，但是在ipip模式时候，与Flannel基本等同，甚至更差。

这里推荐两篇Calico在Docker中使用的教程：

官方：<https://github.com/projectcalico/calico-docker/blob/master/docs/getting-started/default-networking/Demonstration.md>

一个博客：http://xelatex.github.io/2015/09/06/calico-docker/

**三，准备**

1，两台主机：10.11.150.72、10.11.150.74 （我这里是Centos）

2，Calico-node image两个方法获得：

    1，自己从Calico官方github上下载编译：<https://github.com/projectcalico/calico-docker>

    2，直接从Docker hub上下载现成的，不过版本可能没那么新：<https://hub.docker.com/search/?q=calico&page=1&isAutomated=0&isOfficial=0&starCount=0&pullCount=0>

3，Etcd：<http://pan.baidu.com/s/1mgIqn4O> 这里版本是2.2.1

4，一个有ping命令的image，什么都可以，最简单如busybox，我这里使用iperf的镜像

5，calicoctl ：自己找个版本，尽量最新版本，但是注意更新的内容。目前最新的是0.12.0：[https://github.com/projectcalico/calico-docker/release](https://github.com/projectcalico/calico-docker/releases%EF%BC%8C%E8%87%AA%E5%B7%B1%E6%89%BE%E4%B8%AA%E7%89%88%E6%9C%AC%EF%BC%8C%E5%B0%BD%E9%87%8F%E6%9C%80%E6%96%B0%E7%89%88%E6%9C%AC%EF%BC%8C%E4%BD%86%E6%98%AF%E6%B3%A8%E6%84%8F%E6%9B%B4%E6%96%B0%E7%9A%84%E5%86%85%E5%AE%B9%E3%80%82%E7%9B%AE%E5%89%8D%E6%9C%80%E6%96%B0%E7%9A%84%E6%98%AF0.12.0)s

**四，运行ETCD集群**

Calico要求拥有一个etcd cluster作为k-v存储，来存放自己的配置内容。

在10.11.150.72上运行：

[复制代码](javascript:void(0);)

#!/bin/sh

export localip=10.11.150.72

export name=calico0

export port1=2381

export port2=2379

export port3=4002

sudo ./etcd \

-name $name \

-initial-advertise-peer-urls http://$localip:$port1 \

-listen-peer-urls http://0.0.0.0:$port1 \

-listen-client-urls http://0.0.0.0:$port2,http://0.0.0.0:$port3 \

-advertise-client-urls http://$localip:$port2,http://$localip:$port3\

-initial-cluster-token etcd-cluster \

-initial-cluster calico0=http://$localip:$port1,calico1=http://10.11.150.74:$port1 -initial-cluster-state new &

[复制代码](javascript:void(0);)

注意我这里的端口和localip的配置，另外-initial-cluster一定要包含-initial-advertise-peer-urls。

然后在10.11.150.74上运行etcd，组成cluster，两台机子都输出published name后暂时没有新的输出就算是etcd cluster组建完成。

测试一下：

本机：

curl -L 127.0.0.1:2379/version

集群：

curl -L 10.11.150.74:2379/version

**五，Calico Service**

Calico在每个主机上通过一个自己的container与其他主机或者网络通讯，即calico-node的container，这个container里面包含了Bird路由管理、Felix协议等。

介绍：<http://www.projectcalico.org/learn/>

在两台主机上分别运行，后面的ip是主机ip

./calicoctl node --ip=10.11.150.72

运行后在两个主机用docker ps可以看到正在运行calico-node的container

下面为我们的calico网络添加可用的ip pool（在72或者74一台主机上运行即可）：

./calicoctl pool add 172.1.0.0/16 --nat-outgoing

这里由于我的两个主机72和74本身就在同一个子网下，主机相互之间不需要L2 switch。如果是跨子网、跨机房、跨公网就需要加上--ipip选项，或者你有权限直接修改路由器的BGP协议，将两个不同子网的主机连接为BGP peer。

**六，Container开始吧**

在主机72上面新建两个container：

docker run --net=none --name worker-1 -tid iperf

docker run --net=none --name worker-2 -tid iperf

在74上也新建两个

docker run --net=none --name worker-3 -tid iperf

docker run --net=none --name worker-4 -tid iperf

为container在calico中注册一个独立的IP：

./calicoctl container add worker-1 172.1.0.1

./calicoctl container add worker-2 172.1.0.2

./calicoctl container add worker-3 172.1.0.3

./calicoctl container add worker-4 172.1.0.4

Calico通过profile的形式来控制ACL，也以此来完成安全策略。

添加两个profile(72或者74均可)：

./calicoctl profile add PROF\_1

./calicoctl profile add PROF\_2

为container添加策略：

./calicoctl container worker-1 profile append PROF\_1

./calicoctl container worker-2 profile append PROF\_1

./calicoctl container worker-3 profile append PROF\_1

./calicoctl container worker-4 profile append PROF\_2

这样worker 1 2 3同在profile内，worker-4单独在PROF\_2内。

**七，测试一下吧**

测试一下worker-1和worker-3之间可以通：

docker exec worker-1 ping -c 4 172.1.0.3

测试一下worker-1和worker-4之间不在同一个profile下面不能通：

docker exec worker-1 ping -c 4 172.1.0.4

**八，性能测试：**

推荐使用iperf、qperf、scp、ping等常见命令测试。

**九，Calico缺陷**

直接借用<http://xelatex.github.io/2015/09/06/calico-docker/>

* Calico only supports **TCP, UDP, ICMP and ICMPv6** protocol. If you want to use other L4 protocols, you need to choose Flannel, Weave or Docker Overlay Network.
* Calico doesn’t have encryption data path. It’s not safe to build overlay network with Calico over untrusted network.
* The performance of Calico with IP-over-IP option is quite bad, which --ipip option is a must in a public data center connected with IP network.（或者将两主机通过BGP协议连接）
* No IP overlap support. Though Calico community is developing a experimental feature that put overlap IPv4 packages into IPv6 package. But this is only an auxiliary solution and doesn’t fully support IP overlap technically.