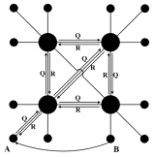


从Docker到Kubernetes 第4周

法律声明

【声明】本视频和幻灯片为炼数成金网络课程的教学资料，所有资料只能在课程内使用，不得在课程以外范围散播，违者将可能被追究法律和经济责任。

课程详情访问炼数成金培训网站

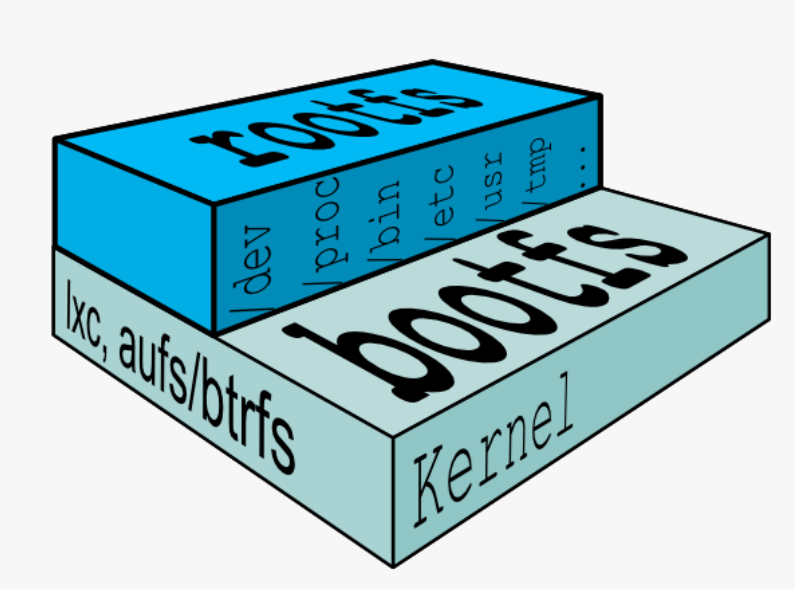
http://edu.dataguru.cn

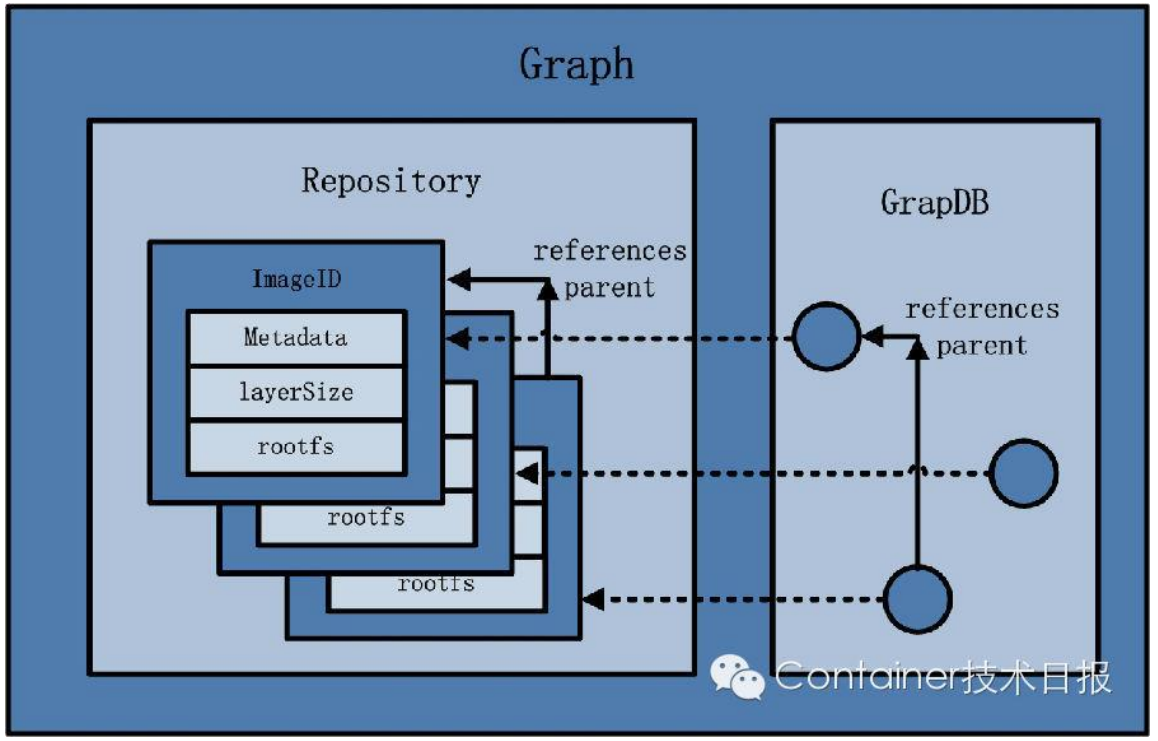
Docker实战之容器互联实战

- 基于Volume的互联
- 基于Link的互联
- 基于网络的互联

基于Volume的互联

理解Docker Volume





/var/lib/docker/graph 存放本地Image里的分层信息

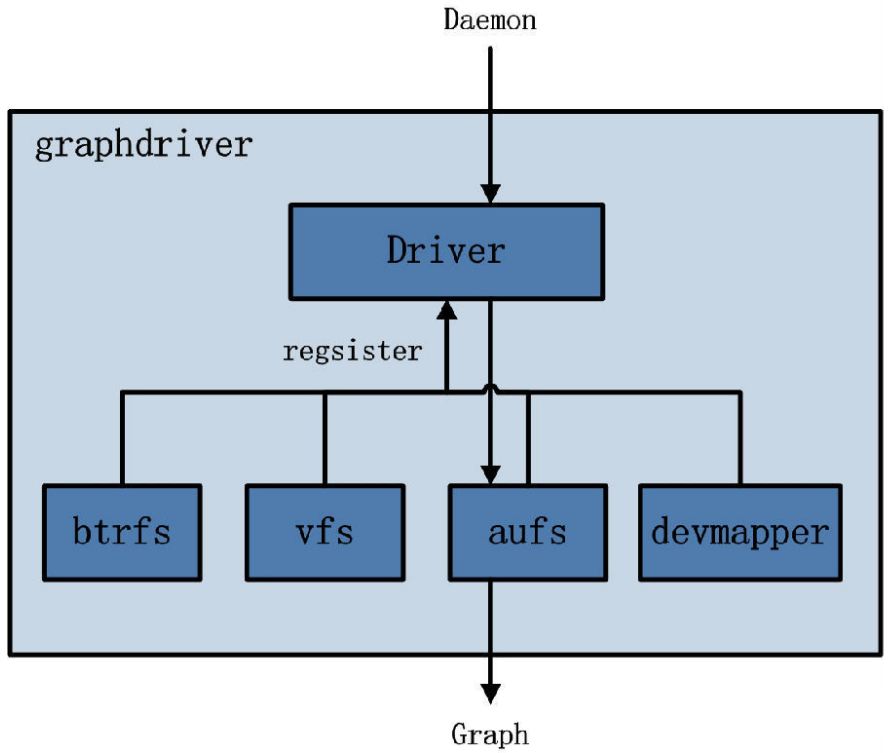
/var/lib/docker/devicemapper/devicemapper/data 存储了Image与Container的二进制数据文件
/var/lib/docker/devicemapper/devicemapper/metadata 存储了相关元数据

基于Volume的互联

理解Docker Volume

```
[root@localhost ~]# du -h /var/lib/docker/devicemapper/devicemapper/data
3.9G    /var/lib/docker/devicemapper/devicemapper/data
```

```
[root@localhost ~]# ls -al /var/lib/docker/devicemapper/devicemapper/data
-rw-------. 1 root root 107374182400 Aug 26 14:03 /var/lib/docker/devicemapper/devicemapper/data
```



Aufs driver是Docker最早支持的driver，但是aufs只是Linux内核的一个补丁集

Device mapper是Linux 2.6内核中提供的一种从逻辑设备到物理设备的映射框架机制，是LVM2的核心，支持块级别的copy on write特性

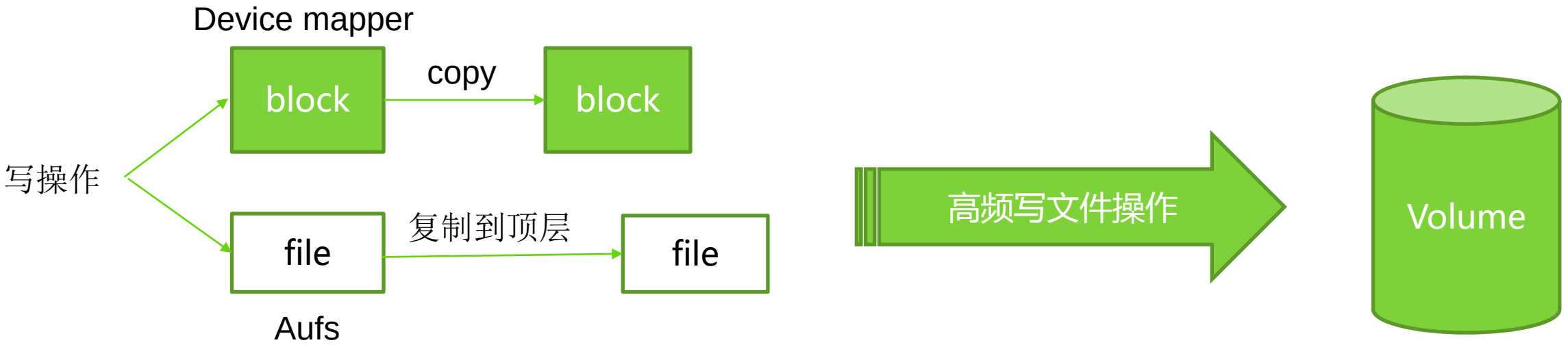
VFS虚拟文件系统的最大缺陷是不支持copy on write特性，每层都是一个单独的目录，如果新增一个child层，则需要将父级层镜像文件一并复制到新目录

btrfs 非常快，采用btrfs的文件系统级的快照能力来实现layer分层功能，缺点是仍然在进化中，还不够成熟，特别是大量写操作的压力下

目前，除少数版本如Ubuntu，Docker基本运行在Devicemapper基础上

基于Volume的互联

理解Docker Volume



为什么需要Volume

基于Volume的互联

理解Docker Volume

删除容器?

docker run --rm=true -it -v /leader java /bin/bash

```
[root@localhost ~]# docker run --rm=true  -it -v /leader java /bin/bash
root@ea667a11631f:/# ls
bin  boot  dev  etc  home  leader  lib lib64  media  mnt  opt   proc  root
```

```
root@ea667a11631f:/# ls
bin  boot  dev  etc  home  leader  lib lib64  media  mnt  opt   proc  root  run  sbin  s
root@ea667a11631f:/# ls /leader/
mybooks
```

docker inspect ea667a11631f

```
"Mounts": [
    {
        "Name": "bb026c87365a426f4ea44a697142549428b727661718afc1e24ebc45df4c8180",
        "Source": "/var/lib/docker/volumes/bb026c87365a426f4ea44a697142549428b727661718afc1e24ebc45df4c8180/_data",
        "Destination": "/leader",
        "Driver": "local",
        "Mode": "",
        "RW": true
    }
```

mkdir /var/lib/docker/volumes/bb026c87365a426f4ea44a697142549428b727661718afc1e24ebc45df4c8180/_data/mybook

基于Volume的互联

理解Docker Volume

docker run --rm=true -it -v /storage /leader java /bin/bash

本机目录　　容器目录

```
[root@localhost ~]# docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS
1b8e145bd5bc        java                "/bin/bash"         7 seconds ago       Up 5 seconds
dd8be4b1bf7a        java                "/bin/bash"         16 seconds ago      Up 14 seconds
```

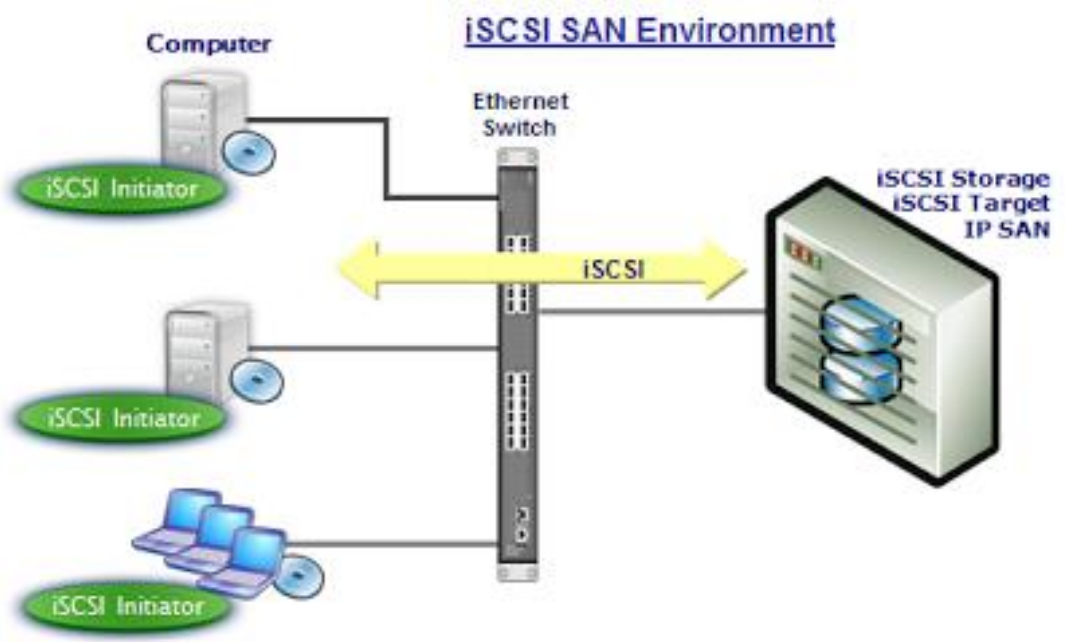
可以多个容器中的Volume指向同一个本机目录，实现基于文件的的共享访问

```
root@dd8be4b1bf7a:/# mkdir /leader/mybook3
mkdir: cannot create directory ? . leader/mybook3? .  Permission denied
root@dd8be4b1bf7a:/# exit
```

```
[root@localhost ~]# docker run --rm=true  --privileged=true  -it -v /storage:/leader java /bin/ba
root@3d1ada481637:/# ls
bin  boot  dev  etc  home  leader  lib  lib64  media  mnt  opt  proc  root  run  sbin  srv  sys
root@3d1ada481637:/# mkdir leader/mybook3
```

基于Volume的互联

基于Volume的互联，也可以解决跨主机的共享问题



iscsi

nfs

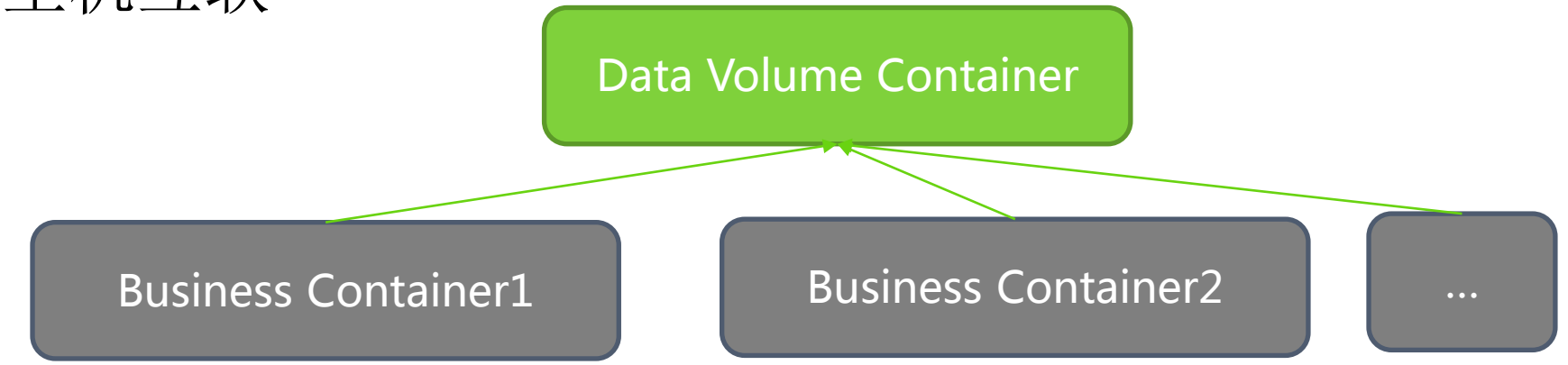
ceph

分布式文件系统

基于Volume的互联

基于数据容器的单主机互联

docker run -it -v /leader java /bin/bash



docker run --rm=true --privileged=true **--volumes-from=3d1ada481637** -it java /bin/bash

```
[root@localhost ~]# docker run --rm=true   --privileged=true   --volumes-from=3d1ada481637  -it  java /bin/bash
root@3a16da688f4f:/# ls /leader
mybook3  mybooks  mybooks2
root@3a16da688f4f:/# mkdir /leader/mybook4
```

基于link的互联

docker run --rm=true --name=mysqlserver -e MYSQL_ROOT_PASSWORD=123456 mysql

```
2015-09-06 22:13:30 1 [Note] Server socket created on IP: '0.0.0.0'.
2015-09-06 22:13:30 1 [Warning] 'proxies_priv' entry '@ root@14b3d1d518c8' ignored in --skip-name-resolve mode.
2015-09-06 22:13:30 1 [Note] Event Scheduler: Loaded 0 events
2015-09-06 22:13:30 1 [Note] mysqld: ready for connections.
Version: '5.6.26'  socket: '/var/run/mysqld/mysqld.sock'  port: 3306  MySQL Community Server (GPL)
```

```
root@1aa4b50d4a81:/# cat /etc/hosts
172.17.0.13     1aa4b50d4a81
127.0.0.1       localhost
::1     localhost ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
172.17.0.12     serverM1 5f9c195dd337 myjaveserver
172.17.0.12     myjaveserver
172.17.0.12     myjaveserver.bridge
172.17.0.13     high_ardinghelli
172.17.0.13     high_ardinghelli.bridge
```

docker run --rm=true   -it java curl 172.17.0.1:3306

```
[root@localhost ~]# docker run --rm=true   -it  java curl 172.17.0.1:3306
5.6.26□fYj*0R/0□□EH{Re*=3*t.&mysql_native_password!□ . #08S01Got packets out of order
```

基于link的互联

link方式

docker默认是允许container互通，通过-icc=false关闭互通。一旦关闭了互通，只能通过-link name:alias命令连接指定container.

-- link redis:db的别名，会在/etc/hosts中生成对应的ip映射

基于link的互联

--link=myjaveserver:serverM1

给一个主机名（DNS名称）用来代替IP地址进行访问

目标容器（需要连接的容器）

```
[root@localhost ~]# docker run --rm=true  --name=myjaveserver  -it  java /bin/bash
root@5f9c195dd337:/# ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
26: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
    link/ether 02:42:ac:11:00:0c brd ff:ff:ff:ff:ff:ff
    inet 172.17.0.12/16 scope global eth0
       valid_lft forever preferred_lft forever
```

```
[root@localhost ~]# docker run --rm=true  --link=myjaveserver:serverM1  -it  java /bin/bash
```

```
root@1aa4b50d4a81:/# ping serverM1
PING serverM1 (172.17.0.12): 56 data bytes
64 bytes from 172.17.0.12: icmp_seq=0 ttl=64 time=0.426 ms
64 bytes from 172.17.0.12: icmp_seq=1 ttl=64 time=0.162 ms
64 bytes from 172.17.0.12: icmp_seq=2 ttl=64 time=0.148 ms
```

```
root@1aa4b50d4a81:/# ping myjaveserver
PING serverM1 (172.17.0.12): 56 data bytes
64 bytes from 172.17.0.12: icmp_seq=0 ttl=64 time=0.133 ms
64 bytes from 172.17.0.12: icmp_seq=1 ttl=64 time=0.141 ms
64 bytes from 172.17.0.12: icmp_seq=2 ttl=64 time=0.143 ms
```

基于link的互联

/usr/bin/docker daemon **--icc=false --iptables=true**

```
^C^C^C[root@localhost ~]# docker run --rm=true  -it java ping 172.17.0.1
PING 172.17.0.1 (172.17.0.1): 56 data bytes
92 bytes from 172.17.42.1: Dest Unreachable, Unknown Code: 10
92 bytes from 172.17.42.1: Dest Unreachable, Unknown Code: 10
92 bytes from 172.17.42.1: Dest Unreachable, Unknown Code: 10
^C92 bytes from 172.17.42.1: Dest Unreachable, Unknown Code: 10
--- 172.17.0.1 ping statistics ---
```

docker run --rm=true --link=mysqlserver:myserver -it java /bin/bash

iptables-save

```
-A DOCKER -s 172.17.0.4/32 -d 172.17.0.1/32 -i docker0 -o docker0 -p tcp -m tcp --dport 3306 -j ACCEPT
-A DOCKER -s 172.17.0.1/32 -d 172.17.0.4/32 -i docker0 -o docker0 -p tcp -m tcp --sport 3306 -j ACCEPT
```

```
root@a3e9762a4036:/# ping 172.17.0.1
PING 172.17.0.1 (172.17.0.1): 56 data bytes
92 bytes from 172.17.42.1: Dest Unreachable, Unknown Code: 10
92 bytes from 172.17.42.1: Dest Unreachable, Unknown Code: 10
92 bytes from 172.17.42.1: Dest Unreachable, Unknown Code: 10
^C--- 172.17.0.1 ping statistics ---
3 packets transmitted, 0 packets received, 100% packet loss
root@a3e9762a4036:/# curl  172.17.0.1:3306
5.6.26□x_Tf$~rB□□/e/XPg5r_cVcmysql_native_password!□ . #08S01Got packets out of orderroot@a3e9762a4036:/#
```

基于link的互联

/usr/bin/docker daemon **--icc=false --iptables=true**

docker run --rm=true **--volumes-from=3d1ada481637** -it java /bin/bash

docker run --rm=true java curl 172.17.0.1:3306

```
[root@localhost ~]# docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS               NAMES
04db7a10960f        mysql               "/entrypoint.sh mysql"   11 seconds ago      Up 10 seconds       3306/tcp            mysqlserver
[root@localhost ~]# docker exec -it 04db7a10960f  ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
46: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP
    link/ether 02:42:ac:11:00:01 brd ff:ff:ff:ff:ff:ff
    inet 172.17.0.1/16 scope global eth0
       valid_lft forever preferred_lft forever
[root@localhost ~]# docker run --rm=true java curl 172.17.0.1:3306
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0curl: (7) Failed to connect to 172.17.0.1 port 3306: No route to host
```

docker run --rm=true --link=mysqlserver:myserver -it java ping myserver

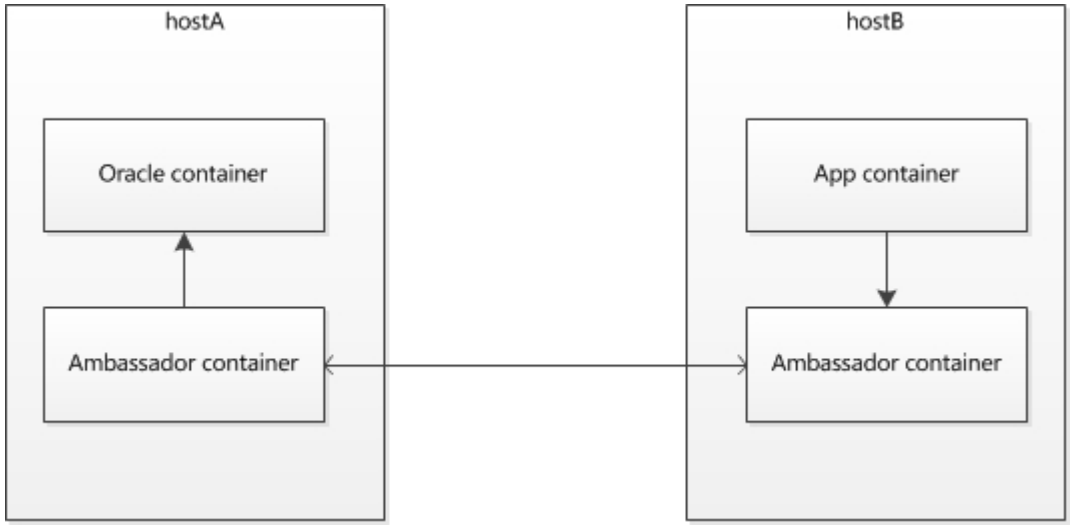
```
[root@localhost ~]# docker run --rm=true  --link=mysqlserver:myserver  -it java ping myserver
PING myserver (172.17.0.1): 56 data bytes
64 bytes from 172.17.0.1: icmp_seq=0 ttl=64 time=1003.274 ms
64 bytes from 172.17.0.1: icmp_seq=1 ttl=64 time=2.076 ms
64 bytes from 172.17.0.1: icmp_seq=2 ttl=64 time=0.077 ms
64 bytes from 172.17.0.1: icmp_seq=3 ttl=64 time=0.064 ms
64 bytes from 172.17.0.1: icmp_seq=4 ttl=64 time=0.066 ms
```

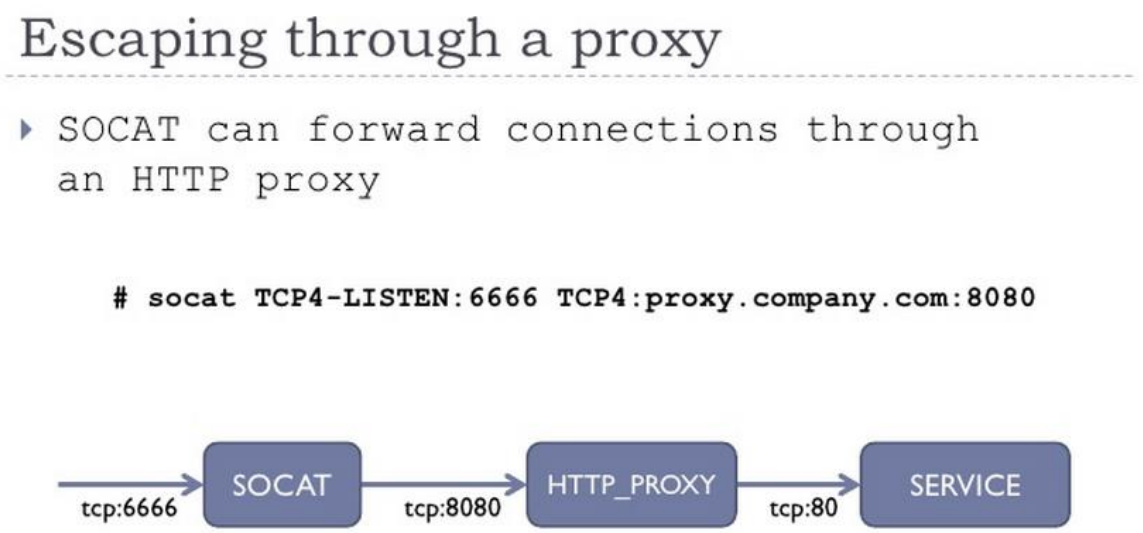
基于link的互联

跨主机Link?

Docker远程代理（Ambassador）模式

https://github.com/gliderlabs/connectable

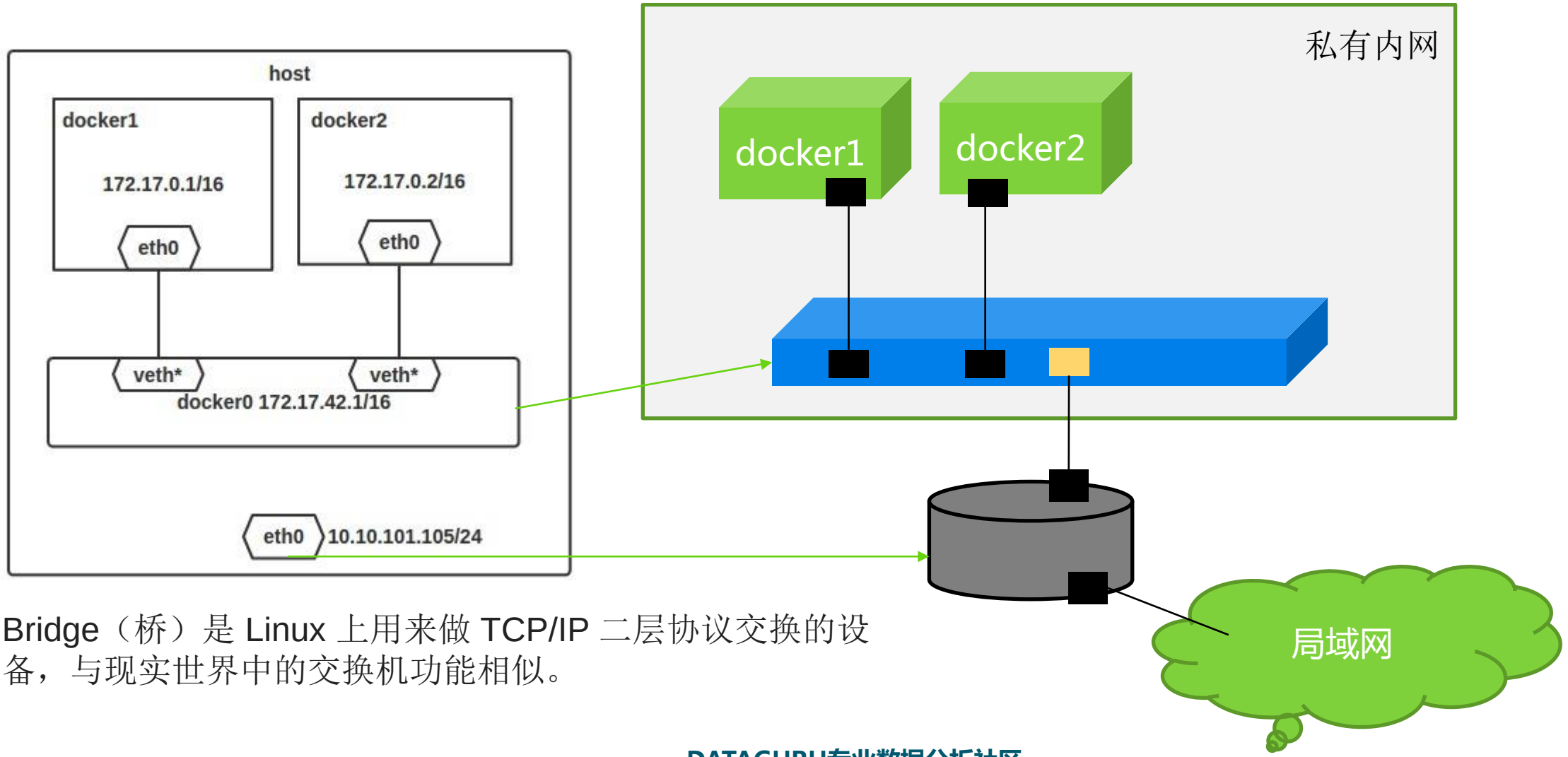




socat是一个多功能的网络工具，名字来由是”SOcket CAT”

基于网络的互联

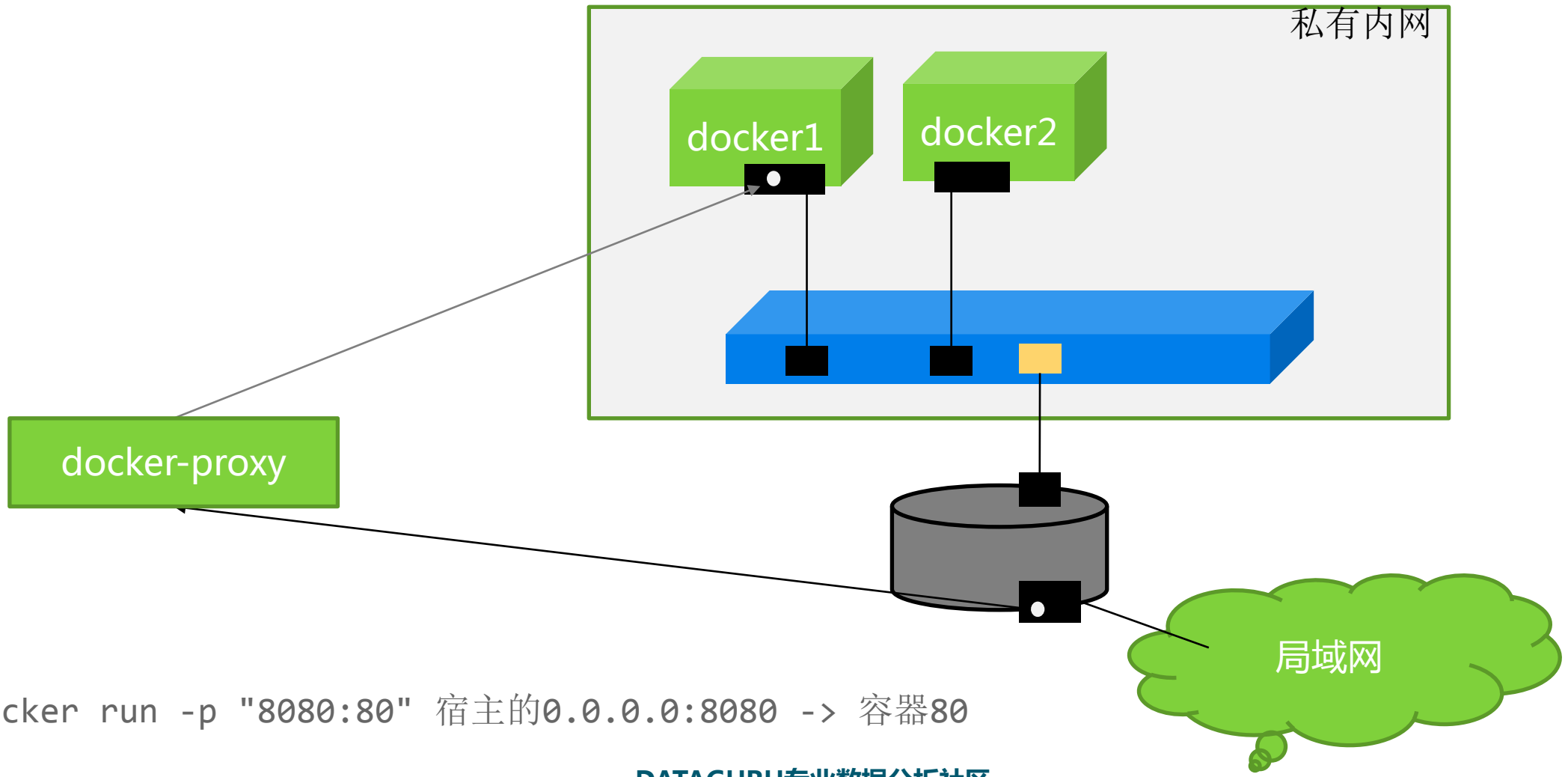
先理解Docker的网络模型



Bridge（桥）是 Linux 上用来做 TCP/IP 二层协议交换的设备，与现实世界中的交换机功能相似。

基于网络的互联

最简单常用的互联方式：端口映射




```
docker run -p "8080:80" 宿主的0.0.0.0:8080 -> 容器80
```

基于网络的互联

最简单常用的互联方式：端口映射

docker run --rm=true --name=mysqlserver -p 8066:3306 -e MYSQL_ROOT_PASSWORD=123456 mysql

```
[root@localhost ~]# ps -efwww|grep docker
root      11271      1  0 17:12 ?        00:00:26 /usr/bin/docker daemon -H fd:// -H=unix:///var/run/docker.sock -H=tcp://0.0.0.0:2375 --registry-mirror=h
root      12646  12437  0 20:38 pts/0    00:00:00 docker run --rm=true --name=mysqlserver -p 8066:3306 -e MYSQL_ROOT_PASSWORD=123456 mysql
root      12716  11271  0 20:38 ?        00:00:00 docker-proxy -proto tcp -host-ip 0.0.0.0 -host-port 8066 -container-ip 172.17.0.5 -container-port 3306
root      12885  12763  0 20:38 pts/1    00:00:00 grep --color=auto docker
```

docker-proxy -proto tcp -host-ip 0.0.0.0 -host-port 8066 -container-ip 172.17.0.5 -container-port 3306

```
[root@localhost ~]# netstat -tlp
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 0.0.0.0:8066            0.0.0.0:*               LISTEN      12716/docker-proxy
tcp        0      0 0.0.0.0:2375            0.0.0.0:*               LISTEN      11271/docker
tcp        0      0 0.0.0.0:ssh             0.0.0.0:*               LISTEN      972/sshd
tcp        0      0 localhost:smtp          0.0.0.0:*               LISTEN      2152/master
```

基于网络的互联

最简单常用的互联方式：端口映射

Apparently there are some edge cases without a better workaround (for now):
- localhost<->localhost routing
- docker instance calling into itself via its published port
- and possibly more

```
-A DOCKER ! -i docker0 -p tcp -m tcp --dport 8066 -j DNAT --to-destination 172.17.0.6:3306
-A DOCKER -d 172.17.0.6/32 ! -i docker0 -o docker0 -p tcp -m tcp --dport 3306 -j ACCEPT
-A POSTROUTING -s 172.17.0.6/32 -d 172.17.0.6/32 -p tcp -m tcp --dport 3306 -j MASQUERADE
```

Do we really need the proxy to be using 11MB of ram... **per port**!

Even one of the `docker-proxy` processes is using more than my znc server, and there are 3 ports forwarded from it!

phemmer referenced this issue on 30 Mar

Docker default bridge network makes networking very slow #11911

The user land proxy is going to be removed. Since removing the user land proxy will fix this problem, I'm going to close this in favor of #11185 because the solution to #11185 is the complete removal of the user land proxy.

基于网络的互联

直接使用宿主机网络

docker run --rm=true --net=host --name=mysqlserver -e MYSQL_ROOT_PASSWORD=123456 mysql

```
[root@localhost ~]# netstat -tlp
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 0.0.0.0:2375            0.0.0.0:*               LISTEN      11271/docker
tcp        0      0 0.0.0.0:mysql           0.0.0.0:*               LISTEN      13940/mysqld
tcp        0      0 0.0.0.0:ssh             0.0.0.0:*               LISTEN      972/sshd
tcp        0      0 localhost:smtp          0.0.0.0:*               LISTEN      2152/master
```

```
[root@localhost ~]# docker  exec -it mysqlserver  ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP qlen 1000
    link/ether 00:0c:29:e8:02:c7 brd ff:ff:ff:ff:ff:ff
    inet 192.168.18.128/24 brd 192.168.18.255 scope global dynamic eth0
       valid_lft 1726sec preferred_lft 1726sec
3: docker0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP
    link/ether 02:42:bc:35:f3:31 brd ff:ff:ff:ff:ff:ff
    inet 172.17.42.1/16 scope global docker0
       valid_lft forever preferred_lft forever
101: veth3839c0b: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue master docker0 state UP
    link/ether 6a:4b:02:90:fc:43 brd ff:ff:ff:ff:ff:ff
```

基于网络的互联

容器共用一个IP网络

docker run --rm=true --name=mysqlserver -e MYSQL_ROOT_PASSWORD=123456 mysql

docker run --rm=true --net=container:mysqlserver java ip addr

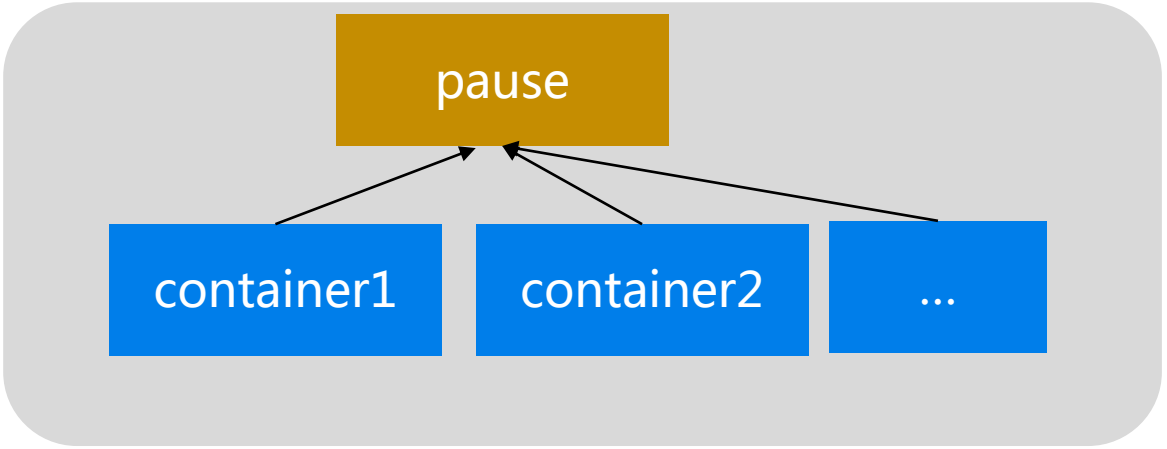
```
[root@localhost ~]# docker run --rm=true  --net=container:mysqlserver  java  ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
102: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
    link/ether 02:42:ac:11:00:08 brd ff:ff:ff:ff:ff:ff
    inet 172.17.0.8/16 scope global eth0
       valid_lft forever preferred_lft forever
```

```
[root@localhost ~]# docker exec  mysqlserver ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
102: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP
    link/ether 02:42:ac:11:00:08 brd ff:ff:ff:ff:ff:ff
    inet 172.17.0.8/16 scope global eth0
       valid_lft forever preferred_lft forever
```

基于网络的互联

容器共用一个IP网络


```
[root@localhost ~]# docker run --rm=true  --net=container:mysqlserver  java  curl localhost:3306
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   110     0   110    0      0    522       0 --:--:-- --:--:-- --:--:--   5315.6.26□<KF\ztV2□□5R|<SnO~PT
```



Kubernetes Pod

基于网络的互联

目前更为复杂的主流方向

docker 容器的IP地址能够被另外主机所访问

Thanks

FAQ时间