Docker端口&容器互联

@姜桥

目录

- 外部容器访问
- Docker网络模型
- Docker网络模式
- Pipework
- Mobike实践

外部容器访问

例如:Docker中安装一个Mysql：

1、从Docker镜像仓库获取Mysql官方镜像包

docker pull docker.io/mysql

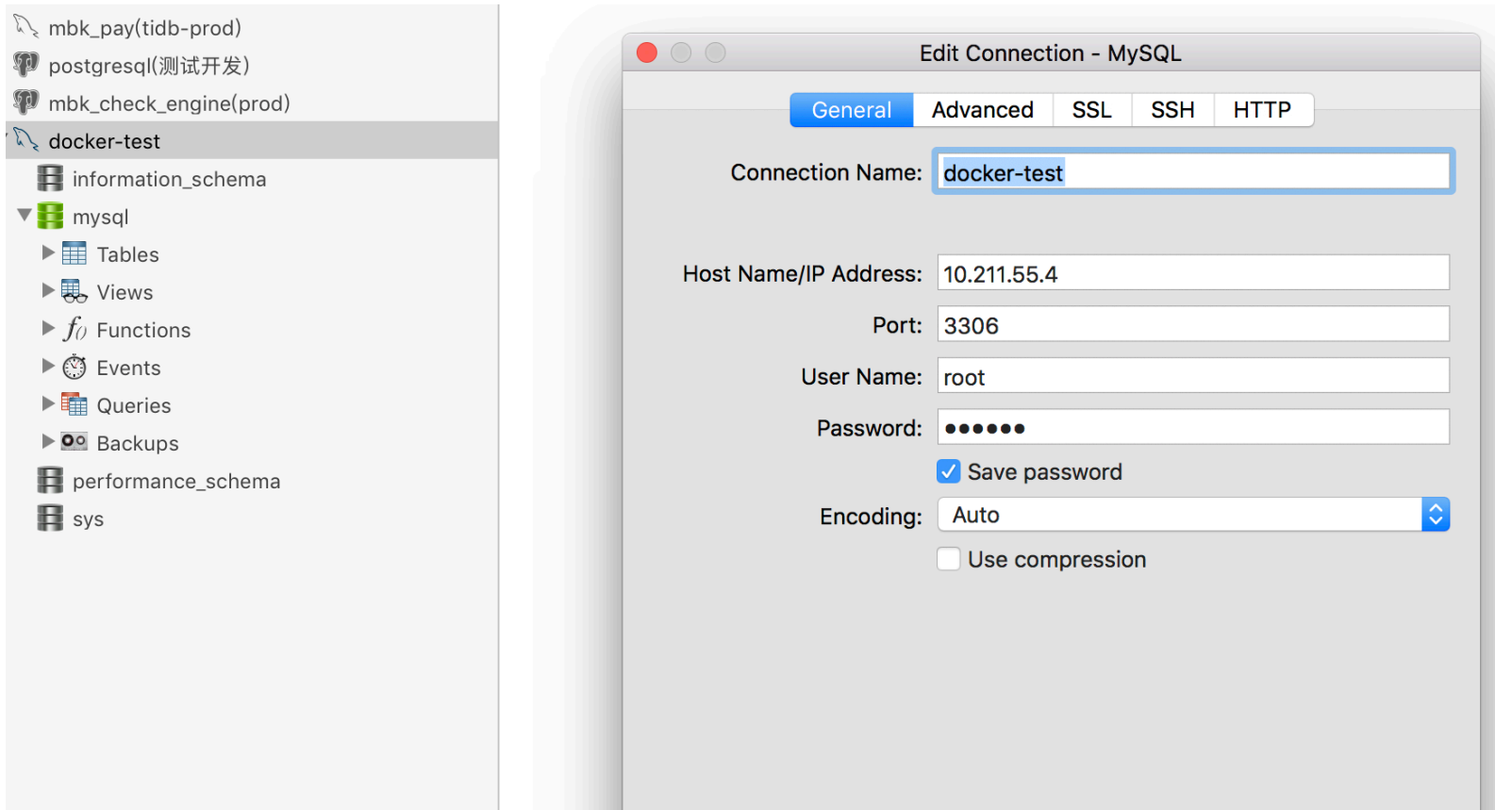
2、启动Mysql容器并指定端口映射

//创建宿主挂载目录：

sudo mkdir –p /data/mysql/logs /data/mysql/conf /data/mysql/data -p

docker run --name mysql **-p 3306:3306** -v /data/mysql/data:/var/lib/mysql -v /data/mysql/conf:/etc/mysql/conf.d -e MYSQL_ROOT_PASSWORD=123456 -d mysql

```
[jiangqiao@localhost logs]$ docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS                               NAMES
b3b8139754c7        mysql               "docker-entrypoint.s…"   34 minutes ago      Up 34 minutes       0.0.0.0:3306->3306/tcp, 33060/tcp   mysql
```



通过宿主机IP访问容器中的MySql

外部容器访问

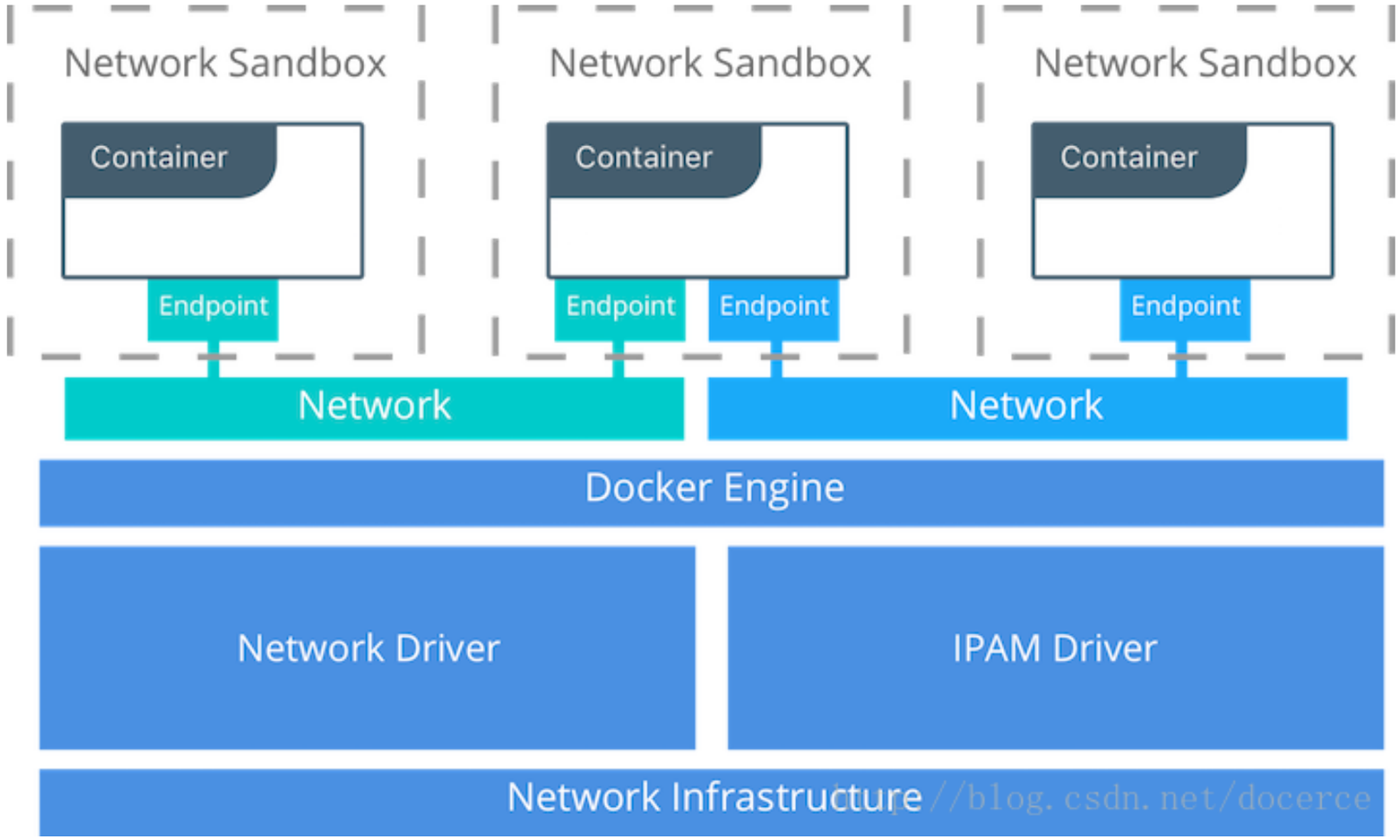
Docker容器中运行的应用，要让外部访问，可以通过**-P 或-p** 参数来指定端口映射。

在上面的例子中就是绑定了宿主机的3306端口到容器的3306端口。

可以通过 docker port来查看当前映射的端口配置和绑定的地址

```
[jiangqiao@localhost logs]$ docker port b3b8139754c7 3306
 0.0.0.0:3306
```

Docker网络模型



容器网络模型（CNM）

- CNM的理念是提供可以跨不同网络基础架构、可实现移植的应用。这种模型虚拟了所有操作系统和基础架构的不可知性，所以应用无论在何种基础架构上都可以有一样的体验；

libnetwork

- libnetwork是基于CNM模型将docker engine 和 lib container中网络相关代码抽离出来，合成的一个单独的库；
- 基于CNM模型概念，进一步对Docker的网络结构细分为：network、sandbox、endpoint三层；

Docker网络模型

CNM结构

■ sandbox

sandbox 实现了容器内部的网络栈，它定义了容器的虚拟网卡，路由表和DNS等配置，其实就是一个标准的linux network namespace实现。

■ network

network 是一个抽象的概念，可以理解为一个网络插件，或者是网络的Driver，是由Linux桥接，Vlan等来实现的；例如 Docker中原生的Driver包括单主机的none、host、bridge，joined container 和多主机的overlay、macvlan，第三方Driver包括多主机的 flannel、weave、calico 等。

网络上收集了所有连接在其上的端点，并实现了这些端点的互连接。

■ endpoint

network实现了一个第三方的网络栈，sanbox则实现了容器内部的网络栈，而它们的互连则是通过endpoint，endpoint实现了veth pair（虚拟网络设备）；一个endpoint就表示一对veth pair，一端挂在容器中，另一端挂在network中。

Docker网络模式

“不同的网络方案如何集成到Docker网络模型中而不改变原有结构？”

Docker提供的网络模式:

单主机网络:

- Bridge
- Host
- Container
- None

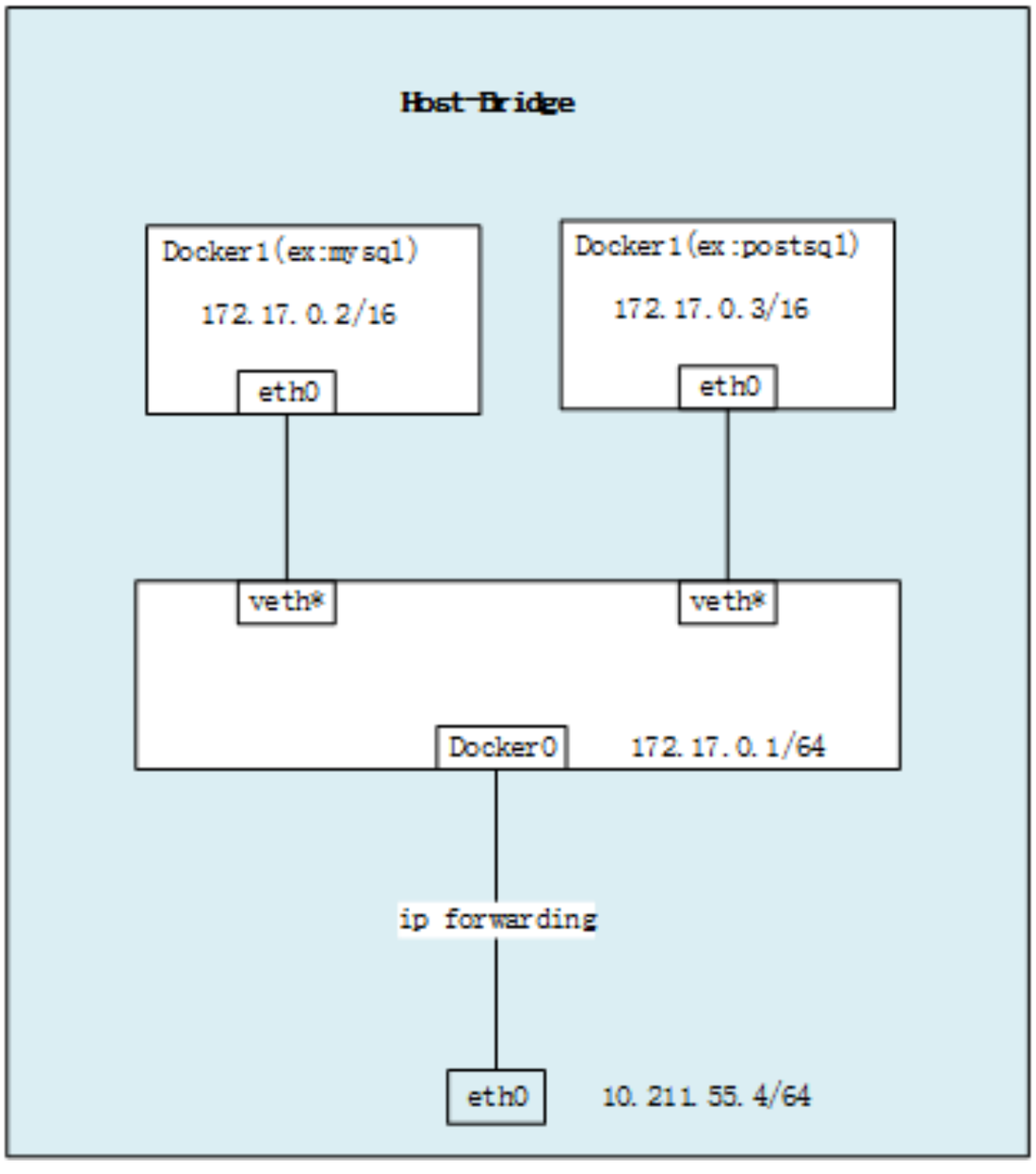
多主机网络

- overlay、macvlan、flannel、weave、calico

网络模式-Bridge

docker run --net=bridge --name mysql -p 3306:3306 -v /data/mysql/data:/var/lib/mysql -v /data/mysql/conf:/etc/mysql/conf.d -e MYSQL_ROOT_PASSWORD=123456 -d mysql

```
docker run --net=bridge --name postgres1 -e POSTGRES_PASSWORD=123456 -p 54321:5432 -d postgres:9.4
```



- 在*Bridge*模式下，*docker*进程启动时会在主机上创建一个名为*docker0*的虚拟网桥，此主机上的*docker*容器会连接到这个虚拟网桥上;
- 虚拟网桥的工作方式和物理交换机类似，这样主机上的所有容器就通过交换机连在了一个二层网络中;
- *docker0*网桥从子网中分配一个*IP*给容器使用，并设置*docker0*的*IP*地址为容器的默认网关;
- 容器启动后会在主机上创建一对虚拟网卡*veth pair*设备，*docker*将*veth pair*设备的一端放在新创建的容器中，并命名为*eth0*作为容器的网卡，另一端放在主机中，以*vethxxx*这样类似的名字命名，并将这个设备加入到*docker0*网桥中;

网络模式-Bridge

```
[jiangqiao@localhost data]$ ifconfig
docker0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 172.17.0.1  netmask 255.255.0.0  broadcast 172.17.255.255
        inet6 fe80::42:1aff:fec1:b1d8  prefixlen 64  scopeid 0x20<link>
        ether 02:42:1a:c1:b1:d8  txqueuelen 0  (Ethernet)
        RX packets 239  bytes 26531 (25.9 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 387  bytes 31872 (31.1 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.211.55.4  netmask 255.255.255.0  broadcast 10.211.55.255
        inet6 fe80::21c:42ff:fea4:e10a  prefixlen 64  scopeid 0x20<link>
        inet6 fdb2:2c26:f4e4:0:21c:42ff:fea4:e10a  prefixlen 64  scopeid 0x0<global>
        ether 00:1c:42:a4:e1:0a  txqueuelen 1000  (Ethernet)
        RX packets 370649  bytes 528536845 (504.0 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 125798  bytes 7728069 (7.3 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 92  bytes 7776 (7.5 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 92  bytes 7776 (7.5 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

veth38f43dc: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet6 fe80::a83c:3dff:fe70:9ab0  prefixlen 64  scopeid 0x20<link>
        ether aa:3c:3d:70:9a:b0  txqueuelen 0  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 9  bytes 729 (729.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

veth71b8b5b: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet6 fe80::cda:6aff:fea2:737b  prefixlen 64  scopeid 0x20<link>
        ether 0e:da:6a:a2:73:7b  txqueuelen 0  (Ethernet)
        RX packets 53  bytes 9084 (8.8 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 87  bytes 7242 (7.0 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

virbr0: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
        inet 192.168.122.1  netmask 255.255.255.0  broadcast 192.168.122.255
        ether 52:54:00:82:75:69  txqueuelen 1000  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

```
[jiangqiao@localhost data]$ docker inspect e2807260e87b
```

```
,
"NetworkSettings": {
    "Bridge": "",
    "SandboxID": "18106dc7ed0c82e21b9f1ec75d7a2d1243a0c3349f6591c45c8d726108c12c57",
    "HairpinMode": false,
    "LinkLocalIPv6Address": "",
    "LinkLocalIPv6PrefixLen": 0,
    "Ports": {
        "5432/tcp": [
            {
                "HostIp": "0.0.0.0",
                "HostPort": "54321"
            }
        ]
    },
    "SandboxKey": "/var/run/docker/netns/18106dc7ed0c",
    "SecondaryIPAddresses": null,
    "SecondaryIPv6Addresses": null,
    "EndpointID": "b157af56181afef7d89e0bc117040bc2741ed88106d4684f96550074c4e797c4",
    "Gateway": "172.17.0.1",
    "GlobalIPv6Address": "",
    "GlobalIPv6PrefixLen": 0,
    "IPAddress": "172.17.0.3",
    "IPPrefixLen": 16,
    "IPv6Gateway": "",
    "MacAddress": "02:42:ac:11:00:03",
    "Networks": {
        "bridge": {
            "IPAMConfig": null,
            "Links": null,
            "Aliases": null,
            "NetworkID": "35aa8a01e83ae1739d809b1f89263c4bc820d2ca8bf6a8ef2929d3385f17a1fa",
            "EndpointID": "b157af56181afef7d89e0bc117040bc2741ed88106d4684f96550074c4e797c4",
            "Gateway": "172.17.0.1",
            "IPAddress": "172.17.0.3",
            "IPPrefixLen": 16,
            "IPv6Gateway": "",
            "GlobalIPv6Address": "",
            "GlobalIPv6PrefixLen": 0,
            "MacAddress": "02:42:ac:11:00:03",
            "DriverOpts": null
        }
    }
}
```

Docker端口映射

Bridge模式时Docker的默认网络模式，不写–net参数，就是bridge模式。在这种模式下，容器如果需要联网，则需要采用NAT（网络地址端口转换）方式；

（1）、默认情况下，容器可以通过SNAT（出去的时候改变源地址，回来的时候改变目的地址）的方式主动访问到外部网络的连接。

sudo iptables –t nat –nL

```
Chain POSTROUTING (policy ACCEPT)
target     prot opt source               destination
MASQUERADE  all  --  172.17.0.0/16        0.0.0.0/0
RETURN     all  --  192.168.122.0/24     224.0.0.0/24
RETURN     all  --  192.168.122.0/24     255.255.255.255
MASQUERADE  tcp  --  192.168.122.0/24    !192.168.122.0/24     masq ports: 1024-65535
MASQUERADE  udp  --  192.168.122.0/24    !192.168.122.0/24     masq ports: 1024-65535
MASQUERADE  all  --  192.168.122.0/24    !192.168.122.0/24
POSTROUTING_direct  all  --  0.0.0.0/0            0.0.0.0/0
POSTROUTING_ZONES_SOURCE  all  --  0.0.0.0/0            0.0.0.0/0
POSTROUTING_ZONES  all  --  0.0.0.0/0            0.0.0.0/0
MASQUERADE  tcp  --  172.17.0.2           172.17.0.2           tcp dpt:3306
MASQUERADE  tcp  --  172.17.0.3           172.17.0.3           tcp dpt:5432
```

上述规则将所有源地址在172.17.0.0/16网端的容器，通过宿主机网卡IP进行外部网络访问。

Docker端口映射

(2)、默认情况下，外部网络无法访问到容器，可以在docker run时通过 -p或-P参数启动。其实就是在本地的iptable的nat表中添加相应的规则。

sudo iptables -t nat -nL

```
Chain DOCKER (2 references)
target     prot opt source               destination
RETURN     all  --  0.0.0.0/0            0.0.0.0/0
DNAT       tcp  --  0.0.0.0/0            0.0.0.0/0           tcp dpt:3306 to:172.17.0.2:3306
DNAT       tcp  --  0.0.0.0/0            0.0.0.0/0           tcp dpt:54321 to:172.17.0.3:5432
```

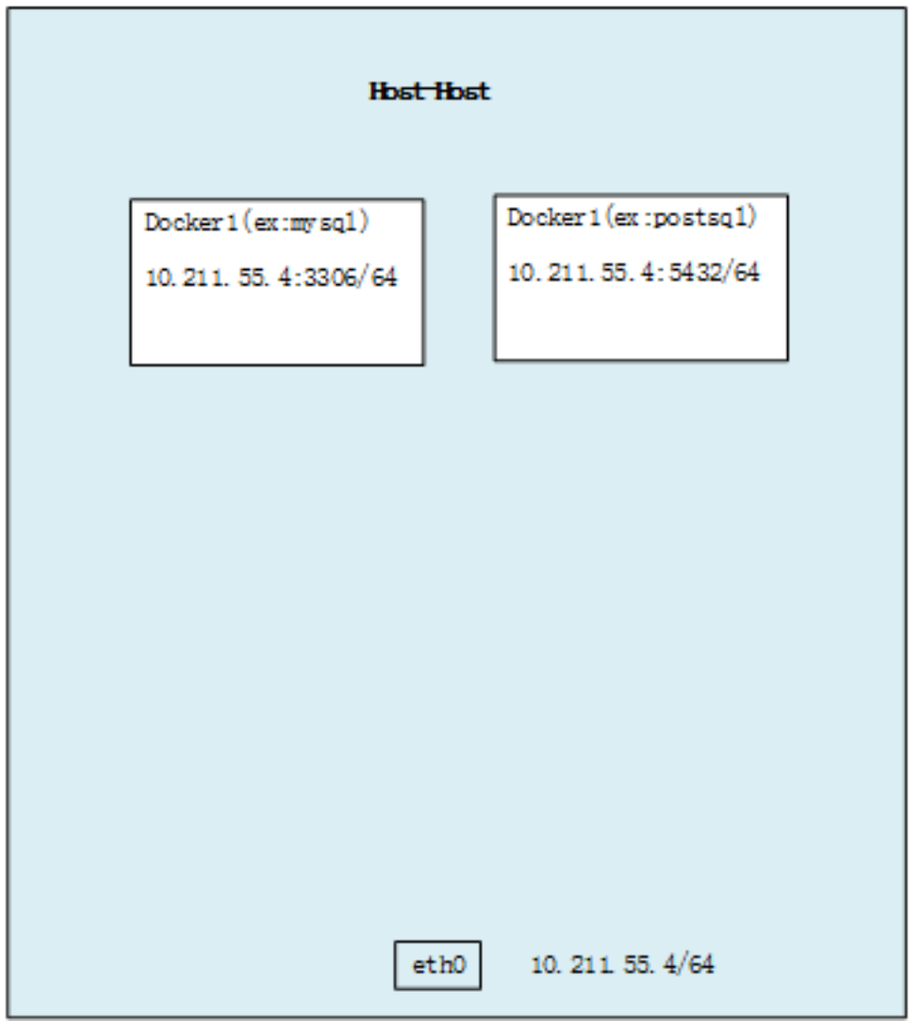
这里的规则映射了**0.0.0.0**，意味着将接收主机来自所有接口的流量。可以通过 -p IP:host_port:container 或者 -p IP::port 来指定允许访问容器的主机上的IP、端口等，以指定更严格的规则；

```
[[jiangqiao@localhost data]$ docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS                              NAMES
e2807260e87b        postgres:9.4        "docker-entrypoint.s…"   4 hours ago         Up 4 hours          0.0.0.0:54321->5432/tcp            postgres1
f94c2b916597        mysql               "docker-entrypoint.s…"   5 hours ago         Up 5 hours          0.0.0.0:3306->3306/tcp, 33060/tcp  mysql
```

网络模式-Host

docker run **--net=host** --name mysql -p 3306:3306 -v /data/mysql/data:/var/lib/mysql -v /data/mysql/conf:/etc/mysql/conf.d -e MYSQL_ROOT_PASSWORD=123456 -d mysql

```
docker run --net=host --name postgres1 -e POSTGRES_PASSWORD=123456 -p 54321:5432 -d postgres:9.4
```



- 在*Host*模式下，容器将不会获得一个独立的*Network Namespace*,而是和宿主机共用一个*Network Namespace*；
- 容器也不会虚拟出自己的网卡，配置自己的*IP*，而是使用宿主机的*IP*和端口；
- 但是，容器的其他方面，如文件系统、进程列表等还是通过相应的*namespace*与宿主机隔离的；

网络模式-Host

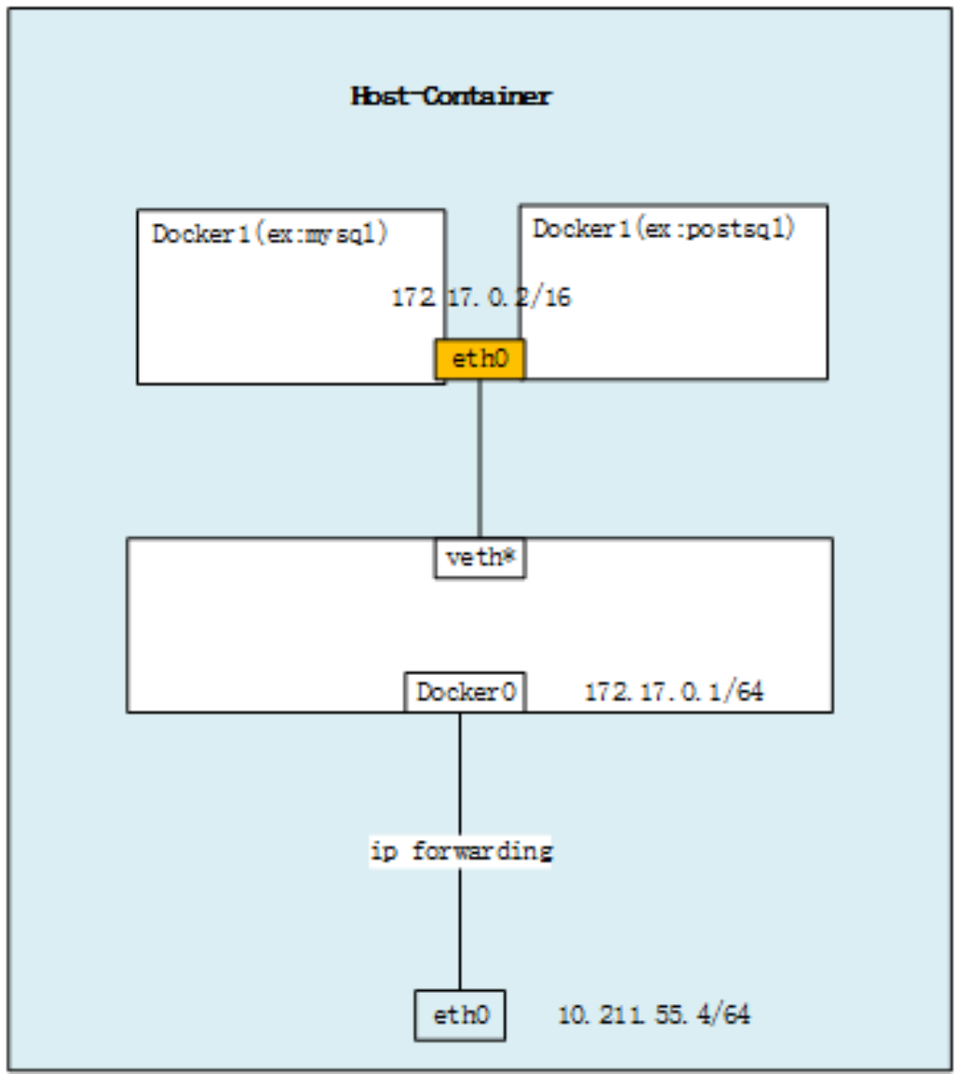
```
[jiangqiao@localhost data]$ ifconfig
docker0: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
        inet 172.17.0.1  netmask 255.255.0.0  broadcast 172.17.255.255
        inet6 fe80::42:1aff:fec1:b1d8  prefixlen 64  scopeid 0x20<link>
        ether 02:42:1a:c1:b1:d8  txqueuelen 0  (Ethernet)
        RX packets 245  bytes 26795 (26.1 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 396  bytes 32463 (31.7 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.211.55.4  netmask 255.255.255.0  broadcast 10.211.55.255
        inet6 fe80::21c:42ff:fea4:e10a  prefixlen 64  scopeid 0x20<link>
        inet6 fdb2:2c26:f4e4:0:21c:42ff:fea4:e10a  prefixlen 64  scopeid 0x0<globa
        ether 00:1c:42:a4:e1:0a  txqueuelen 1000  (Ethernet)
        RX packets 372579  bytes 528706119 (504.2 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 126828  bytes 7888064 (7.5 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 209  bytes 45794 (44.7 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 209  bytes 45794 (44.7 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

virbr0: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
        inet 192.168.122.1  netmask 255.255.255.0  broadcast 192.168.122.255
        ether 52:54:00:82:75:69  txqueuelen 1000  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

```
"NetworkSettings": {
    "Bridge": "",
    "SandboxID": "fba17826a9d967d5a7d814210bdd1034af23fac1cd6b4bdeffddcc6404dc2736",
    "HairpinMode": false,
    "LinkLocalIPv6Address": "",
    "LinkLocalIPv6PrefixLen": 0,
    "Ports": {},
    "SandboxKey": "/var/run/docker/netns/default",
    "SecondaryIPAddresses": null,
    "SecondaryIPv6Addresses": null,
    "EndpointID": "",
    "Gateway": "",
    "GlobalIPv6Address": "",
    "GlobalIPv6PrefixLen": 0,
    "IPAddress": "",
    "IPPrefixLen": 0,
    "IPv6Gateway": "",
    "MacAddress": "",
    "Networks": {
        "host": {
            "IPAMConfig": null,
            "Links": null,
            "Aliases": null,
            "NetworkID": "4f5bf4f7a37beb738ad487357c77634a4881f1714fea128b2af6fab9239bb330",
            "EndpointID": "2a7ed80de47b29c58433aaeaafaa0a34778f6c7c363099afac5a6f8b0688dfb0",
            "Gateway": "",
            "IPAddress": "",
            "IPPrefixLen": 0,
            "IPv6Gateway": "",
            "GlobalIPv6Address": "",
            "GlobalIPv6PrefixLen": 0,
            "MacAddress": "",
            "DriverOpts": null
        }
    }
}
```

网络模式-Container

docker run **--net=bridge** --name mysql -p 3306:3306 -p 5432:5432 -v /data/mysql/data:/var/lib/mysql -v /data/mysql/conf:/etc/mysql/conf.d -e MYSQL_ROOT_PASSWORD=123456 -d mysql

```
docker run --net=container:mysql --name postgres1 -e POSTGRES_PASSWORD=123456 -d postgres:9.4
```



- 在*Container*模式下，可以指定新创建的容器和已经存在的一个容器共享一个*Network Namespace*，而不是和宿主机共享；
- 新创建的容器不会创建自己的网卡、配置自己的*IP*，而是和一个指定的容器共享*IP*、端口范围；
- 两个容器除了网络方面，其他的如文件系统、进程列表等还是通过不同的*namespace*进行隔离；
- 两个容器的进程可以通过*lo*网卡设备进行通信；

网络模式-Container

```
[root@localhost ~]# ifconfig
docker0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 172.17.0.1  netmask 255.255.0.0  broadcast 172.17.255.255
        inet6 fe80::42:1eff:fe8a:19ee  prefixlen 64  scopeid 0x20<link>
        ether 02:42:1e:8a:19:ee  txqueuelen 0  (Ethernet)
        RX packets 25  bytes 2552 (2.4 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 58  bytes 5520 (5.3 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.211.55.4  netmask 255.255.255.0  broadcast 10.211.55.255
        inet6 fe80::21c:42ff:fea4:e10a  prefixlen 64  scopeid 0x20<link>
        inet6 fdb2:2c26:f4e4:0:21c:42ff:fea4:e10a  prefixlen 64  scopeid 0x0<gl
        ether 00:1c:42:a4:e1:0a  txqueuelen 1000  (Ethernet)
        RX packets 382667  bytes 529681292 (505.1 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 132426  bytes 11125198 (10.6 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 1498  bytes 475914 (464.7 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 1498  bytes 475914 (464.7 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

veth410fce2: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet6 fe80::bc3c:edff:fe21:c866  prefixlen 64  scopeid 0x20<link>
        ether be:3c:ed:21:c8:66  txqueuelen 0  (Ethernet)
        RX packets 25  bytes 2902 (2.8 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 66  bytes 6168 (6.0 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

virbr0: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
        inet 192.168.122.1  netmask 255.255.255.0  broadcast 192.168.122.255
        ether 52:54:00:82:75:69  txqueuelen 1000  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

```
"NetworkSettings": {
    "Bridge": "",
    "SandboxID": "c6d6e4cc394fb5f502128910b3e182091d87057d6b7d2383e9c018861dfc6631",
    "HairpinMode": false,
    "LinkLocalIPv6Address": "",
    "LinkLocalIPv6PrefixLen": 0,
    "Ports": {
        "3306/tcp": [
            {
                "HostIp": "0.0.0.0",
                "HostPort": "3306"
            }
        ],
        "33060/tcp": null
    },
    "SandboxKey": "/var/run/docker/netns/c6d6e4cc394f",
    "SecondaryIPAddresses": null,
    "SecondaryIPv6Addresses": null,
    "EndpointID": "0837a17c6649f3e7b881a39841305256e2c0d8664aa72807abae08c1f451185c",
    "Gateway": "172.17.0.1",
    "GlobalIPv6Address": "",
    "GlobalIPv6PrefixLen": 0,
    "IPAddress": "172.17.0.2",
    "IPPrefixLen": 16,
    "IPv6Gateway": "",
    "MacAddress": "02:42:ac:11:00:02",
    "Networks": {
        "bridge": {
            "IPAMConfig": null,
            "Links": null,
            "Aliases": null,
            "NetworkID": "31d8f1be80740665ad6e483361b3f0a6e62786147877025e11f281a47fd15f8c",
            "EndpointID": "0837a17c6649f3e7b881a39841305256e2c0d8664aa72807abae08c1f451185c",
            "Gateway": "172.17.0.1",
            "IPAddress": "172.17.0.2",
            "IPPrefixLen": 16,
            "IPv6Gateway": "",
            "GlobalIPv6Address": "",
            "GlobalIPv6PrefixLen": 0,
            "MacAddress": "02:42:ac:11:00:02",
            "DriverOpts": null
        }
    }
}
```

指定容器

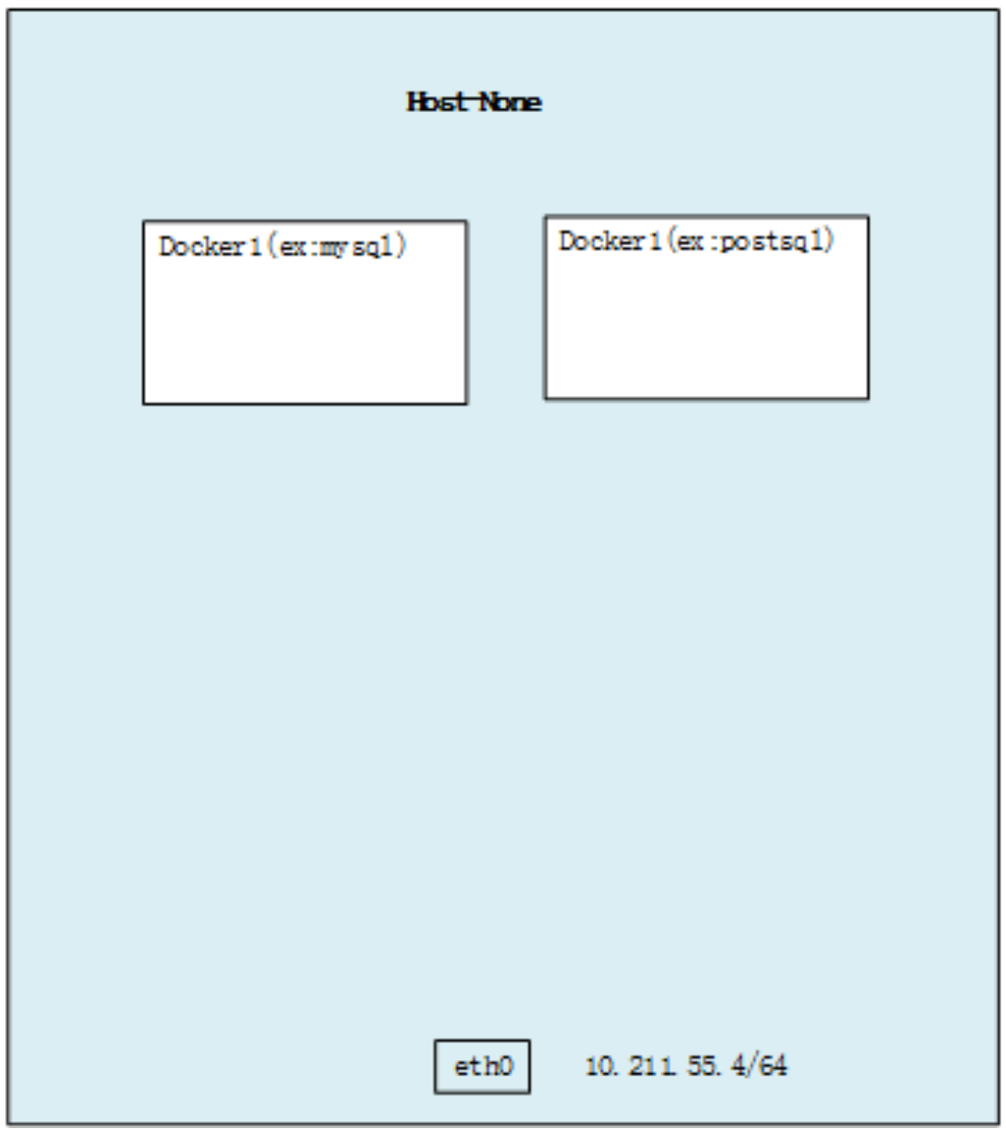
```
"NetworkSettings": {
    "Bridge": "",
    "SandboxID": "",
    "HairpinMode": false,
    "LinkLocalIPv6Address": "",
    "LinkLocalIPv6PrefixLen": 0,
    "Ports": {},
    "SandboxKey": "",
    "SecondaryIPAddresses": null,
    "SecondaryIPv6Addresses": null,
    "EndpointID": "",
    "Gateway": "",
    "GlobalIPv6Address": "",
    "GlobalIPv6PrefixLen": 0,
    "IPAddress": "",
    "IPPrefixLen": 0,
    "IPv6Gateway": "",
    "MacAddress": "",
    "Networks": {}
}
```

被指定容器

网络模式-None

docker run **--net=none** --name mysql -p 3306:3306 -p 5432:5432 -v /data/mysql/data:/var/lib/mysql -v /data/mysql/conf:/etc/mysql/conf.d -e MYSQL_ROOT_PASSWORD=123456 -d mysql

```
docker run --net=none --name postgres1 -e POSTGRES_PASSWORD=123456 -d postgres:9.4
```



- 在*None*模式下，*Docker*容器拥有自己的*Network Namespace*，但是，并不为*Docker*容器进行任何网络配置；
- 这个*Docker*容器没有网卡、*IP*、路由等信息，需要我们自己为*Docker*容器添加网卡、配置*IP*等；

网络模式-None

```
docker0: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
        inet 172.17.0.1  netmask 255.255.0.0  broadcast 172.17.255.255
        inet6 fe80::42:f1ff:feef:c35f  prefixlen 64  scopeid 0x20<link>
        ether 02:42:f1:ef:c3:5f  txqueuelen 0  (Ethernet)
        RX packets 38  bytes 16061 (15.6 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 86  bytes 9139 (8.9 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.211.55.4  netmask 255.255.255.0  broadcast 10.211.55.255
        inet6 fe80::21c:42ff:fea4:e10a  prefixlen 64  scopeid 0x20<link>
        inet6 fdb2:2c26:f4e4:0:21c:42ff:fea4:e10a  prefixlen 64  scopeid 0x0<global>
        ether 00:1c:42:a4:e1:0a  txqueuelen 1000  (Ethernet)
        RX packets 385229  bytes 529902935 (505.3 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 133751  bytes 11330284 (10.8 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 1498  bytes 475914 (464.7 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 1498  bytes 475914 (464.7 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

virbr0: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
        inet 192.168.122.1  netmask 255.255.255.0  broadcast 192.168.122.255
        ether 52:54:00:82:75:69  txqueuelen 1000  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Mysql容器


```
"NetworkSettings": {
    "Bridge": "",
    "SandboxID": "cd6dd4ff91585bca6e3253da53a8091f9a0fd2f0d2b1059d69c81129746ac7ac",
    "HairpinMode": false,
    "LinkLocalIPv6Address": "",
    "LinkLocalIPv6PrefixLen": 0,
    "Ports": {},
    "SandboxKey": "/var/run/docker/netns/cd6dd4ff9158",
    "SecondaryIPAddresses": null,
    "SecondaryIPv6Addresses": null,
    "EndpointID": "",
    "Gateway": "",
    "GlobalIPv6Address": "",
    "GlobalIPv6PrefixLen": 0,
    "IPAddress": "",
    "IPPrefixLen": 0,
    "IPv6Gateway": "",
    "MacAddress": "",
    "Networks": {
        "none": {
            "IPAMConfig": null,
            "Links": null,
            "Aliases": null,
            "NetworkID": "442d24093083bb2d6bf5655b04dc0345ce2acdabb1e57c10b8476db5833cb705",
            "EndpointID": "6ac0d425fd76777ff981a47d5ea4282e9c7b34d3b84e0a75928cf6e2e00225a7",
            "Gateway": "",
            "IPAddress": "",
            "IPPrefixLen": 0,
            "IPv6Gateway": "",
            "GlobalIPv6Address": "",
            "GlobalIPv6PrefixLen": 0,
            "MacAddress": "",
            "DriverOpts": null
        }
    }
}
```


PG容器


```
,,
"NetworkSettings": {
    "Bridge": "",
    "SandboxID": "e3637cea3114abddc332931238bce563efd6aed1b8e4486367c26159191243d2",
    "HairpinMode": false,
    "LinkLocalIPv6Address": "",
    "LinkLocalIPv6PrefixLen": 0,
    "Ports": {},
    "SandboxKey": "/var/run/docker/netns/e3637cea3114",
    "SecondaryIPAddresses": null,
    "SecondaryIPv6Addresses": null,
    "EndpointID": "",
    "Gateway": "",
    "GlobalIPv6Address": "",
    "GlobalIPv6PrefixLen": 0,
    "IPAddress": "",
    "IPPrefixLen": 0,
    "IPv6Gateway": "",
    "MacAddress": "",
    "Networks": {
        "none": {
            "IPAMConfig": null,
            "Links": null,
            "Aliases": null,
            "NetworkID": "442d24093083bb2d6bf5655b04dc0345ce2acdabb1e57c10b8476db5833cb705",
            "EndpointID": "d28e263fb2df44c17498cfe3b540d2332d079063be84750b2a543d3932e17105",
            "Gateway": "",
            "IPAddress": "",
            "IPPrefixLen": 0,
            "IPv6Gateway": "",
            "GlobalIPv6Address": "",
            "GlobalIPv6PrefixLen": 0,
            "MacAddress": "",
            "DriverOpts": null
        }
    }
}
```

网络模式-跨主机通信

Docker默认的网络环境下，单台主机的Docker容器可以通过docker0网桥直接通信，而不同主机上的Docker容器只能在宿主机上做端口映射进行通信。

这种端口映射方式对很多集群应用来说及不方便，如果能让Docker容器之间直接使用自己的IP地址通信，则会解决很多问题。

目前多主机通信方案按照原理可划分为：

- **直接路由方式；**
- **桥接方式（如pipework）；**
- **Overlay隧道方式（如flannel、ovs+gre）；**

网络模式-Pipework

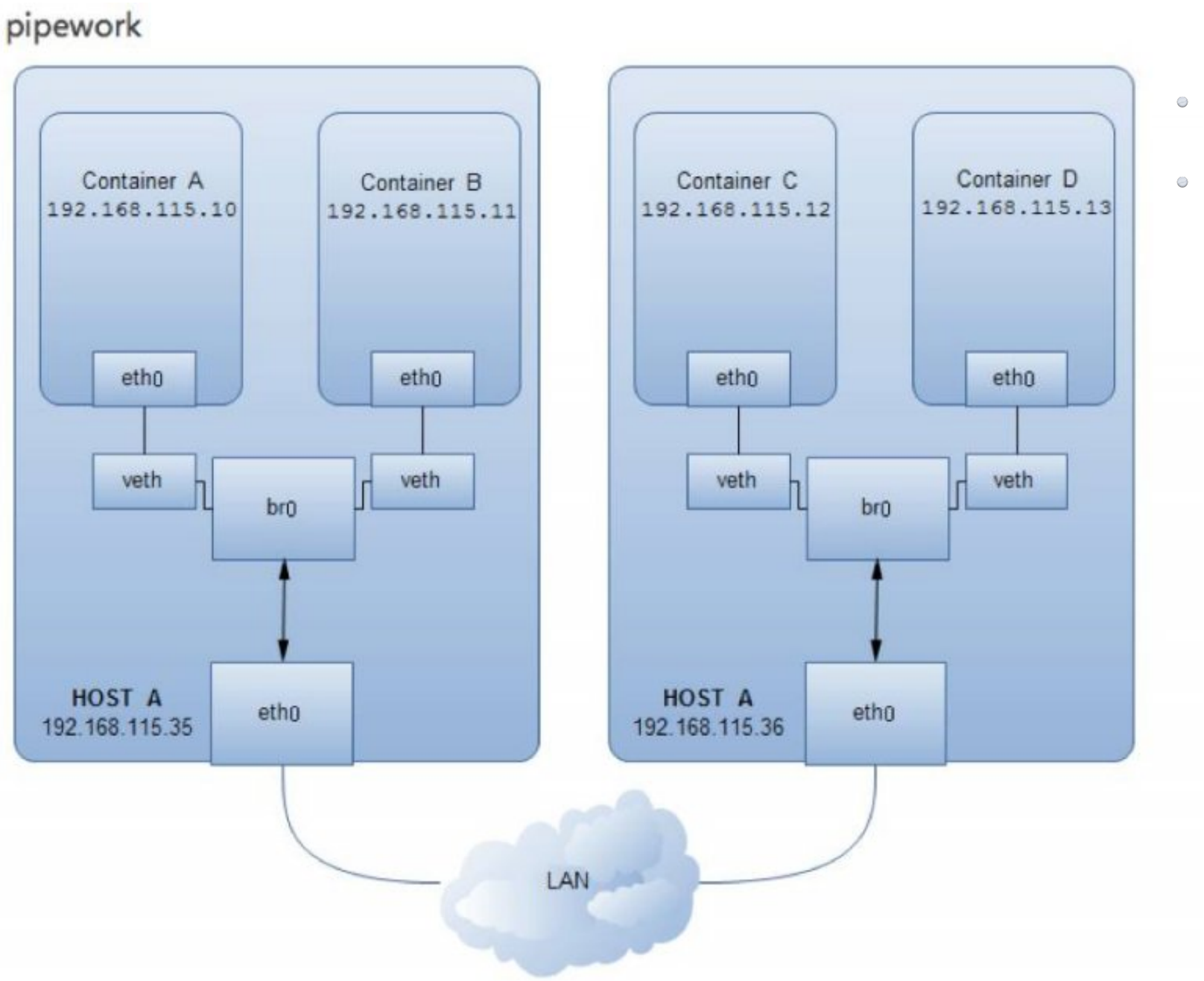
Pipework是一个简易的Docker容器网络配置工具。由200多行shell脚本实现，可以通过使用ip、brctl、ovs-vsctl等命令来为Docker容器配置自定义的网桥、网卡、路由等。

```
sudo service docker stop
sudo ip link set dev docker0 down
sudo brctl delbr docker0

sudo brctl addbr br0
sudo ip addr add 10.211.55.1/24 dev br0
sudo ip link set dev br0 up

ip addr show br0

echo 'DOCKER_OPTS="-b=bridge0"' >> /etc/default/docker
```

```
[Service]
Type=notify
# the default is not to use systemd for c
# exists and systemd currently does not s
# for containers run by docker
ExecStart=/usr/bin/dockerd -b br0
ExecReload=/bin/kill -s HUP $MAINPID
```

```
sudo service docker start
```

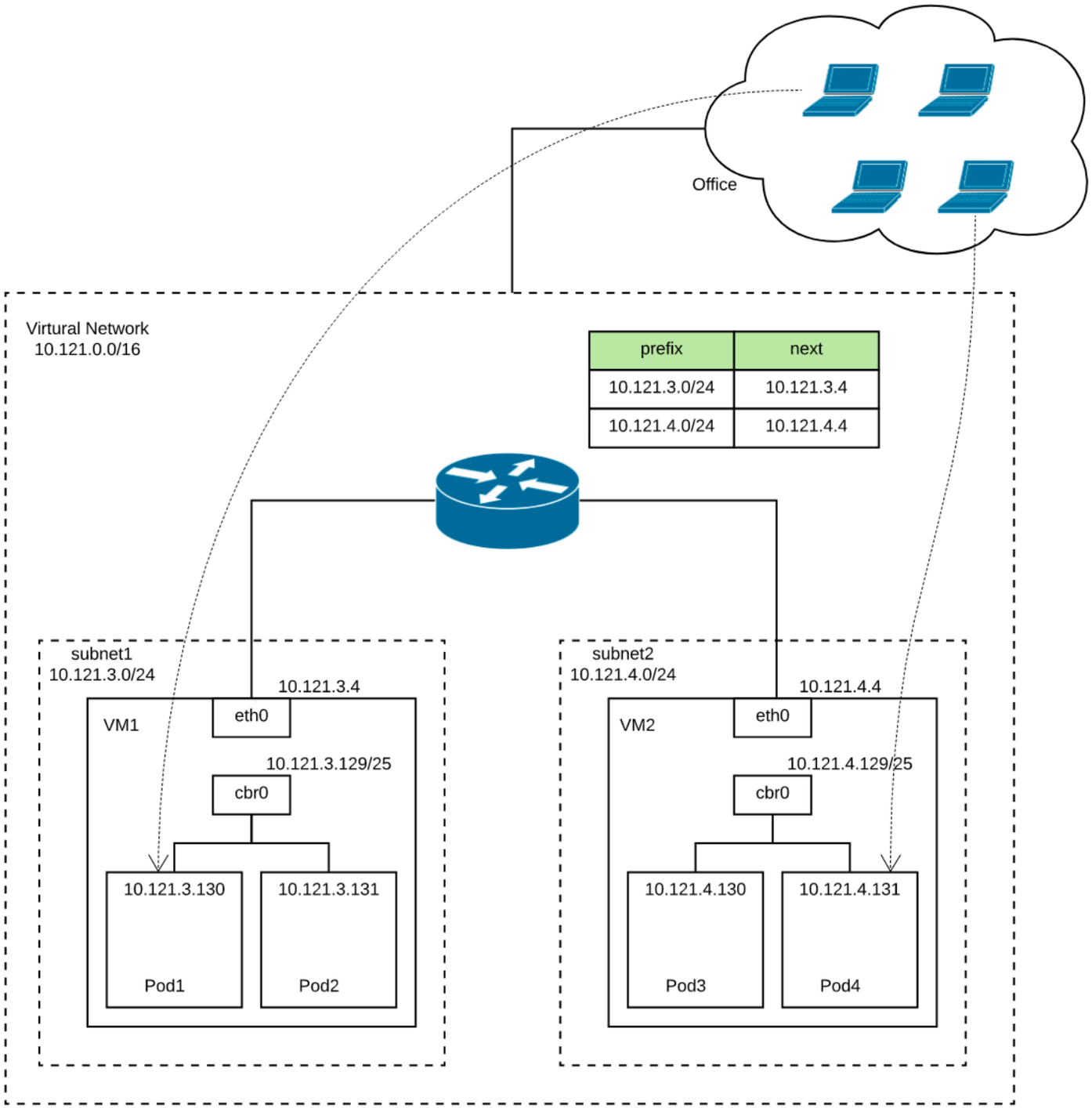
网络模式-Pipework



- 使用新建的*br0*网桥代替缺省的*docker0*网桥；
- *br0*网桥与*docker0*的区别在于*br0*和宿主机*eth0*之间是*veth pair*；

不同容器之间的通信可以借助于 pipework 这个工具给 docker 容器新建虚拟网卡并绑定 IP 桥接到 br0

Mobike实践



扁平网络，集群之间容器都属于相同网段，使得应用之间网络访问速度加快，同时减少管理负担

参考资料

https://wiki.mobike.com/pages/viewpage.action?pageId=29278470

Q&A