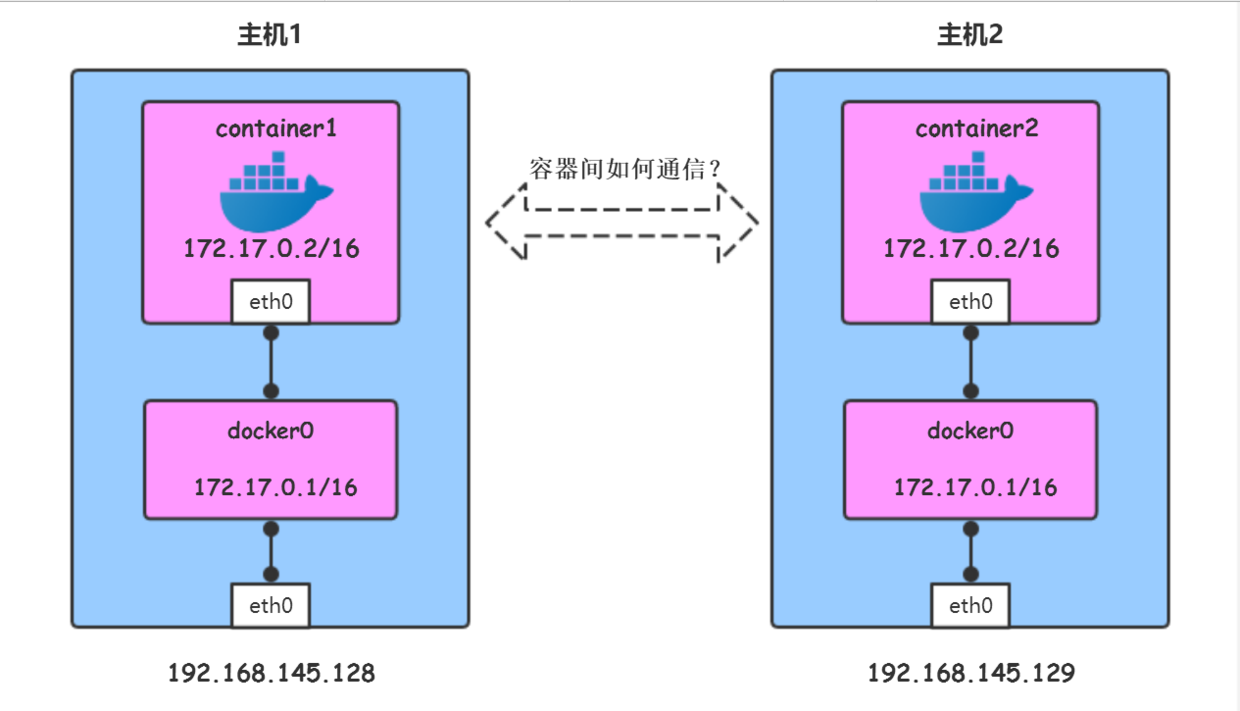
背景

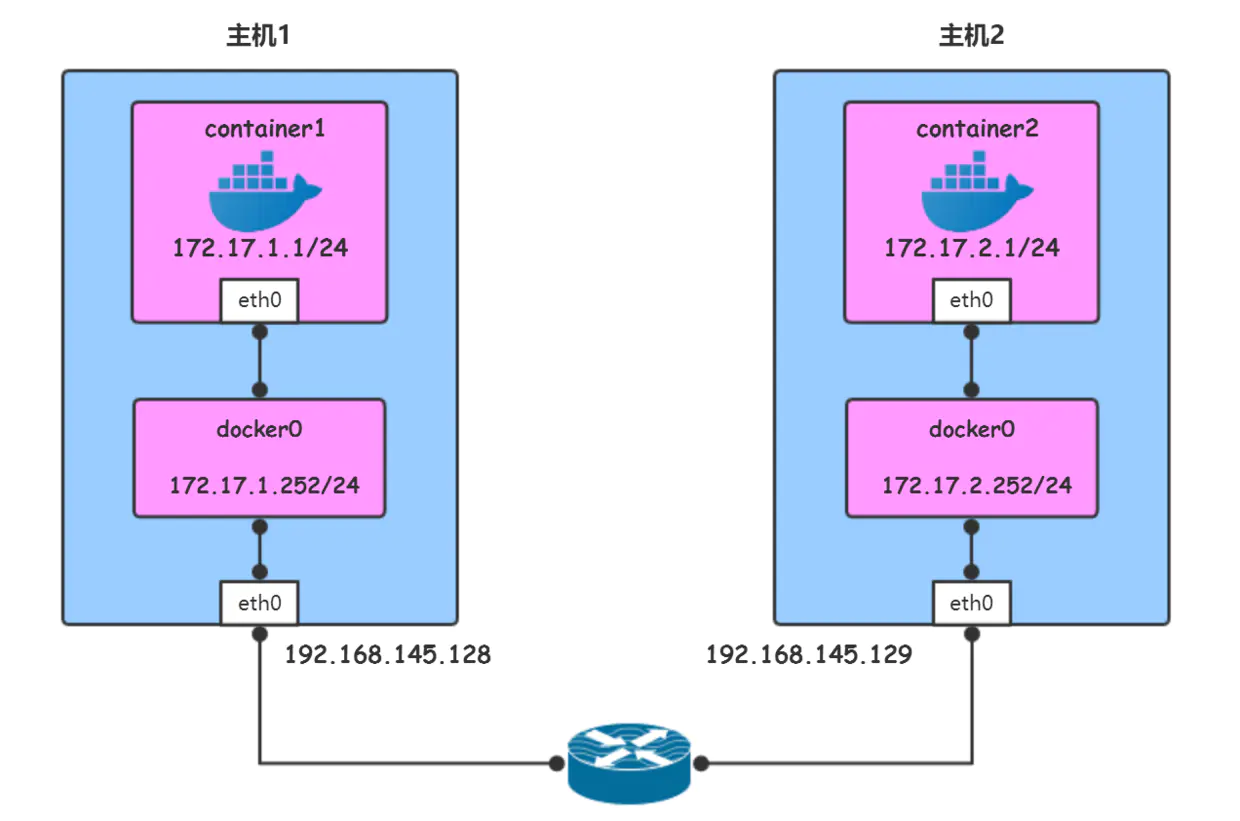
如下图所示，有两个物理主机 1 和主机 2 ，在各自宿主机上启动一个 nginx 容器，启动成功之后，两个容器分别运行在两个宿主机之上，默认的 ip 地址分配如图所示，这也是 docker 自身默认的网络。

此时两台主机上的 docker 容器如何直接通过 ip 地址进行通信？

一种直接想到的方案便是通过分别在各自主机中 **添加路由** 来实现两个 nginx 容器之间的直接通信。

方案原理

由于使用容器的 ip 进行路由，就需要避免不同主机上的容器使用了相同的 ip，为此应该为不同的主机 docker 分配不同的子网。于是构造一下两个容器之间通信的路由方案，如下图所示。



**各项配置如下：**

* 主机 1 的 ip 地址为：192.168.145.128
* 主机 2 的 ip 地址为：192.168.145.129
* 为主机 1 上的 docker 容器分配的子网：172.17.1.0/24
* 为主机 2 上的 docker 容器分配的子网：172.17.2.0/24

这样配置之后，两个主机上的 docker 容器就肯定不会使用相同的 ip 地址从而避免了 ip 冲突。

我们接下来 **定义两条路由规则** 即可：

* 所有目的地址为 172.17.1.0/24 的包都被转发到主机 1 上
* 所有目的地址为 172.17.2.0/24 的包都被转发到主机 2 上

**综上所述，数据包在两个容器间的传递过程如下：**

* 从 container1 发往 container2 的数据包，首先发往 container1 的“网关” docker0，然后通过查找主机 1 的路由得知需要将数据包发给主机 2 ，数据包到达主机 2 后再转发给主机 2 的 docker0，最后由其将数据包转到 container2 中；反向原理相同，不再赘述。

## **实际试验**

**1、分别对主机 1 和主机 2 上的 docker0 进行配置**

编辑主机 1 上的 /etc/docker/daemon.json 文件，添加内容{"bip":"172.17.1.252/24"}

编辑主机 2 上的 /etc/docker/daemon.json 文件，添加内容{"bip":"172.17.2.252/24"}

**2、重启docker服务**

主机 1 和主机 2 上均执行如下命令重启 docker 服务以使修改后的 docker0 网段生效

systemctl restart docker

**3、添加路由规则**

主机 1 上添加路由规则如下：

route add -net 172.17.2.0 netmask 255.255.255.0 gw 192.168.145.129

主机 2 上添加路由规则如下：

route add -net 172.17.1.0 netmask 255.255.255.0 gw 192.168.145.128

**4、配置 iptables 规则（可不配）**

主机 1 上添加如下规则：

iptables -t nat -F POSTROUTING

iptables -t nat -A POSTROUTING -s 172.17.1.0/24 ! -d 172.17.0.0/16 -j MASQUERADE

主机 2 上添加如下规则：

iptables -t nat -F POSTROUTING

iptables -t nat -A POSTROUTING -s 172.17.2.0/24 ! -d 172.17.0.0/16 -j MASQUERADE

**5、启动容器**

主机 1 上启动 nginx 容器：

docker run -it --name nginx nginx /bin/bash

主机 2 上启动 nginx 容器：

docker run -it --name nginx nginx /bin/bash

**6、容器间直接通信**