**一、宿主机与容器通信的建立**

宿主机与容器的通信，主要依赖于一个在主机上创建的网卡，以及一对veth接口。这个对veth接口一段挂载在宿主机的网卡上，另一段挂载容器的网络命名空间内。并且设置路由规则，使得在宿主机上发往该网段的数据由veth接口挂载的网卡处理。

从某种意义来看，docker的bridge网络其实就是宿主机下的一个自定义子网。

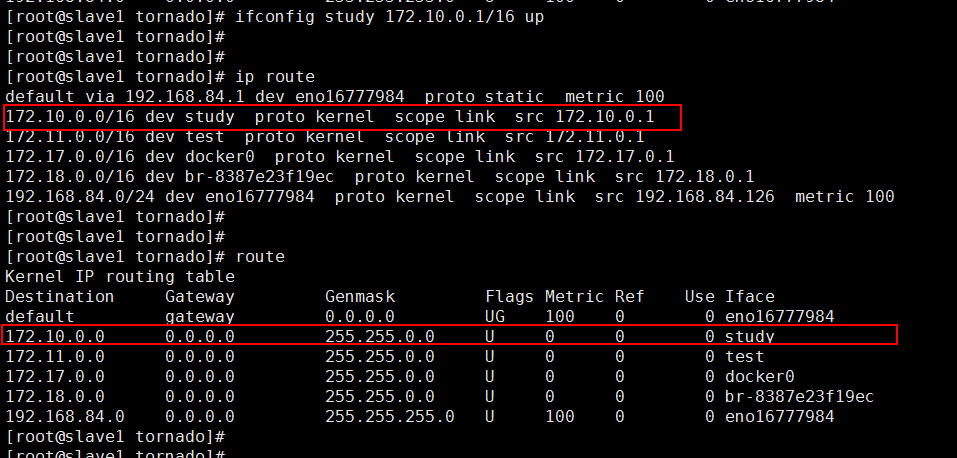
1、docker在创建bridge网络的时候会创建一个的网卡，并给该网卡设置一个网段。将路由规则修改将所有发往该网段的流量都转发到该网卡上。

模拟实验如下：

①创建网桥

②为网桥分配网段

clipboard.png

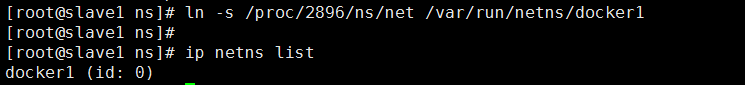


2、将容器网络命名空间放到ip netns命令可以管理的位置

/var/run/netns/

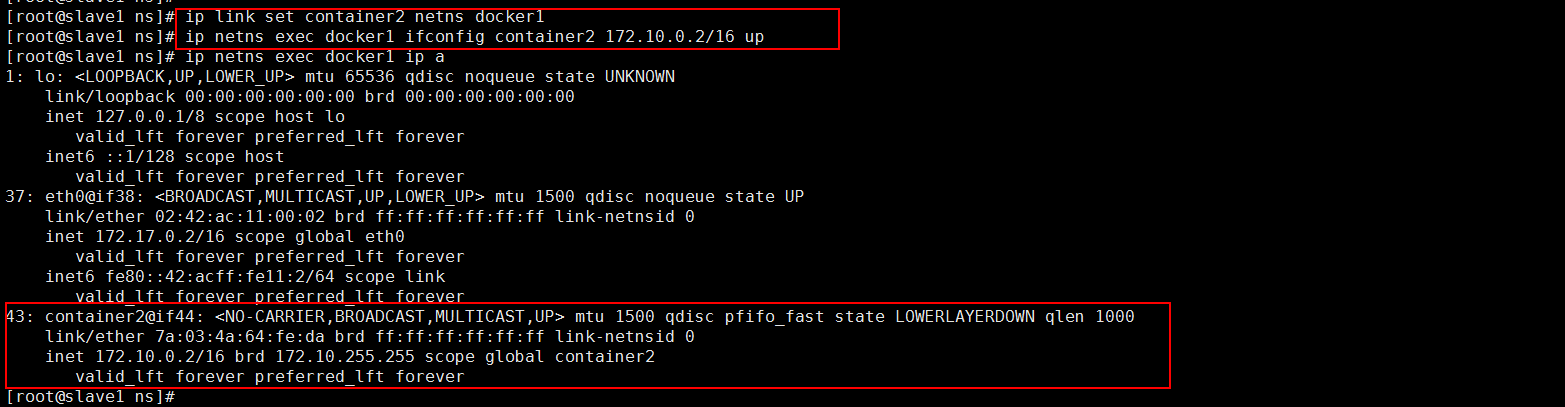
docker为了保护命名空间不被用户随意修改将对应的网络命名空间保存到/proc/container\_Pid/ns/net

为了能够管理docker容器的网络命名空间，将对应文件通过软连接，连接到对应目录下



3、创建veth对，将其中一个veth放入容器命名空间并分配ip地址。

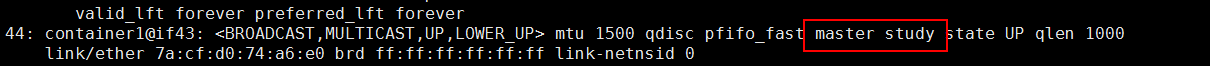
clipboard.png



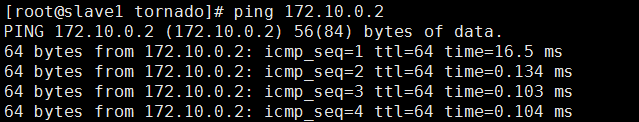
4、将另一个veth接口挂载到宿主机上创建的网卡，并启动网卡和veth接口。

clipboard.png

clipboard.png



5、从主机与容器内的veth接口 container2发起通信



当容器向外部世界发送网络请求时，数据流量会经过veth接口流入主机的网卡，将数据发送到宿主机上。

**二、容器与外网的通信**

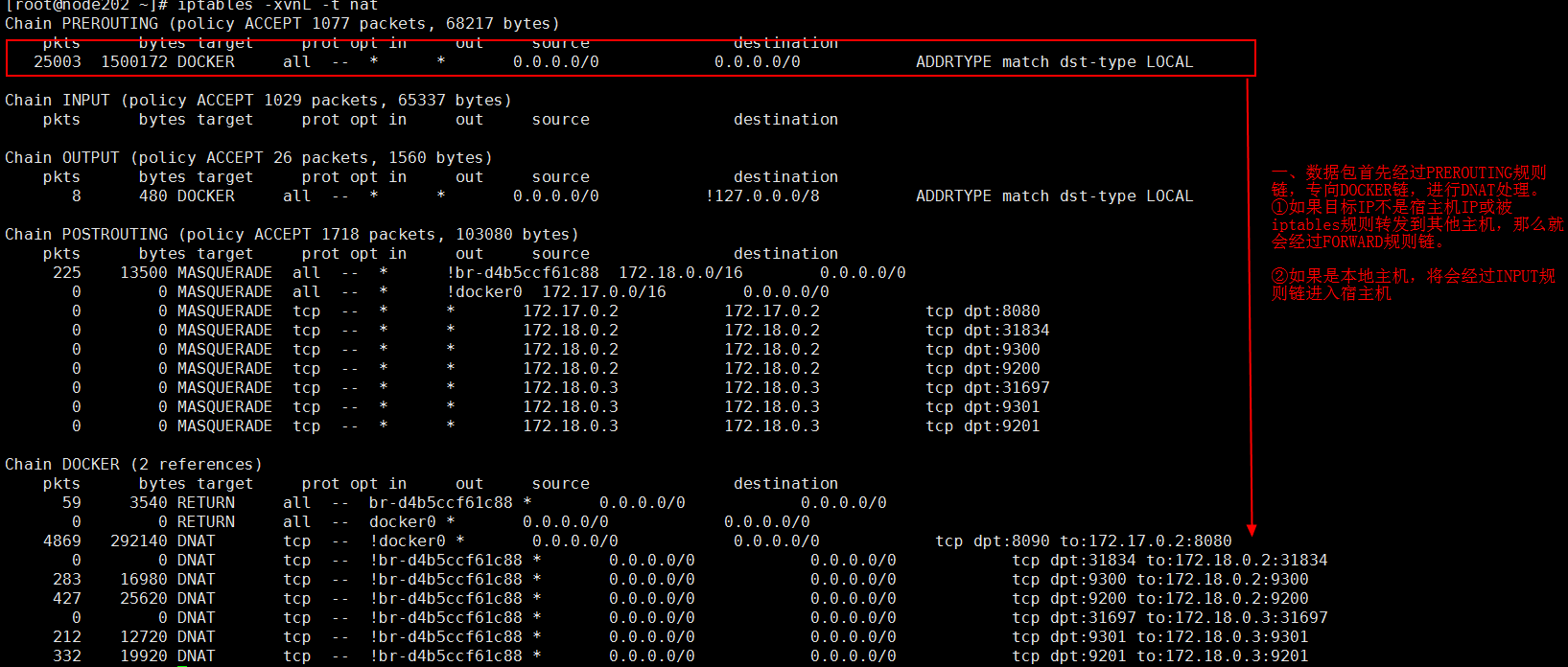
在第一节中，介绍了宿主机与docker容器间的交互网络。随着宿主机与容器间通信的建立，docker便可以藉由宿主机作为中转站，将容器对外部世界的请求发送到对应的外部主机。这一工作又iptables规则进行处理。这里简答的介绍相应的iptables规则。详细查看iptables部分内容。

这是两个部署了elasticsearch服务的节点，两个容器开发的端口映射分别为9200、9300和9201、9301。

首先来看看从容器对外发送请求会触发那些规则

1、

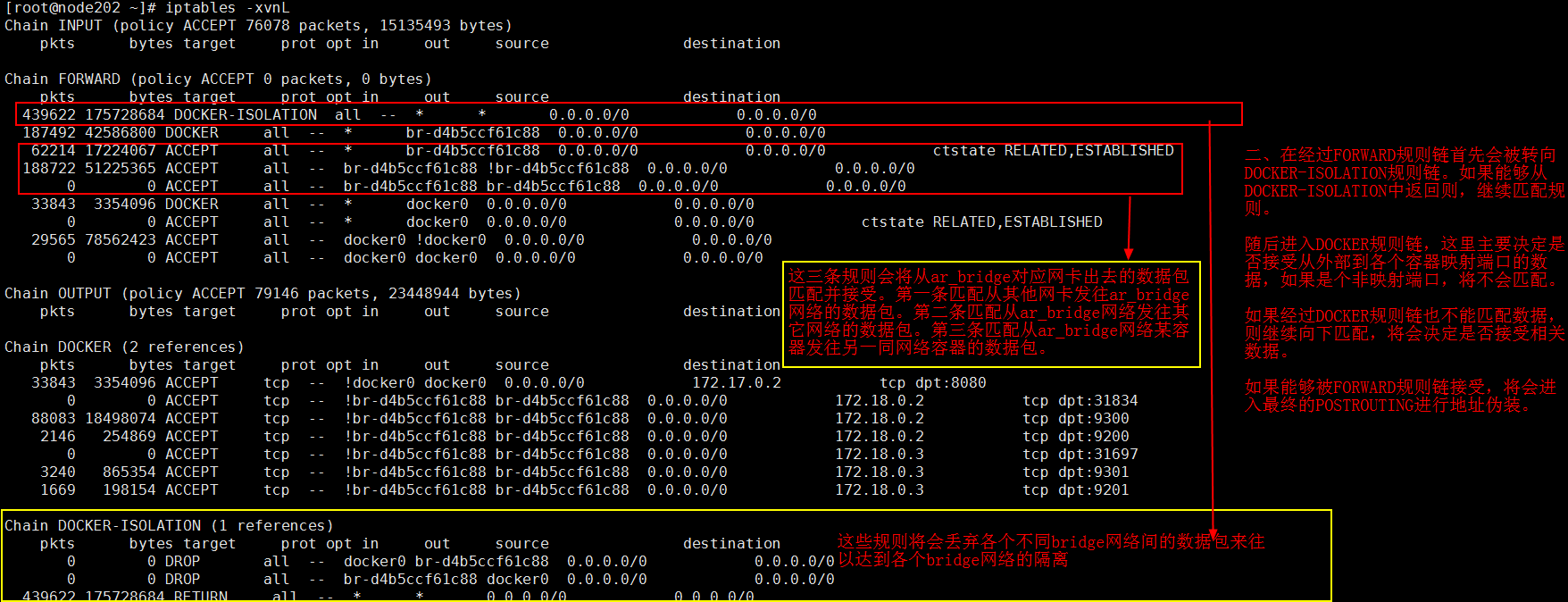
数据包从容器的veth接口留到宿主的某个网卡上，并进行iptables规则匹配，首先会经过PREROUTING的规则，如果最终发往本机，将会继续匹配INPUT规则链，否则将会匹配FORWARD规则链。



2、

如果经过PREROUTING规则链后，目标ip为宿主机，由于上图没有对INPUT规则进行编写，所以会直接进行用户空间。

如果经过PREROUTING规则链后,目标ip为其他节点，那么就会进入FORWARD规则链进行匹配，这里主要是对网络进行隔离并接收、抛弃相关数据包的。

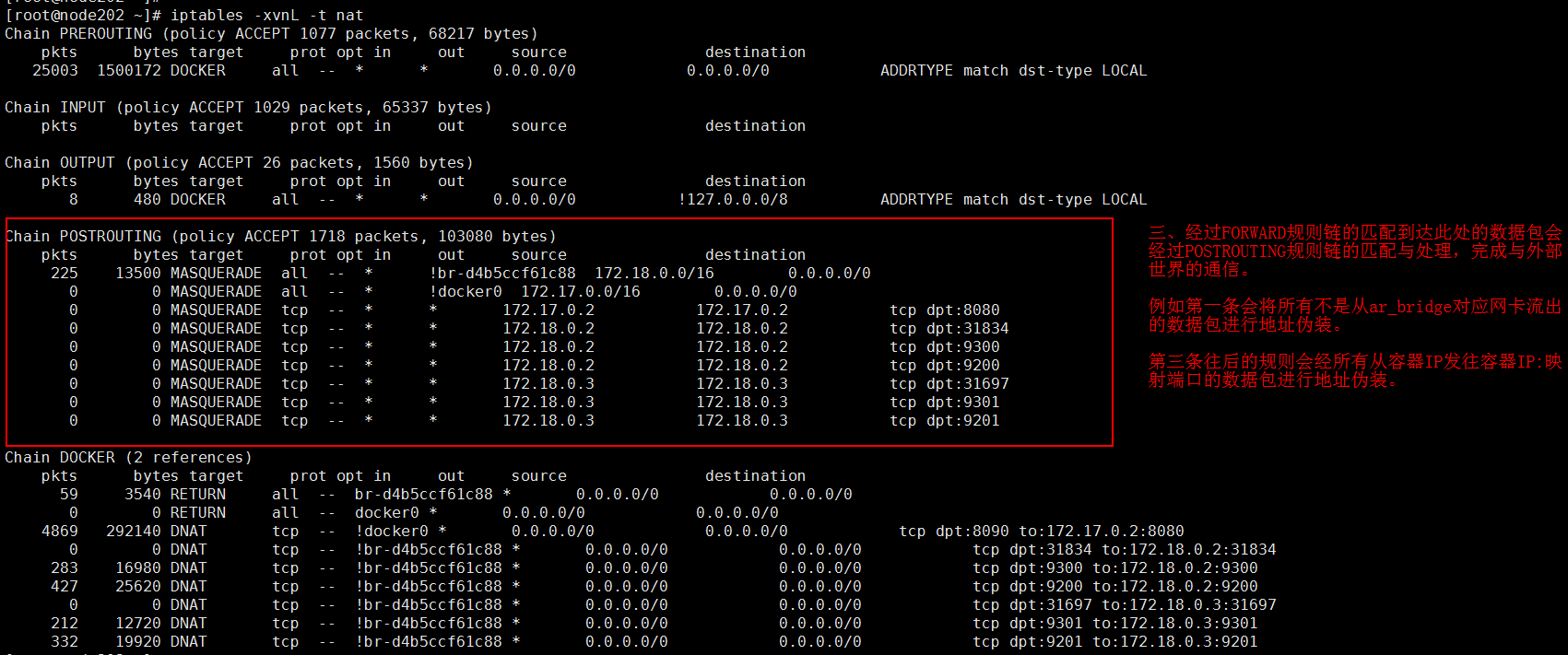


3、

被FORWARD规则链接受的数据包将会继续匹配POSTROUTING的规则链。这里主要是因为docker创建的bridge网络ip地址都是内部的ip，无法直接用于于外部通信，所以需要对向外发送的数据包进行地址伪装，从而能够与外部网络进行通信。

地址伪装，将源IP地址伪装为可以对外使用的ip地址。

图中红框第三条以后的规则使用场景尚不清楚。

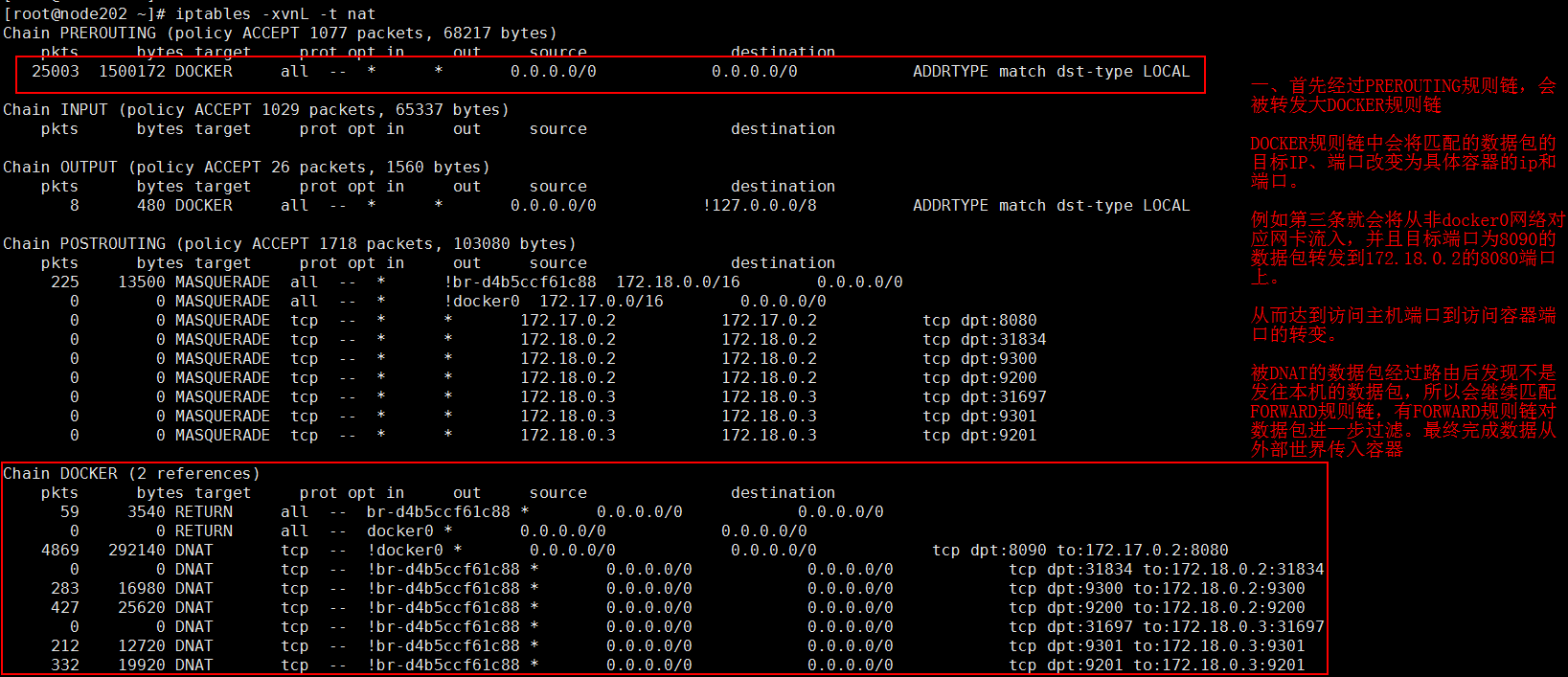


4、容器与外部世界的通信首先会通过veth接口将数据包发送到宿主机，宿主机会根据iptables规则将对应的数据包进行地址伪装将数据包使用一个对外的ip发送数据包。

看看外部主机向容器通信的方式。目前外部主机访问容器服务的方式是将容器的端口映射到主机，外部世界请求宿主机的对应端口。

1、

外部请求来到宿主机后，首先会经过PREROUTING的规则链，从而被转发到了DOCKER规则链，在DOCKER规则链定义了一系列的DNAT规则，从而将发送到对应端口的数据包转发大对应的docker bridge网络的具体地址



2、

图中红框第二条目前猜测是用于在同一网络间各个容器的交互使用的

