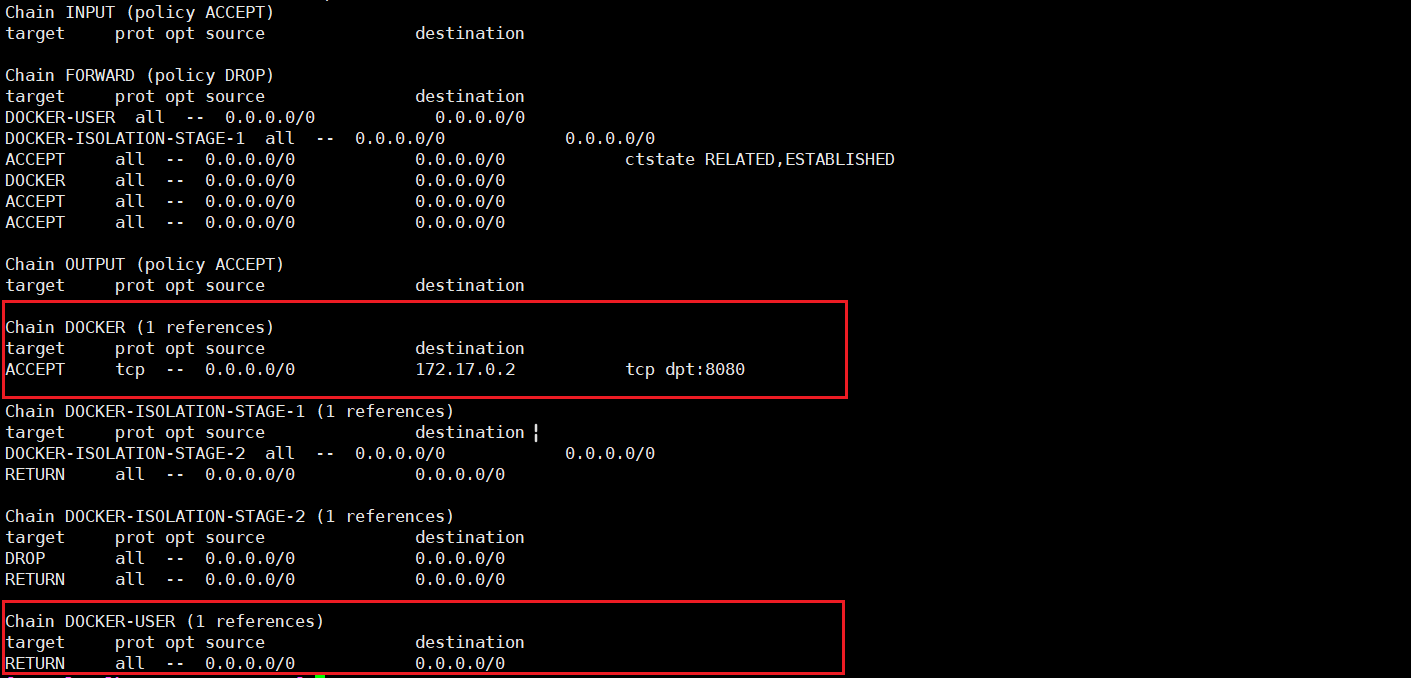
# **简介**

Docker至少会在iptables规则中自动安装两个新链，一个是DOCKER，一个是DOCKER-USER，可以通过如下方式查看

iptables -S -t nat 查看DOCKER链规则

iptables -S 查看DOCKER-USER链规则

随着docker版本不同，可能还会有其他链，但一般来说，我们**应该只修改DOCKER-USER链**即可。



在DOCKER-USER 链中添加的规则会先于DOCKER链生效，因此在这个链中手动添加规则将会生效。这是最推荐的方式，没有之一。

1、不要尝试去手动修改DOCKER链，这样可能会使docker网络出现问题。

2、注意添加的规则要插在DOCKER-USER链的顶部，默认情况下所有到达docker主机的连接都会被允许。

3、限制条件就是，如果端口不是在Docker中开放的，比如22端口，那么在DOCKER链中插入规则不会生效，需要在INPUT链中设置规则。

# **通过**DOCKER-USER**链规则限制**

* 1. **添加规则**

**第一种**

|  |
| --- |
| #iptables -I DOCKER-USER -i ens33 -p tcp --dport 8080 -j DROP  #iptables -I DOCKER-USER -i ens33 -s 192.168.31.42 -p tcp --dport 8080 -j ACCEPT |

第二种

|  |
| --- |
| #iptables -I DOCKER-USER -p tcp -m conntrack --ctorigdstport 8080 --ctdir ORIGINAL -j DROP  #iptables -I DOCKER-USER -s 192.168.31.42/32 -p tcp -m conntrack --ctorigdstport 8080 --ctdir ORIGINAL -j ACCEPT |

一定要先执行禁止命令，再执行放行命令，否则放行命令不会生效，**目前使用第二种**

|  |
| --- |
| **连接状态匹配模块:**  -m conntrack [[!] [--ctstate | --ctstatus 状态列表]] [[!] --ctproto tcp|udp|sctp|..四层协议.. ] [...其它选项...]  --ctdir {ORIGINAL|REPLY} 　　　　 #指定仅匹配(ORIGINAL)源或(REPLY)目的地址,默认都匹配.  --ctorigsrcport |--ctorigdstport |--ctreplsrcport |--ctrepldstport port[:port]]     #匹配起始方或响应方的源或目的端口 |

**1.2 查看docker-user规则**

iptables --line -nvL  DOCKER-USER

**1.3 保存规则**

以上是临时性的，服务器重启后配置会失效。

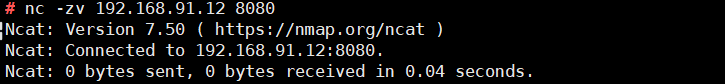
iptables-save > /etc/sysconfig/iptables

然后在**系统重启后**手动加载此配置文件

iptables-restore < /etc/sysconfig/iptables

**1.4 测试**

# nc -zv 192.168.91.12 8080



**1.5 参考文档**

（1）<https://blog.csdn.net/yeqinghanwu/article/details/125979997>

加了-i ens33 没添加 -m conntrack --ctorigdstport

1. <https://blog.huati365.com/dd810c2f4d4b6b36>

在-m conntrack --ctorigdstport 基础上加了-i ens33

1. <https://www.kejiwanjia.com/jiaocheng/75308.html>

在-m conntrack --ctorigdstport 没加-i ens33 加了 --ctdir ORIGINAL

-m conntrack --ctorigdstport 主要是这个与第（1）个文档不同

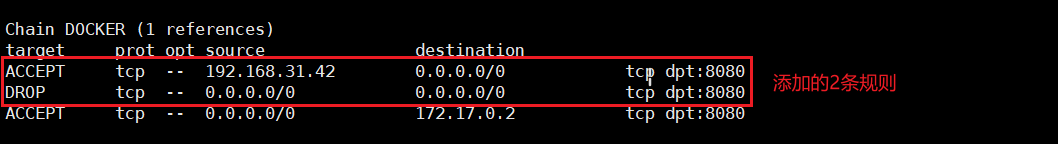
# **3、通过**DOCKER**链规则限制**

服务是用docker起的，docker自己会往系统中注册一个虚拟网卡叫docker0，访问docker服务的流量会直接被转发到这张网卡上，而因为docker0在linux系统中会被视为一张网卡，所以他的iptables规则是独有的

**2.1添加规则**

只允许特定IP访问特定端口

|  |
| --- |
| iptables -I DOCKER -p tcp --dport 8080 -j DROP  iptables -I DOCKER -s 192.168.31.42 -p tcp --dport 8080 -j ACCEPT  iptables -L -n |



iptables -A ，-A的意思是加在最后，而iptables的匹配顺序是从上到下，所以 ****用-A往最后追加的话，那就是【先写先匹配】**** 而第二个代码块中的写法是 iptables -I ，-I的意思是加在最前，也就是上面截图中的效果

**2.2保存规则**

以上是临时性的，服务器重启后配置会失效。

iptables-save > /etc/sysconfig/iptables

然后在**系统重启后**手动加载此配置文件

iptables-restore < /etc/sysconfig/iptables

分析原因应该是： 服务器重启后，iptables服务先启动，启动后会去加载备份的配置文件，但是 ****此时，docker服务还未启动，DOCKER规则链还未创建**** ，所以虽然iptables有加载我们备份的配置，但他没法加载DPOCKER规则链上的内容，从而导致我们写的DOCKER规则链上的规则没有成功加载

而docker启动后，DOCKER规则链就存在了，这时不管是手动让iptables加载配置文件还是直接重启iptables让他自己再读一遍配置，我们备份的DOCKER规则就都能成功加载了

**2.3 参考文档**

<https://www.freeaihub.com/post/107770.html>

# **4 删除规则**

**4.1查看出当前的防火墙规则**

iptables -L -n

**4.2查看到每个规则chain 的序列号**

iptables -L -n --line-number

还可以具体查看某个链规则

\*查询DOCKER表并显示规则编号

iptables -L DOCKER -n --line-number

\*查询DOCKER-USER表并显示规则编号

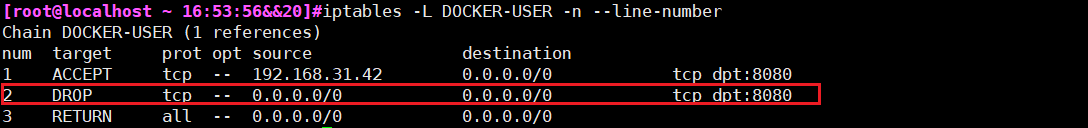
iptables -L DOCKER-USER -n --line-number

**4.3实例**

例如我们DOCKER-USER 这里边的某个规则，就使用使用命令

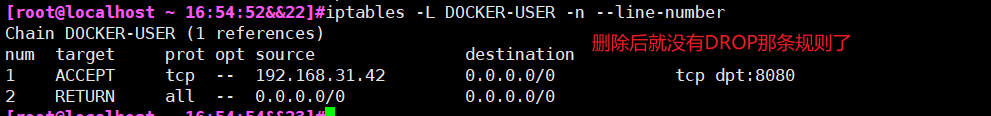
4.4.1 例如删除DOCKER-USER的第2条规则用如下命令。

iptables -D DOCKER-USER 2



4.4.2再使用，就可以看出删除成功了

iptables -L DOCKER-USER -n --line-number



4.4.3 保存规则

**4.4 参考文档**

<https://blog.csdn.net/weixin_57095087/article/details/123126217>

# **5 iptables常用命令**

| **命令** | **效果** |
| --- | --- |
| iptables -nvL | 列出所有iptables规则 |
| iptables -L -n | 列出所有iptables规则 |
| iptables -L -n --line-number | 列出所有iptables规则，并显示编号【有编号才好删】 |
| iptables-save > 文件绝对路径 | 将当前iptables规则保存到指定文件中 |
| iptables-restore < 文件绝对路径 | 从指定文件中加载iptables规则 |

# **6 重启自动添加iptables规则**

**方式一**

|  |
| --- |
| echo "/usr/sbin/iptables-restore < /etc/sysconfig/iptables" >> /etc/rc.d/rc.local |

**方式二**

|  |
| --- |
| #!/bin/bash  num=$(iptables -L DOCKER-USER -n --line-number|egrep -v 'Chain|num'|wc -l)  if [ ${num} = 1 ];then  /usr/sbin/iptables-restore < /etc/sysconfig/iptables  else  sed -i "**s/^[^#].\*add\_docker\_user\_chain.\*/#&/g**" /etc/crontab  fi |

#：s:替换

#: ^:开头匹配

#： [^#]:匹配非#

#： #&:中的&代表匹配整行，整个意思就是行前面加上#号

#： g:全部（只匹配特定行不加）

# .\* 是匹配任意字符（此处匹配包含**add\_docker\_user\_chain关键字的行**） 也可以用$表示以某个字符结尾

**主要是方式一 方式二还有待改善（被注释后，下次重启就无法跑定时任务）**

# **7 iptables使用总结**

<https://www.shuzhiduo.com/A/GBJr3WmR50/>

<https://www.kancloud.cn/jiftle/iptables-detailed-introduction/1982086> iptables详解

<https://juejin.cn/post/6969751429556731941> Centos7 Docker iptables规则链说明

# **8 docker容器在不同场景下被访问时数据包在防火墙中的过程**

下面列一下docker容器在不同场景下被访问时数据包在防火墙中的过程：

1.容器出外网： PREROUTING链路----->FORWORD链路------->OUTPUT链路---------->POSTROUTING链路

2.宿主机访问127.0.0.1： OUTPUT链路--------->POSTROUTING链路 ------->PREROUTING链路--------->INPUT链路----->容器

4.访问宿主机的默认网卡地址：OUTPUT链路--------->DOCKER链路------->POSTROUTING链路----->容器

4.宿主机之外的主机访问容器： PREROUTING链路---------->DOCKER链路----->FORWORD链路----------->DOCKER链路-------->POSTROUTING链路------->容器

5.宿主机访问容器： OUTOUT链路------>POSTROUTING链路------->容器

6.容器访问宿主机端口：PREROUNTING链路---->DOCKER链路------->INPUT链路--->容器

# **9 限制容器访问外部网络，外部通过端口映射可正常访问容器**

# 禁止容器访问宿主机，需要获取宿主机IP地址172.16.2.2

sudo iptables -I INPUT -i docker0 -d 172.16.2.2 -j DROP

# 禁止容器访问外部192.168.0.0/16段

sudo iptables -I FORWARD -i docker0 -d 192.168.0.0/16 -m state --state NEW -j DROP

# 禁止容器访问外部所有网络

sudo iptables -I FORWARD -i docker0 -d 0.0.0.0/0 -m state --state NEW -j DROP

# **10 其他**

<https://blog.csdn.net/Liv2005/article/details/112850208>

