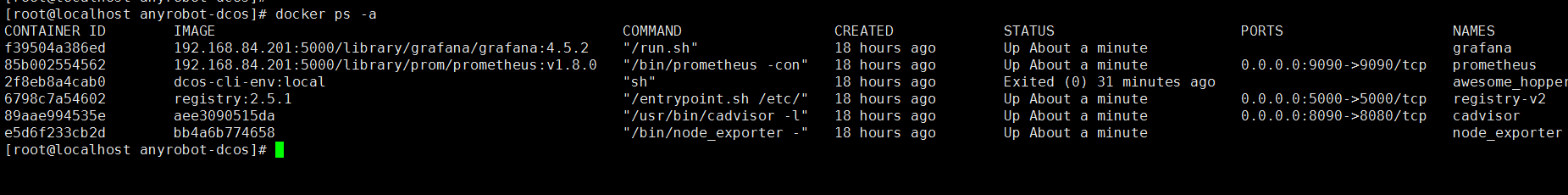
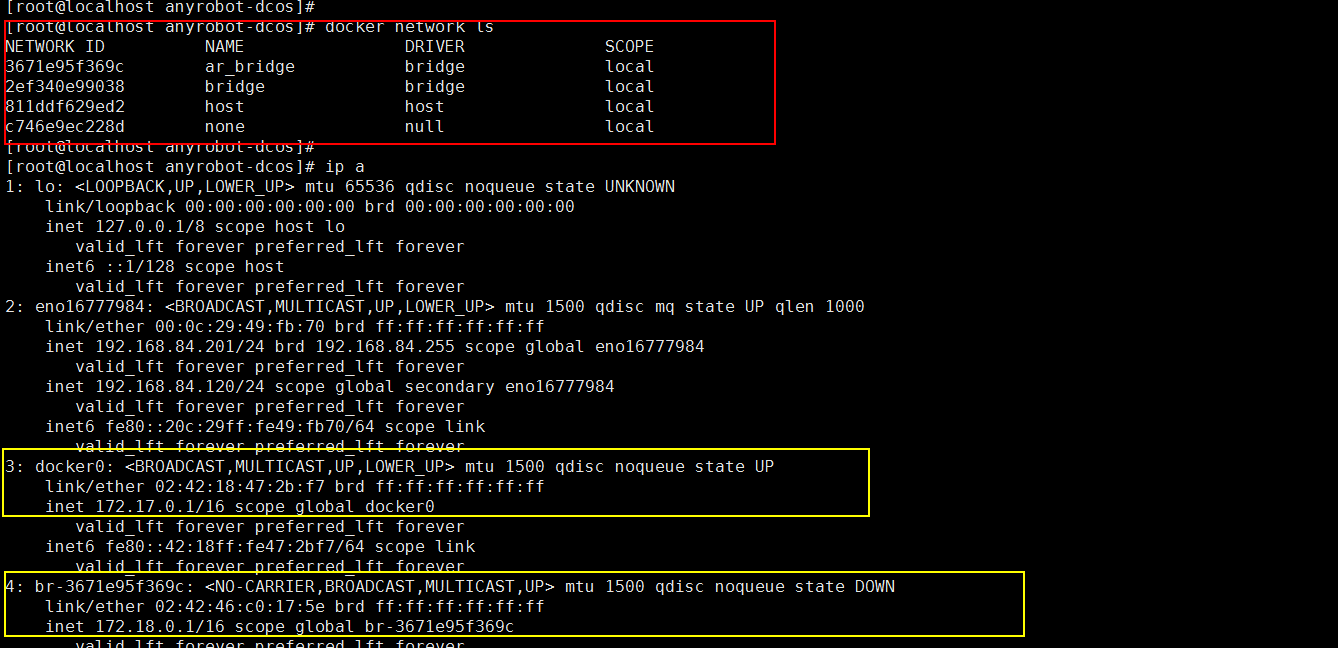
实验环境如下

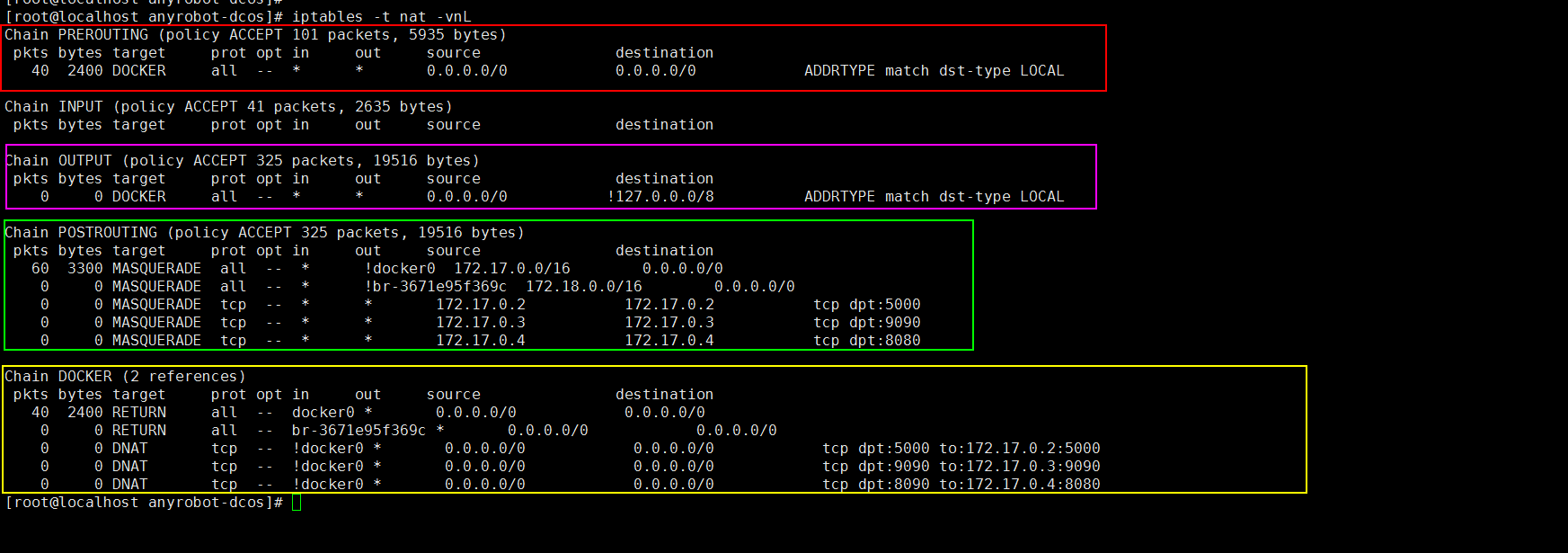


存在三个容器使用端口映射将容器内部端口映射到宿主机。并且使用的是默认的bridge网络。宿主机上的网络环境为



docker主要维护的iptables表有filter和nat表

nat表比较简单，先整体看一下nat表内的规则



①图中红框部分的规则为docker定义的RPEROUTING规则，该规则的意义为：满足“从任意网卡来到任意网卡去，不论源IP和目标IP为何”条件的数据包将跳转到DOCKER规则链进行匹配。

②图中紫色部分规则为docker定义的OUTPUT规则，该规则意义为：满足“从任意网卡来到任意网卡去，除了127.0.0.8网段”条件的数据包将跳转到DOCKER规则链进行匹配。

③图中绿色部分为docker维护的POSTING规则链，图中对匹配规则采取的策略为“MASQUERADE”。策略的作用为IP地址伪装，让内部网络到外部网络的数据可以自动的将源IP地址改变为一个可以与外部通信的IP地址。其作用与SNAT类似，不过使用SNAT时使用的外部IP是静态的，而MASQUERADEE可以做到动态分配。

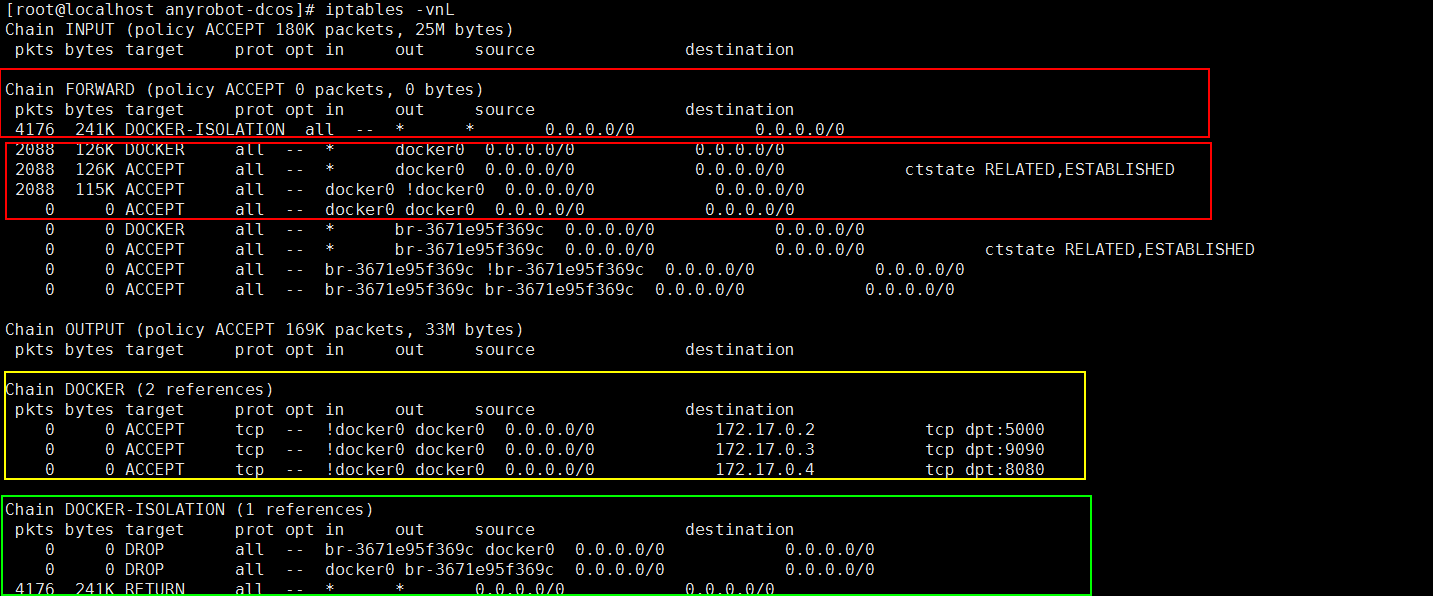
所以绿框部分前两条规则其实就是定义了“所有不是将要从docker0（bridge）和br-3671e95f369c（ar\_bridge）网卡流出的网络数据包”的IP地址伪装。

绿框剩下的三条规则则是定义了所有从“任意网卡出来，源IP地址为a，目标地址为a，端口为c的网络数据包”的IP地址伪装。

④图中黄色部分为DOCKER创建的自义定规则链，可以发现DOCKER规则链被应用了两次（在PREROUTING和OUTPUT链）。前两条规则是的定义可以概括为“将要经过docker0、br-3671e95f369c网卡的流入的网络数据包”将会采用“RETURN”策略。而“RETURN”策略则会将该网络数据包重新返回到跳转到DOCKER规则链的规则处继续匹配其他规则。

黄框部分剩下的三条规则是定义了“所有不从docker0网卡流入，目标端口为5050、9090、8090的网络数据包”采用DNAT策略将数据包转发到对应IP的对应端口。这里其实就是外部网络在向宿主机的5050、9090、8090端口发送请求时会被这些规则匹配，从而将对应的请求转发到对应容器内对应端口处。这就是外部世界访问容器时会发生的转发规则匹配。

filte表规则如下



主要由三个部分构成，主要用于对docker网络和容器的过滤

①图中第一个红框部分首选会将所有经过FORWARD规则链的数据包跳转到DOCKER-ISOLATION规则链进行匹配。即图中绿框部分。绿框中前两个规则中将两个网卡之间来往的数据包采用DROP策略，以此完成两个网卡之间的隔离。而最后一条将前两条规则未匹配的数据包返回原处继续匹配。

②若数据包不是在两个网卡间来往，则会进入第二个红框部分。如果数据是从docker0网卡流出，那么将跳转到DOCKER链进行匹配，即图中黄框部分。黄框部分的DOCKER规则链的三条规则可以解读为“若非从docker0网桥流入，将从docker0流出并且数据包目标IP为容器IP，端口为容器映射端口的数据包”将采用“ACCEPT”策略

③最后红框剩下的三条规则意义分别为，“从任意网卡流入，从docker0网卡流出的状态数据”采用“ACCEPT”；“从docker0网卡流入，从非docker0网卡流出的数据”采用“ACCEPT”；“从docker0网桥流入，从docker0网桥流出的数据”采用“ACCEPT”