Docker 容器要出网/容器互访 FORWARD (转发) UFW

详细配置

先说结论：这些规则本身没语法错误，但它们几乎都作用在 **INPUT** (进宿主机) 链上，而 Docker 容器要出网/容器互访走的是 **FORWARD** (转发) 链。UFW 默认 `FORWARD` 策略是 **DROP**，所以容器流量被丢了——这正是"容器内网络不通"的根因。

为什么容器不通

- **`Anywhere ALLOW 172.18.0.0/16`** 是 **INPUT** 规则，不是 **FORWARD** 规则这条等价于 `ufw allow from 172.18.0.0/16`，只允许"容器网段 → 宿主机"的入站包。它不会放行"容器 → 外网""容器 → 容器"的转发流量，所以你以为开了 docker 网段，其实没开到点上。
- **UFW** 默认转发策略为 **DROP**（`/etc/default/ufw` 里 `DEFAULT_FORWARD_POLICY="DROP"`）。容器发包要经宿主机 FORWARD 链转发出去，没有 ACCEPT 规则命中就被丢。
- 网段可能写错/写窄：你只放了 `172.18.0.0/16`。Docker 默认网桥是 `172.17.0.0/16`，compose 网络常是 `172.18/19/20…` 不一。容器若落在 172.17 或别的子网，这条也匹配不上。Docker 私有段整体是 `172.16.0.0/12`。
- **IPv6** 只有 **DENY** 没有放行：底部 `8067,20828 (v6) DENY`、`22 (v6) DENY`，但没有任何 v6 的 docker 放行；若容器走 IPv6 同样不通。

其它规则层面的问题

- **`Anywhere ALLOW 172.18.0.0/16`** 过宽：对该网段放开了所有端口访问宿主机，安全面偏大，建议按需收敛到具体端口。
- 底部 **`22/tcp DENY Anywhere`**、**`8067,20828/tcp DENY Anywhere`** 是冗余：UFW 默认 incoming 就是 deny，这两条不改变行为（无害，但可清理）。
- **.51-.57** 那一大段重复端口组：只是啰嗦，功能没问题，可用地址段或 `ufw allow from 192.168.6.48/29 to any port 8067,20828` 之类合并。

修复（三选一，推荐 A）

A. 只给 **Docker** 转发放行（最小改动、安全）

编辑 `/etc/ufw/after.rules`，在 `COMMIT` 之前加入：

```
-A ufw-forward -s 172.16.0.0/12 -d 172.16.0.0/12 -j ACCEPT
-A ufw-forward -s 172.16.0.0/12 -m conntrack --ctstate RELATED,ESTABLISHED -j ACCEPT
-A ufw-forward -d 172.16.0.0/12 -m conntrack --ctstate RELATED,ESTABLISHED -j ACCEPT
```

然后：

```
sudo ufw reload
sudo systemctl restart docker
```

（更规范的做法是用 Docker 的 `DOCKER-USER` 链，但 UFW 激活时经 `after.rules` 加 `ufw-forward` 更稳，不会被 docker 重启冲掉。）

B. 直接放开转发策略（最省事，安全性略降）

`/etc/default/ufw` 改 `DEFAULT_FORWARD_POLICY="ACCEPT"`，再：

```
sudo ufw disable && sudo ufw enable
sudo systemctl restart docker
```

C. 用 **ufw** 的 **route**（**FORWARD**）规则

```
sudo ufw route allow in on docker0 out on <你的外网网卡>
sudo ufw route allow in on <外网网卡> out on docker0
```

（`<外网网卡>` 用 `ip route` 里 default 那条的 dev 替换。）

改完先自查

```
grep FORWARD /etc/default/ufw
sudo iptables -L FORWARD -n -v --line-numbers | head
ip -br addr | grep -E 'docker|br-'       # 确认真实容器网段
docker network inspect <net> -f '{{range .IPAM.Config}}{{.Subnet}}{{end}}'
```

确认 FORWARD 里 docker 流量能命中 ACCEPT，再进容器 `ping -c3 1.1.1.1` / `wget -qO- https://www.baidu.com` 验证。

不正确，有两处问题，其中第一处是致命的——照现在这样 `ufw reload`/`ufw enable` 会直接报错、整段规则都加载不了。

问题 1（致命）：链名写错了

你加的是 `-A ufw-forward ...`，但 **UFW** 根本没有 `ufw-forward` 这条链。看你自己文件顶部声明的三条链就知道：

```
:ufw-after-input - [0:0]
:ufw-after-output - [0:0]
:ufw-after-forward - [0:0]
```

FORWARD 相关的链只有 `ufw-before-forward` 和 `ufw-after-forward`（还有 `ufw-user-forward`）。往不存在的 `ufw-forward` 追加，`iptables-restore` 是原子执行的，一旦这条报错，整个 after.rules 全部回滚不加载，ufw 还会进入报错状态。

> 这是我上一条回复给你的示例里写错了链名，抱歉，应改成 `ufw-after-forward`。

问题 2（逻辑）：只开了 "容器 ↔ 容器"，没开 "容器 → 外网"

就算链名改对，你第一条是：

```
-s 172.16.0.0/12 -d 172.16.0.0/12 -j ACCEPT
```

源和目的都限定在 172.16/12，只放行容器互访。容器访问外网时目的 IP 是公网地址，不匹配这条；而外网回程的 NEW 包也进不来。所以 "上不了外网/DNS" 这个最常见的症状仍然没解决。

正确写法

把三行替换成两行（链名对 + 放开 egress）：

```
-A ufw-after-forward -s 172.16.0.0/12 -j ACCEPT
-A ufw-after-forward -d 172.16.0.0/12 -m conntrack --ctstate RELATED,ESTABLISHED -j ACCEPT
```

- 第 1条：允许容器发起的所有出站（含上外网、含容器互访）。
- 第 2条：允许回到容器的已建立/关联回程包。
- 位置放在 `COMMIT` 之前即可（你现在放的位置没问题）。

应用与验证

```
sudo ufw reload          # 或 sudo ufw disable && sudo ufw enable
sudo systemctl restart docker
sudo iptables -S ufw-after-forward     # 确认两条规则真的进去了、没报错
```

进容器测：

```
ping -c3 223.5.5.5       # 通外网 IP
wget -qO- https://www.baidu.com | head # 通 DNS+HTTP
```

补充提醒

- `172.16.0.0/12` 覆盖 Docker 全部私有段（172.17 网桥、172.18/19… compose 网络），比你原来只写 `172.18.0.0/16` 稳妥。
- 若你只想"容器能上外网、但禁止容器互访/禁止访问宿主机内网"，那是另一套更严格的规则，需要单独配；当前这套是"先保证连通"。
- 改完 `iptables -S ufw-after-forward` 若为空或 reload 报错，多半还是链名/缩进问题，可把报错原文发我。

需要我把它收敛成"只放行到特定外网/指定端口"的最小权限版本，告诉我你的容器实际要访问哪些目标即可。

Revision #2
Created 2026-09-04 02:46:06 UTC by Admin
Updated 2026-09-04 03:28:19 UTC by Admin