



# REDES E FIREWALL NO DOCKER

Guia Oficial iT Guys

**Código:** ITGINF 0009/26 | **Classificação:** RESTRITO

**Responsável:** João Pedro Toledo Gonçalves | **Data:** 25/01/2026

## 1. HISTÓRICO DE REVISÃO

### NOTA

REGRA DE OURO:

### NOTA

1. **Autor:** João Pedro Toledo Gonçalves.

### NOTA

2. **Descrição:** Criação do documento.

| Data       | Versão | Descrição       | Autor                       |
|------------|--------|-----------------|-----------------------------|
| 25/01/2026 | 1.0    | Criação Inicial | João Pedro Toledo Gonçalves |

## 2. OBJETIVO

Explicar a arquitetura de redes do Docker (Bridge, Host, Overlay), como segregar containers e como gerenciar regras de Firewall com segurança.

## 3. PRÉ-REQUISITOS

- [ ] Docker Engine instalado e rodando.
- [ ] Acesso root para visualizar regras de iptables.

## 4. PASSO A PASSO (EXECUÇÃO)

### Etapa 1: Drivers de Rede Padrão

1. Liste as redes existentes no host.

- \* **bridge:** Padrão. Containeres recebem IP interno (172.17.x.x).
- \* **host:** Remove isolamento. Container usa a eth0 do host.
- \* **none:** Sem rede.

```
docker network ls
```

## Etapa 2: Criando Redes Segregadas (User-defined Bridge)

### NOTA

**MELHOR PRÁTICA:** Nunca use a rede `bridge` padrão para produção (falta DNS resolution entre nomes). Crie a sua.

1. Crie uma rede dedicada para sua stack (ex: `app-network`).
2. Suba containeres conectados a ela.

```
docker network create --driver bridge app-network
docker run -d --name meu-db --network app-network postgres
docker run -d --name meu-app --network app-network nginx
```

### NOTA

### NOTA

Agora o container `meu-app` consegue pingar `meu-db` pelo NOME. A rede bridge padrão não faz isso.

## Etapa 3: Exposição de Portas e Segurança (Firewall)

### IMPORTANTE

**IMPORTANTE:** O Docker insere regras no `iptables` ANTES do UFW. Liberar `-p 8080:80` ignora o bloqueio do UFW.

1. **Modo Inseguro (Padrão):** Libera para internet (0.0.0.0).

```
docker run -p 8080:80 nginx
```

2. **Modo Seguro (Localhost Apenas):** Use se for usar proxy reverso (Nginx/Traefik) no host.

```
docker run -p 127.0.0.1:8080:80 nginx
```

## Etapa 4: Auditoria de Regras

1. Verifique como o Docker mapeia as regras no NAT.

```
sudo iptables -t nat -L DOCKER
```

## 5. SOLUÇÃO DE PROBLEMAS (TROUBLESHOOTING)

### Problema 1: Container não resolve DNS externo

- \* **Causa:** Conflito de DNS no `/etc/docker/daemon.json` ou firewall bloqueando porta 53 UDP.
- \* **Solução:**
  1. Teste com `docker run busybox nslookup google.com`.
  2. Check se o host tem acesso.

### Problema 2: UFW não bloqueia portas do Docker

- \* **Causa:** Design do Docker ("Docker bypasses UFW").
- \* **Solução:**
  1. Edite `/etc/default/docker` e adicione `iptables=false` (Cuidado: quebra o NAT!).
  2. **Recomendado:** Use o bloqueio na chain `DOCKER-USER` do iptables se precisar restringir IPs externos.

## 6. DADOS TÉCNICOS

| Campo         | Valor         | Descrição                  |
|---------------|---------------|----------------------------|
| Subnet Padrão | 172.17.0.0/16 | Rede Bridge padrão         |
| DNS Interno   | 127.0.0.11    | Resolver interno do Docker |

## 7. VALIDAÇÃO FINAL

- [ ] `docker network ls` mostra sua rede customizada?
- [ ] Containeres na mesma rede se pingam por nome?
- [ ] Portas expostas estão acessíveis (`netstat -tuln`)?