Docker with iptables

- Docker and iptables chains
- Add iptables policies before Docker's rules
- Match the original IP and ports for requests
- Allow forwarding between host interfaces
- Restrict external connections to containers

Docker creates iptables rules in the host's network namespace for bridge networks. For bridge and other network types, iptables rules for DNS are also created in the container's network namespace.

Creation of iptables rules can be disabled using daemon options `iptables` and `ip6tables`, see Prevent Docker from manipulating firewall rules. However, this is not recommended for most users as it will likely break container networking.

Docker and iptables chains

To support bridge and overlay networks, Docker creates the following custom `iptables` chains in the `filter` table:

- `DOCKER-USER`
  - A placeholder for user-defined rules that will be processed before rules in the `DOCKER-FORWARD` and `DOCKER` chains.
- `DOCKER-FORWARD`
  - The first stage of processing for Docker's networks. Rules that pass packets that are not related to established connections to the other Docker chains, as well as rules to accept packets that are part of established connections.
- `DOCKER`, `DOCKER-BRIDGE`, `DOCKER-INTERNAL`
  - Rules that determine whether a packet that is not part of an established connection should be accepted, based on the port forwarding configuration of running containers.
- `DOCKER-CT`
  - Per-bridge connection tracking rules.
- `DOCKER-INGRESS`
  - Rules related to Swarm networking.

In the `FORWARD` chain, Docker adds rules that unconditionally jump to the `DOCKER-USER`, `DOCKER-FORWARD` and `DOCKER-INGRESS` chains.

In the `nat` table, Docker creates chain `DOCKER` and adds rules to implement masquerading and port-mapping.

Docker requires IP Forwarding to be enabled on the host for its default bridge network configuration. If it enables IP Forwarding, it also sets the default policy of the iptables `FORWARD` chain in the `filter` table to `DROP`.

Add iptables policies before Docker's rules

Packets that get accepted or rejected by rules in these custom chains will not be seen by user-defined rules appended to the `FORWARD` chain. So, to add additional rules to filter these packets, use the `DOCKER-USER` chain.

Rules appended to the `FORWARD` chain will be processed after Docker's rules.

> By default, remote hosts can only reach a container through a port published to one of the Docker host's addresses. Sending packets to the container's own IP address instead ("direct routed" access) is not allowed.
>
> These packets are dropped by a rule in the `raw` table's `PREROUTING` chain, which is processed before the `filter` table. So, they never reach the `DOCKER-USER` chain, and a rule in `DOCKER-USER` cannot allow them.
>
> Any packet that reaches the host's firewall rules already addressed to a container is treated this way. For example, a Kubernetes CNI plugin may translate a Service address to a container address before the packet reaches the host's `PREROUTING` rules. Docker cannot distinguish the result from a packet routed to the container by a remote host. Which host interface these packets arrive on depends on the plugin's data path — for an overlay network it is normally the tunnel device.
>
> To allow direct routed access to a container's *published* ports, use network option `com.docker.network.bridge.trusted_host_interfaces`, or daemon option `allow-direct-routing`. Unpublished ports are still protected. See Direct routing to containers in bridge networks. To allow direct routed access to all container ports, published or not, see Gateway modes.

Match the original IP and ports for requests

When packets arrive to the `DOCKER-USER` chain, they have already passed through a Destination Network Address Translation (DNAT) filter. That means that the `iptables` flags you use can only match internal IP addresses and ports of containers.

If you want to match traffic based on the original IP and port in the network request, you must use the `conntrack` iptables extension. For example:

```
$ sudo iptables -I DOCKER-USER -p tcp -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT
$ sudo iptables -I DOCKER-USER -p tcp -m conntrack --ctorigdst 198.51.100.2 --ctorigdstport 80
-j ACCEPT
```

> Using the `conntrack` extension may result in degraded performance.

Allow forwarding between host interfaces

If Docker has set the default policy of the `FORWARD` chain in the `filter` table to `DROP`, a rule in `DOCKER-USER` can be used to allow forwarding between host interfaces. For example:

```
$ iptables -I DOCKER-USER -i src_if -o dst_if -j ACCEPT
```

Restrict external connections to containers

By default, all external source IPs are allowed to connect to ports that have been published to the Docker host's addresses.

To allow only a specific IP or network to access the containers, insert a negated rule at the top of the `DOCKER-USER` filter chain. For example, the following rule drops packets from all IP addresses except `192.0.2.2`:

```
$ iptables -I DOCKER-USER -i ext_if ! -s 192.0.2.2 -j DROP
```

You will need to change `ext_if` to correspond with your host's actual external interface. You could instead allow connections from a source subnet. The following rule only allows access from the subnet `192.0.2.0/24`:

```
$ iptables -I DOCKER-USER -i ext_if ! -s 192.0.2.0/24 -j DROP
```

Finally, you can specify a range of IP addresses to accept using `--src-range` (Remember to also add `-m iprange` when using `--src-range` or `--dst-range`):

```
$ iptables -I DOCKER-USER -m iprange -i ext_if ! --src-range 192.0.2.1-192.0.2.3 -j DROP
```

You can combine `-s` or `--src-range` with `-d` or `--dst-range` to control both the source and destination. For example, if the Docker host has addresses `2001:db8:1111::2` and `2001:db8:2222::2`, you can make rules specific to `2001:db8:1111::2` and leave `2001:db8:2222::2` open.

You may need to allow responses from servers outside the permitted external address ranges. For example, containers may send DNS or HTTP requests to hosts that are not allowed to access the container's services. The following rule accepts any incoming or outgoing packet belonging to a flow that has already been accepted by other rules. It must be placed before `DROP` rules that restrict access from external address ranges.

```
$ iptables -I DOCKER-USER -m state --state RELATED,ESTABLISHED -j ACCEPT
```

For more information about iptables configuration and advanced usage, refer to the Netfilter.org HOWTO.

□□□ #2
□ □□□ □□□ 2026-10-31 19:26:07 UTC
□ □□□ □□□ 2026-10-31 19:26:08 UTC