

Basic Configuration of dnsmasq in an Incus Container on Debian with Netplan

1 Introduction

This guide provides step-by-step instructions for setting up `dnsmasq` as a DNS and DHCP server in an Incus container running Debian. The network configuration is managed using Netplan to ensure proper network integration.

2 Prerequisites

Before proceeding, ensure the following:

- Incus is installed on the host system (`sudo apt install incus`).
- A Debian-based container is created in Incus.
- Basic knowledge of Linux networking and container management.
- Root or sudo access to the host and container.

3 Step-by-Step Configuration

3.1 Creating and Setting Up the Incus Container

Create a Debian container named `dnsmasq-container` using the following commands on the host:

```
1 incus create images:debian/12 dnsmasq-container
2 incus config set dnsmasq-container security.syscalls.intercept.
  mount true
3 incus config set dnsmasq-container security.nesting true
4 incus config set dnsmasq-container security.privileged true
5 incus start dnsmasq-container
```

The `security.syscalls.intercept.mount`, `security.nesting`, and `security.privileged` settings are required for `dnsmasq` and Docker to function correctly in the container.

3.2 Firewall Configuration

To allow traffic forwarding between the `incusbr0` bridge and the `wlo1` wireless interface, the following iptables rules are applied:

```
1 sudo iptables -A FORWARD -i incusbr0 -o wlo1 -j ACCEPT
2 sudo iptables -A FORWARD -i wlo1 -o incusbr0 -m state --state
   RELATED,ESTABLISHED -j ACCEPT
```

3.3 Installing Additional Packages

Install the necessary packages inside the container:

```
1 incus exec dnsmasq-container -- apt update
2 incus exec dnsmasq-container -- apt install -y \
3     netplan.io \
4     sudo vim nano git tmux mc zip unzip curl wget htop lynx \
5     iproute2 termshark bridge-utils \
6     python3 python3-ipython python3-pyroute2 python3-scapy \
7     docker.io docker-compose
```

3.4 Configuring Users and Permissions

Configure user access and permissions within the container.

3.4.1 Changing the Root Password

Set the root password to "passroot":

```
1 incus exec dnsmasq-container -- bash -c 'echo "root:passroot" | \
   chpasswd'
```

3.4.2 Adding a New User

Add a new user named "user" with the password "pass" and add them to the "sudo" and "docker" groups:

```
1 incus exec dnsmasq-container -- useradd -m -s /bin/bash user
2 incus exec dnsmasq-container -- bash -c 'echo "user:pass" | \
   chpasswd'
3 incus exec dnsmasq-container -- usermod -aG sudo user
4 incus exec dnsmasq-container -- usermod -aG docker user
```

3.5 Accessing the Container

Access the container's shell:

```
1 incus exec dnsmasq-container -- bash
```

3.6 Installing dnsmasq

Update the package list and install dnsmasq:

```
1 incus exec dnsmasq-container -- apt update
2 incus exec dnsmasq-container -- apt install dnsmasq -y
```

3.7 Configuring the Network with Netplan

Configure the container's network using Netplan to assign a static IP address. Create or edit the Netplan configuration file at `/etc/netplan/01-netcfg.yaml`:

```
1 incus exec dnsmasq-container -- nano /etc/netplan/01-netcfg.yaml
```

Add the following configuration:

```
1 network:
2   version: 2
3   ethernets:
4     eth0:
5       dhcp4: no
6       addresses:
7         - 192.168.1.10/24
8       routes:
9         - to: default
10           via: 192.168.1.1
11       nameservers:
12         addresses: [8.8.8.8, 8.8.4.4]
```

Apply the configuration:

```
1 incus exec dnsmasq-container -- netplan apply
```

3.8 Configuring dnsmasq

Edit the dnsmasq configuration file at `/etc/dnsmasq.conf`:

```
1 incus exec dnsmasq-container -- nano /etc/dnsmasq.conf
```

Add or modify the following settings to enable DNS and DHCP:

```
1 # DNS settings
2 domain-needed
3 bogus-priv
4 no-resolv
5 server=8.8.8.8
6 server=8.8.4.4
7 local=/example.local/
8 domain=example.local
9
10 # DHCP settings
11 dhcp-range=192.168.1.100,192.168.1.200,12h
```

```
12 dhcp-option=3,192.168.1.1
13 dhcp-option=6,8.8.8.8,8.8.4.4
```

Explanation:

- **domain-needed:** Prevents incomplete domain names from being sent to upstream DNS.
- **bogus-priv:** Blocks reverse DNS lookups for private IP ranges.
- **no-resolv:** Disables reading `/etc/resolv.conf`.
- **server:** Specifies upstream DNS servers (Google DNS in this case).
- **local and domain:** Configures a local domain.
- **dhcp-range:** Defines the IP range for DHCP clients (from 192.168.1.100 to 192.168.1.200, lease time 12 hours).
- **dhcp-option:** Sets the default gateway (option 3) and DNS servers (option 6).

3.9 Starting and Enabling dnsmasq

Restart and enable the `dnsmasq` service:

```
1 incus exec dnsmasq-container -- systemctl restart dnsmasq
2 incus exec dnsmasq-container -- systemctl enable dnsmasq
```

Verify that `dnsmasq` is running:

```
1 incus exec dnsmasq-container -- systemctl status dnsmasq
```

3.10 Testing the Configuration

Test DNS resolution from within the container:

```
1 incus exec dnsmasq-container -- nslookup example.local
    192.168.1.10
```

To test DHCP, connect a client device to the same network and verify that it receives an IP address in the range 192.168.1.100–192.168.1.200.

4 Troubleshooting

If `dnsmasq` fails to start:

- Check the logs: `incus exec dnsmasq-container - journalctl -u dnsmasq`.
- Ensure no other service is using port 53 (DNS) or 67 (DHCP).
- Verify the network configuration with `incus exec dnsmasq-container - ip a` and `incus exec dnsmasq-container - ping 8.8.8.8`.

5 Conclusion

This guide configures `dnsmasq` as a DNS and DHCP server in an Incus container on Debian. The Netplan configuration ensures proper network setup. For advanced configurations, refer to the `dnsmasq` documentation (`man dnsmasq`).