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 `deb1` using the following commands on the host:

```
incus create images:debian/12 deb1
incus config set deb1 security.syscalls.intercept.mount true
incus config set deb1 security.nesting true
incus config set deb1 security.privileged true
incus start deb1
```

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:

```
sudo iptables -A FORWARD -i incusbr0 -o wlo1 -j ACCEPT
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:

```
incus exec deb1 -- apt update
incus exec deb1 -- apt install -y \
    netplan.io \
    sudo vim nano git tmux mc zip unzip curl wget htop lynx \
    iproute2 termshark bridge-utils \
    python3 python3-ipython python3-pyroute2 python3-scapy \
    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":

```
incus exec deb1 -- 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:

```
sudo useradd -m -s /bin/bash -G sudo user && echo 'user:pass' |
   sudo chpasswd
```

3.5 Accessing the Container

Access the container's shell:

```
incus exec deb1 -- su - user
```

4 Setting Up a Veth Pair Between Container and Network Namespace

To enable direct communication between a container and a network namespace, a virtual Ethernet (`veth`) pair is created. The following Python script (`link.py`) is used to create a `veth` pair between the `deb1` (an Incus container) and the `ns1` network namespace, with interfaces named `vA` and `vB`.

```
sudo python3 link.py -n1 vA -t2 incus -ns2 deb1 -n2 vB
```

This command:

- Creates a `veth` pair with one end (`vA`) in the default namespace and the other end (`vB`) in the `deb1`'s network namespace.

- Ensures the interfaces are set up and operational, allowing network traffic to flow between the container and the `ns1` namespace (or default namespace if `ns1` is not explicitly created).

The script uses the `pyroute2` library to manage network interfaces and namespaces, and supports container types such as Incus, LXC, LXD, and Docker. Ensure the `deb1` is running in Incus before executing the command.

4.1 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`:

```
incus exec deb1 -- nano /etc/netplan/01-netcfg.yaml
```

Add the following configuration:

```
network:
  version: 2
  ethernets:
    vB:
      dhcp4: no
      addresses:
        - 192.168.1.10/24
      routes:
        - to: default
          via: 192.168.1.1
      nameservers:
        addresses: [8.8.8.8, 8.8.4.4]
```

Apply the configuration:

```
incus exec deb1 -- netplan apply
```

4.2 Installing dnsmasq

Update the package list and install `dnsmasq`:

```
incus exec deb1 -- apt update
incus exec deb1 -- apt install dnsmasq -y
```

4.3 Configuring dnsmasq

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

```
incus exec deb1 -- nano /etc/dnsmasq.conf
```

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

```
# DNS settings
domain-needed
bogus-priv
no-resolv
server=8.8.8.8
server=8.8.4.4
local=/example.local/
domain=example.local

# DHCP settings
dhcp-range=192.168.1.100,192.168.1.200,12h
dhcp-option=3,192.168.1.1
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).

4.4 System-Level Adjustments for Network Stability

In some cases, especially in nested or privileged containers, additional system-level adjustments are necessary to ensure proper network functionality and avoid conflicts.

To remount the `/sys` filesystem as read-write (required if certain networking tools fail due to mount restrictions):

```
sudo mount -o remount,rw /sys
sudo systemctl restart systemd-udevd
```

Additionally, to prevent DNS conflicts with `systemd-resolved`, which may interfere with `dnsmasq`, stop and disable the service:

```
sudo systemctl stop systemd-resolved
sudo systemctl disable systemd-resolved
```

This ensures that `dnsmasq` can bind to port 53 without conflicts. If you require `systemd-resolved`, consider configuring it to listen on a different interface or using socket activation.

4.5 Starting and Enabling dnsmasq

Restart and enable the `dnsmasq` service:

```
incus exec deb1 -- systemctl restart dnsmasq
incus exec deb1 -- systemctl enable dnsmasq
```

Verify that `dnsmasq` is running:

```
incus exec deb1 -- systemctl status dnsmasq
```

4.6 Testing the Configuration

Test DNS resolution from within the container:

```
incus exec deb1 -- 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`.

5 Troubleshooting

If `dnsmasq` fails to start:

- Check the logs: `incus exec deb1 - journalctl -u dnsmasq`.

- Ensure no other service is using port 53 (DNS) or 67 (DHCP).

- Verify the network configuration with `incus exec deb1 - ip a` and `incus exec deb1 - ping 8.8.8.8`.

6 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`).