2025/12/16 12:34 1/10 LXC Quick-Start

# **LXC Quick-Start**

- Basis: Frisch installiertes Debian 10 "Buster" (hier: als virtuelle Maschine)
- Zum Spielen reichen 2 Cores, 2GB RAM und 10GB HD.
   Jeweils das doppelte macht aber mehr Spaß;)
- Eine NIC, mit Internet-Anbindung

# Vorbereitung

## **Notwendige Pakete**

- 1xc Die zentrale Befehle zum Umgang mit LXContainer
- bridge-utils Zum Verwaltung von Network Bridges
- iptables (oder nftables oder ...) zum NAT der ausgehenden Verbindungen
- ggf. dnsmasq kleiner DHCP- und DNS-Server (wird auch von libvirt genutzt)

## Bridge als Netzwerk-Basis für die LXContainer

Vorteil dieses Setups ist, dass die Container erst mal nur vom lokalen Host angesprochen werden können und man sich keine Sorgen um Sicherheitslücken o.ä. "von außen angreifbar" machen muss.

Bei den meist verwendeten veth-Interfaces ("virtual ethernet") sieht man einen Teil im Container (meist "eth0", konfigurierbar)und einen Teil im Host-System (meist "veth?????"). Damit diese sauber ins Internet kommen und/oder untereinander kommunizieren können, brauchen wir eine "Bridge". Diese kann man sich im wesentlichen wie einen Switch (na gut, eher ein Hub) vorstellen, welcher in Software realisiert wird. Dazu legt man eben so ein Interface an, hier mit dem Namen "br-Ixc":

#### /etc/network/interfaces.d/br-lxc

```
iface br-lxc inet static
  address 192.168.42.1/24

bridge_ports none
  bridge_maxwait 1
  bridge_stp on

up /sbin/ip link set up dev br-lxc
```

Dann fährt man die Bridge hoch:

```
root@lxc-host:~# ifup br-lxc
Waiting for br-lxc to get ready (MAXWAIT is 1 seconds).
```

Last update: 2021/01/15 14:54

```
root@lxc-host:~# ip -4 -brief -c a ls
                 UNKNOWN
                                 127.0.0.1/8
enp1s0
                 UP
                                 192.168.99.42/24
br-lxc
                 DOWN
                                 192.168.42.1/24
root@lxc-host:~# brctl show br-lxc
                bridge id
bridge name
                                         STP enabled
                                                          interfaces
br-lxc
                8000.000000000000
                                         yes
```

- Später sieht man beim brctl show br-lxc unter "interfaces" dann die veth?????? der LXContainer.
- enp1s0 ist hier das externe Interfaces (private IP weil es ja auch nur eine virtuelle Maschine ist)

### **Routing & NAT**

Normalerweise ist das Routing von Paketen "durch den Linux-Kernel" abgeschaltet. Für uns heißt das, dass die LXContainer noch nicht ins Netz kommen. Wir müssen also das Routing ("IP forwarding") aktivieren und anschließend die internen IP-Adressen der Container noch auf unsere "öffentliche" Adresse NAT'en.

#### /etc/sysctl.conf

```
# Uncomment the next line to enable packet forwarding for IPv4
net.ipv4.ip_forward=1
```

Um die Änderung zu aktivieren, rufen wir sysctl eben von Hand auf - inkl. anschließender Kontrolle. Bei einem Reboot wird der Eintrag automatisch gelesen und gesetzt.

```
root@lxc-host:~# sysctl -p /etc/sysctl.conf
net.ipv4.ip_forward = 1
root@lxc-host:~# cat /proc/sys/net/ipv4/ip_forward
1
```

Last but not least das NAT'ing der internen IPs (siehe br-lxc-Konfiguration oben, IP-Netz 192.168.42.1/24) Richtung Internet

```
root@lxc-host:~# iptables -t nat -A POSTROUTING -s 192.168.42.1/24 -o en1ps0
-j MASQUERADE
```

- ACHTUNG Dieser Aufruf gilt nur bis zum Reboot!
- Permanent kann man das z. B. mit Hilfe des Pakets iptables-persistent speichern
- Alternativ trägt man in /etc/network/interfaces.d/br-lxc zusätzlich die Zeilen

```
up /sbin/iptables -t nat -A POSTROUTING -s 192.168.42.1/24 -o en1ps0 -j MASQUERADE down /sbin/iptables -t nat -D POSTROUTING -s 192.168.42.1/24 -o en1ps0 -j MASQUERADE
```

2025/12/16 12:34 3/10 LXC Quick-Start

ein

## **DHCP & DNS - dnsmasq**

Die meisten Images für Container gehen davon aus, dass IP-Adressen per DHCP vergeben werden. Außerdem müssten wir noch für einen DNS-Server/-Forwarder sorgen. Beides kann dnsmasq recht einfach bereitstellen. Nach der Installation fügt man folgende Zeilen am Ende der Datei /etc/dnsmasq.conf ein:

#### /etc/dnsmasq.conf

```
local=/br-lxc/
domain=br-lxc
expand-hosts
interface=br-lxc
dhcp-range=br-lxc,192.168.42.100,192.168.42.199,60m
dhcp-no-override
dhcp-authoritative
dhcp-option=option:dns-server,192.168.42.1
dhcp-option=option:router,192.168.42.1
```

Restart des Dienstes:

```
root@lxc-host # service dnsmasq restart
```

## **Container erstellen**

Der vermutlich am häufigsten aufgerufene Befehl ist lxc-ls. Sinnvoll ist, wenn man sich gleich den Aufruf lxc-ls - f ("fancy") merkt - oder sich ein Alias darauf anlegt. Zum jetzigen Zeitpunkt gibt dieser noch nichts aus, es gibt ja auch noch keinen Container.

Daher legen wir doch einfach mal einen Container (hier Alpine 3.11, AMD64) mit Hilfe des Befehls lxc-create an, kontrollieren dies und starten den Container:

```
root@lxc-host # lxc-create -n container1 -t download -- -d alpine -r 3.11 -a amd64
Setting up the GPG keyring
Downloading the image index
Downloading the rootfs
Downloading the metadata
The image cache is now ready
Unpacking the rootfs
```

- - bei lxc-create sorgt dafür, dass die Parameter -d, -r und -a an das Template download (selbst ein Shell-Script) übergeben werden.
- Der sleep 60 soll nur ein gewisses Warten signalisieren
- Der Container wird aber auch nach 600 oder 6000 Sekunden keine IP bekommen!

## In den Container wechseln

Warum bekommt der Container keine IP? Nun, gehen wir doch mal in den Container und sehen uns dort die Netzwerk-Konfiguration an:

```
root@lxc-host # lxc-attach container1
~ # ip a
1: lo: <L00PBACK> mtu 65536 qdisc noop state DOWN qlen 1000
        link/loopback 00:00:00:00:00 brd 00:00:00:00:00:00
~ # exit
root@lxc-host #
```

• Der Container hat schlicht und einfach nur das lo-Interface!

Will man eben nur schnell einen Befehl im Container ausführen, kann man dies auch direkt mit lxcattach machen:

• Der - - ist notwendig, sonst interpretiert lxc-attach die Option -a und da er diese nicht kennt, bricht er mit einer Fehlermeldung ab

## (Netzwerk-)Konfiguration des Containers

LXContainer werden unter /var/lib/lxc/<CONTAINERNAME>/config konfiguriert, auch die Netzwerk-Interfaces:

Printed on 2025/12/16 12:34

2025/12/16 12:34 5/10 LXC Quick-Start

#### /var/lib/lxc/container1/config

```
# Template used to create this container: /usr/share/lxc/templates/lxc-
download
# Parameters passed to the template: -d alpine -r 3.11 -a amd64
# Template script checksum (SHA-1):
273c51343604eb85f7e294c8da0a5eb769d648f3
# For additional config options, please look at lxc.container.conf(5)
# Uncomment the following line to support nesting containers:
#lxc.include = /usr/share/lxc/config/nesting.conf
# (Be aware this has security implications)
# Distribution configuration
lxc.include = /usr/share/lxc/config/common.conf
lxc.arch = linux64
# Container specific configuration
lxc.apparmor.profile = generated
lxc.apparmor.allow nesting = 1
lxc.rootfs.path = dir:/var/lib/lxc/container1/rootfs
lxc.uts.name = container1
# Network configuration
lxc.net.0.type = empty
```

• Ganz unten steht lxc.net.0.type = empty - es gibt kein Netzwerk!

lxc-create liest die Datei /etc/lxc/default.conf ein und übergibt die dort platzierten Parameter in der Container-Config. Und in dieser steht eben:

#### /etc/lxc/default.conf

```
lxc.net.0.type = empty
lxc.apparmor.profile = generated
lxc.apparmor.allow_nesting = 1
```

#### Man kann jetzt

- von Hand (und für jeden zukünftigen Container) diese Config anpassen
- eine Datei /etc/lxc/br-lxc.conf anlegen und diese dann beim Aufruf von lxc-create übergeben:

```
lxc-create -n container2 -t download -f /etc/lxc/br-lxc.conf -- -d
alpine -r 3.11 -a amd64
```

• die Datei /etc/lxc/default.conf anpassen

Für Fall (2 und) 3 würde es so aussehen:

#### Last update: 2021/01/15 14:54

# /etc/lxc/default.conf

```
lxc.apparmor.profile = generated
lxc.apparmor.allow_nesting = 1

lxc.net.0.type = veth
lxc.net.0.flags = up
# lxc.net.0.veth.pair = LXCxxx
lxc.net.0.name = eth0
lxc.net.0.hwaddr = 0e:0e:XX:XX:XX
lxc.net.0.link = br-lxc

# lxc.net.0.ipv4.address = 192.168.42.XXX
# lxc.net.0.ipv4.gateway = 192.168.42.1
```

- 0 bezeichnet das erste Interface im Container. Es kann also mehrere geben bis sechs funktioniert das auch auf jeden Fall ;)
- Ich verwende hier veth (mehr oder minder default)
- Man könnte dem Interface im Host einen Namen (veth.pair) geben. Das muss aber je Container erfolgen, daher hier auskommentiert.
- Im Container soll das Interface eth0 heißen
- Die MAC-Adresse hwaddr wird beim Anlegen festgelegt.
  - Die XX werden durch Zufallswerte aufgefüllt. So bekommt jeder Container eine eigene MAC-Adresse und (hoffentlich) beim Neustart die gleiche IP wie früher - abhängig von den Einstellungen des dnsmasq
  - Im ersten Byte sind 02, 06, 0A und 0E für private Nutzung (eben zum Beispiel Virtualisierung) reserviert.
- Man kann (statt DHCP im Container) auch von außen eine IP und Default-GW festlegen, hier auskommentiert, weil DHCP default ist

Mit dieser Einstellung (Netzwerk-Konfiguration in /etc/lxc/default.conf) bekommt ein neuer Container nun eine IP:

• Da das Image für Alpine 3.11 schon im Cache (/var/cache/lxc/) liegt, nimmt er direkt dieses beim Anlegen

```
root@lxc-host # lxc-attach container2 -- ping -c3 8.8.8.8
PING 8.8.8.8 (8.8.8.8): 56 data bytes
64 bytes from 8.8.8.8: seq=0 ttl=118 time=4.978 ms
```

2025/12/16 12:34 7/10 LXC Quick-Start

```
64 bytes from 8.8.8.8: seq=1 ttl=118 time=5.065 ms
64 bytes from 8.8.8.8: seq=2 ttl=118 time=5.308 ms

--- 8.8.8.8 ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss round-trip min/avg/max = 4.978/5.117/5.308 ms
```

# Welche Images gibt es?

Will man wissen, welche Images es überhaupt gibt, kann man das entweder auf der Web-Seite <a href="http://uk.images.linuxcontainers.org/">http://uk.images.linuxcontainers.org/</a> nachsehen, oder so tun, als ob man einen neuen Container anlegen möchte:

```
root@lxc-host # lxc-create -n xx -t download -- --list --no-validate
root@lxc-host # lxc-create -n xx -t download -- --list -a amd64 --no-
validate
[...]
root@lxc-host # lxc-create -n xx -t download -- --list -d devuan --no-
validate
[\ldots]
root@lxc-host # lxc-create -n xx -t download -- --list -d debian -a amd64 --
no-validate
DIST
       RELEASE ARCH
                        VARIANT BUILD
debian
       bullseye
                        amd64
                                default 20200625 05:24
debian
       buster amd64
                        default 20200625 05:24
                        default 20200625 05:24
debian jessie amd64
                        default 20200625 05:24
debian
       sid
                amd64
debian stretch amd64
                        default 20200625 05:24
```

- Ausgaben bis auf den letzten Fall gekürzt
- –no-validate damit's ein bisschen schneller geht (Nicht aber beim "echte" Anlegen!)
- Zum (weiteren) Filtern kann man die Parameter angeben:
  - ∘ -d (Distribution, z. B. "debian", "devuan", "alpine", …")
  - r (Release, z. B. "buster", "stretch" für Debian, "beowulf" für Devuan, "3.12, "3.11" für Alpine, …)
  - a (Architektur, heute praktisch immer "amd64". "i386" braucht zwar etwas weniger RAM, aber neue Ubuntus oder CentOS gibt es gar nicht mehr für 32-Bit)

#### Verschiedene Distris in LXContainern

Hier nochmal ein paar laufende Container mit Open-SSH in der erweiterten Ausgabe:

```
root@lxc-host # lxc-ls -f -F
name,state,pid,ram,swap,autostart,groups,interface,ipv4,ipv6,UNPRIVILEGED
```

```
NAME
           STATE
                   PID
                        RAM
                                 SWAP
                                        AUTOSTART GROUPS INTERFACE IPV4
IPV6 UNPRIVILEGED
Alpine3C
           RUNNING 3059 5.87MB 0.00MB 0
                                                         eth0, lo
192.168.42.144 -
                    false
Archlinux RUNNING 4704 39.93MB 0.00MB 0
                                                         eth0, lo
192.168.42.132 -
                    false
           RUNNING 4765 50.61MB 0.00MB 0
Cent0S8
                                                         eth0, lo
192.168.42.133 -
                    false
Debian10
           RUNNING 3340 83.50MB 0.00MB 0
                                                         eth0, lo
192.168.42.130 -
                    false
           RUNNING 3414 25.03MB 0.00MB 0
Devuan10
                                                         eth0, lo
192.168.42.131 -
                    false
Voidlinux RUNNING 5271 7.46MB 0.00MB 0
                                                         eth0, lo
192.168.42.158 -
                    false
```

• Deswegen verwende ich immer häufiger Alpine :D

# Template "oci"

Mit dem Template "oci" kann man aus diversen Container-Registries (z. B. https://hub.docker.com) Images herunterladen und diese als LXContainer auf die Platte legen. Dies klappt nicht immer auf Anhieb, die meisten sind dann für lxc-execute konfiguriert (nur ein Command im Container), ggf. muss noch DHCP aktiviert werden, usw.

Damit dies auf Debian 10/Buster funktioniert, muss man das Paket "skopeo" händisch aus Testing/Bullseye installieren (http://ftp.de.debian.org/debian/pool/main/s/skopeo/), dazu noch umoci und die Datei /etc/containers/policy.json anlegen:

#### /etc/containers/policy.json

http://vvv.lusc.de/dokuwiki/

2025/12/16 12:34 9/10 LXC Quick-Start

## **Reverse Proxy**

Laufen in den LXC-Containern Web-Anwendungen, müssen diese natürlich noch irgendwie "ins Web" gebracht werden. Dafür gibt es verschiedenste Wege. Einer davon wäre, auf dem Host einen Apache zu installieren. Dieser kann dann mit einem VHost Reverse-Proxy für einen oder mehrere Container sein. Je nach Anwendungsfall müssen noch weitere Einträge vorgenommen werden, z.B. "Websockets".

/etc/apache2/sites-enabled/subdom.lusc.de.conf

```
<IfModule ssl module>
    <VirtualHost *:443>
        DocumentRoot "/var/www/empty"
        <Directory "/var/www/empty">
            Options Indexes FollowSymLinks
            AllowOverride None
            Require all granted
        </Directory>
        ServerName subdom.lusc.de
        ServerAlias www.subdom.lusc.de
        ServerAdmin webmaster@localhost
        ErrorLog ${APACHE_LOG_DIR}/subdom.lusc.de.error.log
        CustomLog ${APACHE_LOG_DIR}/subdom.lusc.de.access.log combined
        <IfModule http2 module>
            Protocols h2 http/1.1
        </IfModule>
        SSLEngine on
        SSLCertificateFile
/var/lib/dehydrated/certs/subdom.lusc.de/cert.pem
        SSLCertificateKeyFile
/var/lib/dehydrated/certs/subdom.lusc.de/privkey.pem
        SSLCertificateChainFile
/var/lib/dehydrated/certs/subdom.lusc.de/chain.pem
        SSLProtocol
                           All -SSLv2 -SSLv3 -TLSv1 -TLSv1.1
        SSLCipherSuite
EECDH+AESGCM: EDH+AESGCM: AES256+EECDH: AES256+EDH
        SSLHonorCipherOrder
                               0n
        SSLSessionTickets
                             0ff
        ProxyPreserveHost
                             0n
        RequestHeader
                                  X-Forwarded-For
                         unset
                                X-Forwarded-Proto
        RequestHeader
                                                      https
                         set
        ProxyPass
                              http://192.168.1.80/
                                 http://192.168.1.80/
        ProxyPassReverse
                            /
```

```
</VirtualHost>
</IfModule>
<VirtualHost *:80>
    DocumentRoot "/var/www/empty"
    <Directory "/var/www/empty">
        Options Indexes FollowSymLinks
        AllowOverride None
        Require all granted
    </Directory>
    ServerName subdom.lusc.de
    ServerAlias www.subdom.lusc.de
    ServerAdmin webmaster@localhost
    ErrorLog ${APACHE LOG DIR}/subdom.lusc.de.error.log
    CustomLog ${APACHE LOG DIR}/subdom.lusc.de.access.log combined
    <IfModule http2 module>
        Protocols h2c http/1.1
    </IfModule>
    <IfModule ssl_module>
                         https://subdom.lusc.de/
        Redirect /
    </IfModule>
</VirtualHost>
# vim: syntax ft=apache noet
```

From:

http://vvv.lusc.de/dokuwiki/ - LUSC

Permanent link:

http://vvv.lusc.de/dokuwiki/orga/2020/lxc?rev=1610722485

Last update: **2021/01/15 14:54** 

