Creating a full LXD server On Rocky Linux (Ukrainian version)

A book from the Documentation Team

Version : 2024/04/29

Rocky Documentation Team

Table of contents

1. Licence

RockyLinux offers Linux courseware for trainers or people wishing to learn how to administer a Linux system on their own.

The documents and their sources are freely downloadable from:

- https://docs.rockylinux.org
- https://github.com/rocky-linux/documentation

Our media sources are hosted at github.com. You'll find the source code repository where the version of this document was created.

From these sources, you can generate your own personalized training material using mkdocs. You will find instructions for generating your document here.

How can I contribute to the documentation project?

You'll find all the information you need to join us on our git project home page.

We wish you all a pleasant reading and hope you enjoy the content.

2. Створення повноцінного сервера LXD

2.1 Вступ

LXD найкраще описано на офіційному сайті, але розглядайте його як контейнерну систему, яка надає переваги віртуальних серверів у контейнері.

Він дуже потужний, і з правильним апаратним забезпеченням і налаштуваннями його можна використовувати для запуску багатьох екземплярів сервера на одному апаратному забезпеченні. Якщо ви об’єднаєте це з сервером snapshot, у вас також буде набір контейнерів, які можна розкрутити майже негайно, якщо ваш основний сервер вийде з ладу.

(Не слід сприймати це як традиційне резервне копіювання. Вам усе ще потрібна якась звичайна система резервного копіювання, наприклад rsnapshot.)

Крива навчання для LXD може бути трохи крутою, але ця книга спробує дати вам велику кількість знань, щоб допомогти вам розгорнути та використовувати LXD на Rocky Linux.

Для тих, хто хоче використовувати LXD як лабораторне середовище на власних ноутбуках або робочих станціях, перегляньте Додаток A: налаштування робочої станції.

2.2 Передумови та припущення

- Один сервер Rocky Linux, гарно налаштований. Вам слід розглянути окремий жорсткий диск для дискового простору ZFS (вам потрібно, якщо ви використовуєте ZFS) у робочому середовищі. І так, ми припускаємо, що це чистий сервер, а не VPS.
- Це слід вважати розширеною темою, але ми зробили все можливе, щоб зробити її максимально зрозумілою для всіх. Тим не менш, знання кількох основних речей про керування контейнерами займе у вас довгий шлях.
- Ви маєте добре володіти командним рядком на своїй машині (машинах) і вільно володіти редактором командного рядка. (У цьому прикладі ми

використовуємо *vi*, але ви можете замінити його но свій улюблений редактор.)

- Ви повинні бути непривілейованим користувачем для більшості процесів LXD. Якщо не зазначено, вводьте команди LXD як непривілейований користувач. Ми припускаємо, що ви ввійшли як користувач з іменем "lxdadmin" для команд LXD. Більшість налаштувань *виконується* від імені користувача root, доки ви не пройдете ініціалізацію LXD.
- Для ZFS переконайтеся, що безпечне завантаження UEFI НЕ ввімкнено. В іншому випадку вам доведеться підписати модуль ZFS, щоб змусити його завантажити.
- Здебільшого ми використовуємо контейнери на основі Rocky Linux.

2.3 Короткий Огляд

- **Розділ 1: Встановлення та налаштування** стосується встановлення основного сервера. Загалом, правильний спосіб зробити LXD у виробництві — мати як основний сервер, так і сервер знімків.
- **Розділ 2: Налаштування ZFS** розповідає про налаштування та налаштування ZFS. ZFS — це диспетчер логічних томів і файлова система з відкритим вихідним кодом, створена Sun Microsystems спочатку для своєї операційної системи Solaris.
- **Розділ 3: Ініціалізація LXD і налаштування користувача** стосується базової ініціалізації та параметрів, а також налаштування непривілейованого користувача, якого ви використовуватимете протягом більшої частини решти процесу
- **Розділ 4: Налаштування брандмауера** стосується параметрів налаштування як `iptables`, так і `firewalld`, але ми рекомендуємо використовувати `firewalld` для всіх поточних версій Rocky Linux.
- **Розділ 5: Налаштування образів і керування ними** описує процес встановлення образів ОС до контейнера та їх налаштування.
- **Розділ 6: Профілі** розповідає про додавання профілів і їх застосування до контейнерів, а також охоплює macvlan і його важливість для IP-адресації у вашій LAN або WAN

- **Розділ 7: Параметри конфігурації контейнера** коротко описує деякі основні параметри конфігурації для контейнерів і пропонує деякі плюси та мінуси для зміни параметрів конфігурації.
- **Розділ 8: Знімки контейнерів** детально описує процес створення знімків для контейнерів на основному сервері.
- **Розділ 9. Сервер Snapshot** розповідає про налаштування та конфігурацію сервера snapshot, а також про те, як створити симбіотичний зв’язок між основним і сервером snapshot.
- **Розділ 10: Автоматизація Snapshots** розповідає про автоматизацію створення snapshot і заповнення сервера snapshot новими snapshots.
- **Додаток A: налаштування робочої станції** технічно не є частиною документів робочого сервера, але пропонує рішення для людей, які хочуть у простий спосіб створити лабораторію контейнерів LXD на своїх особистих ноутбуках або робочіх станціях.

2.4 Висновки

Ви можете використовувати ці розділи для ефективного налаштування основного сервера LXD на рівні підприємства та знімків. У процесі ви дізнаєтесь багато про LXD, і ми *розглянемо* багато функцій. Просто майте на увазі, що ще багато чого потрібно дізнатися, і розглядайте ці документи як відправну точку.

Найбільша перевага LXD полягає в тому, що його економічно використовувати на сервері, він дозволяє швидко інсталювати ОС, його можна використовувати для кількох автономних серверів додатків, що працюють на одному «голому металі», і все це належним чином використовує це обладнання для максимального використання.

3. Розділ 1: Встановлення та налаштування

У цьому розділі вам потрібно буде бути користувачем root або вміти використовувати *sudo* для отримання прав root.

3.1 Встановлення репозиторіїв EPEL і OpenZFS

Для LXD потрібен репозиторій EPEL (Extra Packages for Enterprise Linux), який легко встановити за допомогою:

```
dnf install epel-release
```

Після встановлення перевірте наявність оновлень:

```
dnf upgrade
```

Якщо під час процесу оновлення ядро було оновлено, перезавантажте сервер.

3.1.1 Репозиторій OpenZFS для 8 і 9

Встановіть репозиторій OpenZFS за допомогою:

```
dnf install https://zfsonlinux.org/epel/zfs-release-2-2$(rpm --eval "%
{dist}").noarch.rpm
```

3.2 Встановіть snapd, dkms, vim і kernel-devel

Для Rocky Linux LXD потрібно інсталювати з оснастки. З цієї причини нам потрібно встановити `snapd` (і кілька інших корисних програм) з:

```
dnf install snapd dkms vim kernel-devel
```

А тепер увімкніть і запустіть snapd:

```
systemctl enable snapd
```

А потім запустіть:

```
systemctl start snapd
```

Перш ніж продовжити, перезавантажте сервер.

3.3 Встановлення LXD

Встановлення LXD вимагає використання команди snap. На даний момент ми просто встановлюємо його, ми не виконуємо налаштування:

```
snap install lxd
```

3.4 Встановлення OpenZFS

```
dnf install zfs
```

3.5 Налаштування середовища

Більшості параметрів ядра сервера недостатньо для запуску великої кількості контейнерів. Якщо ми з самого початку припустимо, що будемо використовувати наш сервер у робочому стані, тоді нам потрібно внести ці зміни заздалегідь, щоб уникнути таких помилок, як «Забагато відкритих файлів».

На щастя, налаштувати параметри для LXD легко за допомогою кількох змін файлів і перезавантаження.

3.5.1 Модифікація limits.conf

Перший файл, який нам потрібно змінити, це файл limits.conf. Цей файл самодокументований, тому подивіться пояснення у файлі щодо того, що він робить. Перегляньте пояснення в коментарях до файлу, щоб зрозуміти, що робить цей файл. Щоб внести зміни, введіть:

```
vi /etc/security/limits.conf
```

Весь цей файл позначено/закоментовано, а внизу показано поточні налаштування за замовчуванням. У порожньому місці над маркером кінця файлу (#End of file) нам потрібно додати власні налаштування. Коли ви закінчите, кінець файлу виглядатиме як показано нижче:

```
# Modifications made for LXD

*               soft    nofile          1048576
*               hard    nofile          1048576
root            soft    nofile          1048576
root            hard    nofile          1048576
*               soft    memlock         unlimited
*               hard    memlock         unlimited
```

Збережіть зміни та вийдіть. (`SHIFT:wq!` для *vi*)

3.5.2 Модифікація sysctl.conf за допомогою 90-lxd.override.conf

За допомогою *systemd* ми можемо вносити зміни до загальної конфігурації нашої системи та параметрів ядра *без* зміни основного файлу конфігурації. Замість цього ми розмістимо наші параметри в окремому файлі, який просто замінить потрібні нам налаштування.

Щоб внести ці зміни в ядро, ми створимо файл під назвою *90-lxd-override.conf* у /etc/sysctl.d. Щоб зробити цей тип:

```
vi /etc/sysctl.d/90-lxd-override.conf
```

RL 9 і MAX значення `net.core.bpf_jit_limit`

Через останні оновлення безпеки ядра максимальне значення net.core.bpf_jit_limit становить 1000000000. Будь ласка, налаштуйте це значення у самодокументованому файлі нижче, якщо ви використовуєте Rocky Linux 9.x. Якщо ви встановите його вище цього ліміту **АБО**, якщо ви взагалі не встановите його, за умовчанням буде встановлено системне значення за замовчуванням 264241152, чого може бути недостатньо, якщо ви запускаєте велику кількість контейнерів.

Розмістіть наступний вміст у цьому файлі. Зверніть увагу: якщо вам цікаво, що ми тут робимо, вміст файлу нижче є самодокументованим:

```
## The following changes have been made for LXD ##

# fs.inotify.max_queued_events specifies an upper limit on the number of events
that can be queued to the corresponding inotify instance
```

```
 - (default is 16384)

fs.inotify.max_queued_events = 1048576

# fs.inotify.max_user_instances This specifies an upper limit on the number of
inotify instances that can be created per real user ID -
(default value is 128)

fs.inotify.max_user_instances = 1048576

# fs.inotify.max_user_watches specifies an upper limit on the number of watches
that can be created per real user ID - (default is 8192)

fs.inotify.max_user_watches = 1048576

# vm.max_map_count contains the maximum number of memory map areas a process
may have. Memory map areas are used as a side-effect of cal
ling malloc, directly by mmap and mprotect, and also when loading shared
libraries - (default is 65530)

vm.max_map_count = 262144

# kernel.dmesg_restrict denies container access to the messages in the kernel
ring buffer. Please note that this also will deny access t
o non-root users on the host system - (default is 0)

kernel.dmesg_restrict = 1

# This is the maximum number of entries in ARP table (IPv4). You should
increase this if you create over 1024 containers.

net.ipv4.neigh.default.gc_thresh3 = 8192

# This is the maximum number of entries in ARP table (IPv6). You should
increase this if you plan to create over 1024 containers.Not nee
ded if not using IPv6, but...

net.ipv6.neigh.default.gc_thresh3 = 8192

# This is a limit on the size of eBPF JIT allocations which is usually set to
PAGE_SIZE * 40000. Set this to 1000000000 if you are running Rocky Linux 9.x

net.core.bpf_jit_limit = 3000000000

# This is the maximum number of keys a non-root user can use, should be higher
than the number of containers

kernel.keys.maxkeys = 2000
```

```
# This is the maximum size of the keyring non-root users can use

kernel.keys.maxbytes = 2000000

# This is the maximum number of concurrent async I/O operations. You might need
to increase it further if you have a lot of workloads th
at use the AIO subsystem (e.g. MySQL)

fs.aio-max-nr = 524288
```

Збережіть зміни та вийдіть.

На цьому етапі вам слід перезавантажити сервер.

3.5.3 Перевірка значень *sysctl.conf*

Після завершення перезавантаження знову увійдіть на сервер. Нам потрібно перевірити, чи справді наш файл заміни виконав роботу.

Це легко зробити. Немає необхідності перевіряти кожне налаштування, якщо ви цього не хочете, але перевірка кількох підтвердить, що налаштування було змінено. Це робиться за допомогою команди *sysctl*:

```
sysctl net.core.bpf_jit_limit
```

Який повинен показати вам:

```
net.core.bpf_jit_limit = 3000000000
```

Зробіть те саме з кількома іншими параметрами у файлі перевизначення (вище), щоб перевірити, чи внесено зміни.

4. Розділ 2: Налаштування ZFS

У цьому розділі вам потрібно буде бути користувачем root або вміти використовувати `sudo`, щоб стати root.

Якщо ви вже інсталювали ZFS, цей розділ проведе вас через налаштування ZFS.

4.1 Увімкнення ZFS і налаштування пулу

Спочатку введіть цю команду:

```
/sbin/modprobe zfs
```

Якщо помилок немає, він повернеться до підказки та нічого не повторить. Якщо ви отримали помилку, зупиніться зараз і почніть усунення несправностей. Знову переконайтеся, що безпечне завантаження вимкнено. Це буде найімовірнішим винуватцем.

Далі нам потрібно поглянути на диски в нашій системі, визначити, яка на них завантажена ОС і що можна використовувати для пулу ZFS. Ви зробите це за допомогою `lsblk`:

```
lsblk
```

Що має повернути щось на зразок цього (ваша система буде іншою!):

```
AME   MAJ:MIN RM   SIZE RO TYPE MOUNTPOINT
loop0    7:0    0  32.3M  1 loop /var/lib/snapd/snap/snapd/11588
loop1    7:1    0  55.5M  1 loop /var/lib/snapd/snap/core18/1997
loop2    7:2    0  68.8M  1 loop /var/lib/snapd/snap/lxd/20037
sda      8:0    0 119.2G  0 disk
├─sda1   8:1    0   600M  0 part /boot/efi
├─sda2   8:2    0     1G  0 part /boot
├─sda3   8:3    0  11.9G  0 part [SWAP]
├─sda4   8:4    0     2G  0 part /home
└─sda5   8:5    0 103.7G  0 part /
sdb      8:16   0 119.2G  0 disk
├─sdb1   8:17   0 119.2G  0 part
└─sdb9   8:25   0     8M  0 part
```

```
sdc      8:32   0 149.1G  0 disk
└─sdc1   8:33   0 149.1G  0 part
```

У цьому списку ви можете побачити, що */dev/sda* використовується операційною системою. Ви збираєтеся використовувати */dev/sdb* для нашого zpool. Зверніть увагу: якщо у вас багато доступних жорстких дисків, ви можете розглянути можливість використання raidz (програмний рейд спеціально для ZFS).

Це виходить за рамки цього документа, але безумовно розглядається для виробництва. Він пропонує кращу продуктивність і резервування. Наразі створіть свій пул на одному пристрої, який ви визначили:

```
zpool create storage /dev/sdb
```

Це означає створити пул під назвою «сховище», тобто ZFS, на пристрої */dev/sdb*.

Після створення пулу знову перезавантажте сервер.

5. Розділ 3: Ініціалізація LXD і налаштування користувача

У цьому розділі вам потрібно мати права root або вміти викорисовувати `sudo`, щоб стати root. Крім того, припускається, що ви налаштували пул зберігання даних ZFS, описаний у главі 2. Ви можете використовувати інший пул зберігання, якщо ви вирішили не використовувати ZFS, але вам потрібно буде внести зміни до запитань і відповідей щодо ініціалізації.

5.1 Ініціалізація LXD

Серверне середовище налаштовано. Ви готові ініціалізувати LXD. Це автоматизований сценарій, який задає низку запитань, щоб запустити ваш екземпляр LXD:

```
lxd init
```

Ось запитання та наші відповіді щодо сценарію з невеликими поясненнями, де це необхідно:

```
Бажаєте використовувати кластеризацію LXD? (Would you like to use LXD
clustering?) (yes/no) [default=no]:
```

Якщо вас цікавить кластеризація, проведіть додаткові дослідження щодо цього тут

```
Бажаєте налаштувати новий пул зберігання? (Do you want to configure a new
storage pool?) (yes/no) [default=yes]:
```

Це здається неінтуїтивним. Ви вже створили свій пул ZFS, але це стане зрозумілим у наступному питанні. Прийняти значення за замовчуванням.

```
Ім’я нового пулу сховищ [default=default]: storage
```

Можна залишити це значення «за замовчуванням», але для ясності краще використовувати ту саму назву, яку ви дали нашому пулу ZFS.

```
Назва серверної частини сховища для використання (btrfs, dir, lvm, zfs, ceph)
[default=zfs]:
```

Ви хочете прийняти значення за замовчуванням.

```
Створити новий пул ZFS? (yes/no) [default=yes]: no
```

Тут вступає в дію вирішення попереднього питання про створення пулу зберігання.

```
Назва існуючого пулу або набору даних ZFS: storage
Бажаєте підключитися до сервера MAAS? (yes/no) [default=no]:
```

Metal As A Service (MAAS) виходить за рамки цього документа.

```
Бажаєте створити новий міст локальної мережі? (yes/no) [default=yes]:
Як мав би називатися новий міст? [default=lxdbr0]:
Яку адресу IPv4 слід використовувати? (CIDR subnet notation, “auto” or “none”)
[default=auto]:
Яку адресу IPv6 слід використовувати? (CIDR subnet notation, “auto” or “none”)
[default=auto]: none
```

Якщо ви хочете використовувати IPv6 у своїх контейнерах LXD, ви можете ввімкнути цю опцію. Це залежить від вас.

```
Бажаєте, щоб сервер LXD був доступний через мережу? (yes/no) [default=no]: yes
```

Це необхідно для створення snapshot сервера.

```
Address to bind LXD to (not including port) [default=all]:
Port to bind LXD to [default=8443]:
Trust password for new clients:
Again:
```

Цей пароль довіри означає, як ви підключатиметеся до snapshot сервера або повертатиметеся із snapshot сервера. Встановіть це з тим, що має сенс у вашому оточенні. Збережіть цей запис у безпечному місці, наприклад у менеджері паролів.

```
Бажаєте, щоб застарілі кешовані зображення оновлювалися автоматично? (yes/no)
[default=yes]
Чи бажаєте ви, щоб надруковано передзапуск YAML "lxd init"? (yes/no)
[default=no]:
```

5.2 Налаштування прав користувача

Перш ніж продовжити, нам потрібно створити нашого користувача "lxdadmin" і переконатися, що він має необхідні привілеї. Нам потрібен користувач «lxdadmin», щоб мати змогу використовувати *sudo* для root і він повинен бути членом групи lxd. Щоб додати користувача та переконатися, що він є членом обох груп, виконайте:

```
useradd -G wheel,lxd lxdadmin
```

Потім встановіть пароль:

```
passwd lxdadmin
```

Як і інші паролі, збережіть цей пароль у безпечному місці.

6. Розділ 4: Налаштування брандмауера

У цьому розділі вам потрібно мати права root або вміти використовувати `sudo`, щоб стати root.

Як і будь-який сервер, вам потрібно переконатися, що він захищений від зовнішнього світу та вашої локальної мережі. У той час як наш приклад сервера має лише інтерфейс локальної мережі, цілком можливо мати два інтерфейси, по одному для вашої мережі LAN та WAN.

!!! "warning "Примітка щодо Rocky Linux 9.x і `iptables`""

```
Починаючи з Rocky Linux 9.0, `iptables` та всі пов’язані з ним утиліти офіційно
застаріли. Це означає, що в наступних версіях операційної системи вони взагалі
зникнуть. Попередня версія цього документа містила вказівки щодо налаштування
`iptables`, але тепер їх видалено.

Для всіх поточних версій Rocky Linux рекомендується використовувати
`firewalld`.
```

6.1 Налаштування брандмауера - `firewalld`

Для правил *firewalld* вам потрібно використовувати цю базову процедуру або бути знайомими з цими концепціями. Наші припущення такі: мережа LAN 192.168.1.0/24 і міст під назвою lxdbr0. Щоб було зрозуміло, у вас може бути багато інтерфейсів на вашому сервері LXD, один з яких, можливо, звернений до вашої WAN. Ви також збираєтеся створити зону для мостової та локальної мереж. Це лише для ясності зони. Інші назви зон насправді не застосовуються. Ця процедура передбачає, що ви вже знаєте основи *брандмауера*.

```
firewall-cmd --new-zone=bridge --permanent
```

Вам потрібно перезавантажити брандмауер після додавання зони:

```
firewall-cmd --reload
```

Ви хочете дозволити весь рух з мосту. Просто додайте інтерфейс і змініть ціль із «default» на «ACCEPT»:

!!! "warning "важливо""

```
Зміна цілі зони `firewalld` *має* виконуватися за допомогою параметра `--
permanent`, тож ми можемо також просто ввести цей прапорець в інших наших
командах і відмовитися від `--runtime-to-permanent` варіант.
```

Примітка

Якщо вам потрібно створити зону, у якій ви бажаєте дозволити повний доступ до інтерфейсу чи джерела, але не хочете вказувати будь-які протоколи чи служби, тоді ви *має* змінити ціль із «default» на «ACCEPT». Те саме стосується «DROP» та «REJECT» для певного IP-блоку, для якого у вас є власні зони. Щоб було зрозуміло, зона «відкидання» подбає про це за вас, доки ви не використовуєте спеціальну зону.

```
firewall-cmd --zone=bridge --add-interface=lxdbr0 --permanent
firewall-cmd --zone=bridge --set-target=ACCEPT --permanent
```

Якщо припустити, що помилок немає і все працює, просто перезавантажте:

```
firewall-cmd --reload
```

Якщо ви зараз перерахуєте свої правила за допомогою `firewall-cmd --zone=bridge --list-all`, ви побачите:

```
bridge (active)
  target: ACCEPT
  icmp-block-inversion: no
  interfaces: lxdbr0
  sources:
  services:
  ports:
  protocols:
  forward: no
  masquerade: no
  forward-ports:
  source-ports:
  icmp-blocks:
  rich rules:
```

Зверніть увагу, що ви також хочете дозволити свій локальний інтерфейс. Знову ж таки, включені зони названі невідповідно для цього. Створіть зону та використовуйте вихідний діапазон IP для локального інтерфейсу, щоб забезпечити доступ:

```
firewall-cmd --new-zone=local --permanent
firewall-cmd --reload
```

Додайте вихідні IP-адреси для локального інтерфейсу та змініть ціль на «ACCEPT»:

```
firewall-cmd --zone=local --add-source=127.0.0.1/8 --permanent
firewall-cmd --zone=local --set-target=ACCEPT --permanent
firewall-cmd --reload
```

Давайте перерахуємо «локальну» зону, щоб переконатися, що ваші правила існують за допомогою `firewall-cmd --zone=local --list all`, яка покаже:

```
local (active)
  target: ACCEPT
  icmp-block-inversion: no
  interfaces:
  sources: 127.0.0.1/8
  services:
  ports:
  protocols:
  forward: no
  masquerade: no
  forward-ports:
  source-ports:
  icmp-blocks:
  rich rules:
```

Ви хочете дозволити SSH із нашої надійної мережі. Тут ми будемо використовувати вихідні IP-адреси та вбудовану «довірену» зону. Ціль для цієї зони вже "ACCEPT" за замовчуванням.

```
firewall-cmd --zone=trusted --add-source=192.168.1.0/24
```

Додайте послугу в зону:

```
firewall-cmd --zone=trusted --add-service=ssh
```

Якщо все працює, перемістіть свої правила на постійні та перезавантажте правила:

```
firewall-cmd --runtime-to-permanent
firewall-cmd --reload
```

Перелік вашої «довіреної» зони покаже:

```
trusted (active)
  target: ACCEPT
  icmp-block-inversion: no
  interfaces:
  sources: 192.168.1.0/24
  services: ssh
  ports:
  protocols:
  forward: no
  masquerade: no
  forward-ports:
  source-ports:
  icmp-blocks:
  rich rules:
```

За замовчуванням «загальнодоступна» зона ввімкнена, а SSH дозволено. З міркувань безпеки ви не хочете, щоб SSH дозволявся в «загальнодоступній» зоні. Переконайтеся, що ваші зони правильні та що доступ до сервера ви отримуєте через одну з IP-адрес локальної мережі (у нашому прикладі). Ви можете заблокувати себе на сервері, якщо не підтвердите це, перш ніж продовжити. Коли ви впевнені, що маєте доступ із правильного інтерфейсу, видаліть SSH із загальнодоступної зони:

```
firewall-cmd --zone=public --remove-service=ssh
```

Перевірте доступ і переконайтеся, що ви не заблоковані. Якщо ні, перенесіть свої правила на постійні, перезавантажте та винесіть зону "публічно", щоб забезпечити видалення SSH:

```
firewall-cmd --runtime-to-permanent
firewall-cmd --reload
firewall-cmd --zone=public --list-all
```

На вашому сервері можуть бути інші інтерфейси. Ви можете використовувати вбудовані зони, де це доцільно, але якщо назви не здаються логічними, ви

точно можете додати зони. Просто пам’ятайте, що якщо у вас немає служб або протоколів, які вам потрібно дозволити або відхилити спеціально, вам потрібно буде змінити цільову зону. Якщо використання інтерфейсів працює, як у випадку з мостом, ви можете це зробити. Якщо вам потрібен детальніший доступ до служб, натомість використовуйте вихідні IP-адреси.

7. Розділ 5: Налаштування та керування зображеннями

У цій главі вам потрібно буде виконувати команди як непривілейований користувач ("lxdadmin", якщо ви читаєте цю книгу з самого початку).

7.1 Список доступних зображень

Можливо, вам кортить почати роботу з контейнером. Є багато можливостей контейнерної операційної системи. Щоб зрозуміти, скільки існує можливостей, введіть цю команду:

```
lxc image list images: | more
```

Щоб переглянути список, натисніть пробіл. Цей список контейнерів і віртуальних машин продовжує зростати.

Останнє, що ви хочете зробити, це переглянути сторінку в пошуках образу контейнера для встановлення, особливо якщо ви знаєте образ, який хочете створити. Змініть команду, щоб відображати лише параметри встановлення Rocky Linux:

```
lxc image list images: | grep rocky
```

Це відкриває набагато більш керований список:

```
| rockylinux/8 (3 more)                     | 0ed2f148f7c6 | yes    | Rockylinux
8 amd64 (20220805_02:06)          | x86_64       | CONTAINER       | 128.68MB
| Aug 5, 2022 at 12:00am (UTC)  |
| rockylinux/8 (3 more)                     | 6411a033fdf1 | yes    | Rockylinux
8 amd64 (20220805_02:06)          | x86_64       | VIRTUAL-MACHINE | 643.15MB
| Aug 5, 2022 at 12:00am (UTC)  |
| rockylinux/8/arm64 (1 more)               | e677777306cf | yes    | Rockylinux
8 arm64 (20220805_02:29)          | aarch64      | CONTAINER       | 124.06MB
| Aug 5, 2022 at 12:00am (UTC)  |
| rockylinux/8/cloud (1 more)               | 3d2fe303afd3 | yes    | Rockylinux
8 amd64 (20220805_02:06)          | x86_64       | CONTAINER       | 147.04MB
| Aug 5, 2022 at 12:00am (UTC)  |
| rockylinux/8/cloud (1 more)               | 7b37619bf333 | yes    | Rockylinux
8 amd64 (20220805_02:06)          | x86_64       | VIRTUAL-MACHINE | 659.58MB
| Aug 5, 2022 at 12:00am (UTC)  |
```

```
| rockylinux/8/cloud/arm64                | 21c930b2ce7d | yes    | Rockylinux
8 arm64 (20220805_02:06)          | aarch64      | CONTAINER       | 143.17MB
| Aug 5, 2022 at 12:00am (UTC)  |
| rockylinux/9 (3 more)                   | 61b0171b7eca | yes    | Rockylinux
9 amd64 (20220805_02:07)          | x86_64       | VIRTUAL-MACHINE | 526.38MB
| Aug 5, 2022 at 12:00am (UTC)  |
| rockylinux/9 (3 more)                   | e7738a0e2923 | yes    | Rockylinux
9 amd64 (20220805_02:07)          | x86_64       | CONTAINER       | 107.80MB
| Aug 5, 2022 at 12:00am (UTC)  |
| rockylinux/9/arm64 (1 more)             | 917b92a54032 | yes    | Rockylinux
9 arm64 (20220805_02:06)          | aarch64      | CONTAINER       | 103.81MB
| Aug 5, 2022 at 12:00am (UTC)  |
| rockylinux/9/cloud (1 more)             | 16d3f18f2abb | yes    | Rockylinux
9 amd64 (20220805_02:06)          | x86_64       | CONTAINER       | 123.52MB
| Aug 5, 2022 at 12:00am (UTC)  |
| rockylinux/9/cloud (1 more)             | 605eadf1c512 | yes    | Rockylinux
9 amd64 (20220805_02:06)          | x86_64       | VIRTUAL-MACHINE | 547.39MB
| Aug 5, 2022 at 12:00am (UTC)  |
| rockylinux/9/cloud/arm64                | db3ce70718e3 | yes    | Rockylinux
9 arm64 (20220805_02:06)          | aarch64      | CONTAINER       | 119.27MB
| Aug 5, 2022 at 12:00am (UTC)  |
```

7.2 Встановлення, перейменування та перелік зображень

Для першого контейнера ви збираєтеся використовувати "rockylinux/8". Щоб його встановити, ми *можемо* використати:

```
lxc launch images:rockylinux/8 rockylinux-test-8
```

Це створить контейнер на основі Rocky Linux під назвою "rocky linux-test-8". Ви можете перейменувати контейнер після його створення, але спочатку потрібно зупинити контейнер, який запускається автоматично після створення.

Щоб запустити контейнер вручну, використовуйте:

```
lxc start rockylinux-test-8
```

Щоб перейменувати зображення (ми не збираємося робити це тут, але ось як це зробити), спочатку зупиніть контейнер:

```
lxc stop rockylinux-test-8
```

Використовуйте команду `move`, щоб змінити назву контейнера:

```
lxc move rockylinux-test-8 rockylinux-8
```

Якщо ви все одно виконали цю інструкцію, зупиніть контейнер і поверніть його до вихідної назви, щоб продовжувати слідувати.

Для цілей цього посібника встановіть ще два зображення:

```
lxc launch images:rockylinux/9 rockylinux-test-9
```

та

```
lxc launch images:ubuntu/22.04 ubuntu-test
```

Перевірте, що у вас є, перерахувавши свої зображення:

```
lxc list
```

який поверне це:

```
+-------------------+---------+----------------------+------+-----------
+-----------+
|       NAME        |  STATE  |         IPV4         | IPV6 |   TYPE    |
SNAPSHOTS |
+-------------------+---------+----------------------+------+-----------
+-----------+
| rockylinux-test-8 | RUNNING | 10.146.84.179 (eth0) |      | CONTAINER |
0         |
+-------------------+---------+----------------------+------+-----------
+-----------+
| rockylinux-test-9 | RUNNING | 10.146.84.180 (eth0) |      | CONTAINER |
0         |
+-------------------+---------+----------------------+------+-----------
+-----------+
| ubuntu-test       | RUNNING | 10.146.84.181 (eth0) |      | CONTAINER |
0         |
+-------------------+---------+----------------------+------+-----------
+-----------+
```

8. Розділ 6: Профілі

У цій главі вам потрібно буде виконувати команди як непривілейований користувач ("lxdadmin", якщо ви слідкували за цим із самого початку цієї книги).

Під час встановлення LXD ви отримуєте профіль за замовчуванням, і цей профіль не можна видалити чи змінити. Тим не менш, ви можете використовувати профіль за замовчуванням для створення нових профілів для використання з вашими контейнерами.

Якщо ви перевірите свій список контейнерів, ви помітите, що IP-адреса в кожному випадку походить з мостового інтерфейсу. У виробничому середовищі ви можете використовувати щось інше. Це може бути адреса, призначена DHCP з вашого інтерфейсу LAN, або навіть статична адреса з вашої WAN.

Якщо ви налаштуєте свій сервер LXD з двома інтерфейсами та призначите кожному IP-адресу в глобальній і локальній мережах, можна призначити IP-адреси вашого контейнера на основі інтерфейсу, до якого контейнер має бути спрямований.

Починаючи з версії 9.0 Rocky Linux (і фактично будь-якої помилки для копії помилок Red Hat Enterprise Linux) метод статичного або динамічного призначення IP-адрес із профілями не працює.

Є способи обійти це, але це дратує. Схоже, це якось пов’язано зі змінами, внесеними в Network Manager, які впливають на macvlan. macvlan дозволяє створювати багато інтерфейсів з різними адресами рівня 2.

Наразі майте на увазі, що це має недоліки під час вибору зображень контейнерів на основі RHEL.

8.1 Створення профілю macvlan і його призначення

Щоб створити наш профіль macvlan, скористайтеся цією командою:

```
lxc profile create macvlan
```

Якщо ви користуєтеся комп’ютером із кількома інтерфейсами та бажаєте мати більше одного шаблону macvlan на основі мережі, яку ви хочете охопити, ви можете використати «lanmacvlan» або «wanmacvlan» або будь-яке інше ім’я, яке ви бажаєте використати для ідентифікації профілю. Використовувати «macvlan» у заяві про створення профілю повністю залежить від вас.

Ви хочете змінити інтерфейс macvlan, але перш ніж це зробити, вам потрібно знати, що таке батьківський інтерфейс для нашого сервера LXD. Це буде інтерфейс, якому призначено IP-адресу локальної мережі (у цьому випадку). Щоб дізнатися, що це за інтерфейс, використовуйте:

```
ip addr
```

Шукайте інтерфейс із призначенням IP LAN у мережі 192.168.1.0/24:

```
2: enp3s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP
group default qlen 1000
    link/ether 40:16:7e:a9:94:85 brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.106/24 brd 192.168.1.255 scope global dynamic noprefixroute
enp3s0
       valid_lft 4040sec preferred_lft 4040sec
    inet6 fe80::a308:acfb:fcb3:878f/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
```

В цьому випадку інтерфейс буде "enp3s0".

Тепер давайте змінимо профіль:

```
lxc profile device add macvlan eth0 nic nictype=macvlan parent=enp3s0
```

Ця команда додає всі необхідні параметри до профілю macvlan, необхідні для використання.

Щоб побачити, що створила ця команда, скористайтеся командою:

```
lxc profile show macvlan
```

Що дасть вам результат, подібний до цього:

```
config: {}
description: ""
devices:
  eth0:
    nictype: macvlan
    parent: enp3s0
    type: nic
name: macvlan
used_by: []
```

Ви можете використовувати профілі для багатьох інших речей, але призначення статичної IP-адреси контейнеру або використання власного DHCP-сервера є звичайними потребами.

Щоб призначити профіль macvlan для rockylinux-test-8, нам потрібно зробити наступне:

```
lxc profile assign rockylinux-test-8 default,macvlan
```

Зробимо те саме для rockylinux-test-9:

```
lxc profile assign rockylinux-test-9 default,macvlan
```

Тут сказано, що вам потрібен профіль за замовчуванням, а також застосувати профіль macvlan.

8.2 Rocky Linux macvlan

У дистрибутивах і клонах RHEL Network Manager постійно змінювався. На початку Rocky Linux 8 профіль macvlan був зламаний.

Майте на увазі, що нічого з цього не має нічого спільного з Rocky Linux, а лише з реалізацією пакета вище.

Якщо ви хочете запускати контейнери Rocky Linux і використовувати macvlan для призначення IP-адреси з вашої мережі LAN або WAN, процес залежить від версії контейнера операційної системи (8.x або 9.x).

8.2.1 Rocky Linux 9.0 macvlan – виправлення DHCP

Спочатку давайте проілюструємо, що відбувається під час зупинки та перезапуску двох контейнерів після призначення профілю macvlan.

Однак призначення профілю не змінює конфігурацію за замовчуванням, якою за замовчуванням є DHCP.

Щоб перевірити це, просто виконайте такі дії:

```
lxc restart rocky-test-8
lxc restart rocky-test-9
```

Перелічіть свої контейнери ще раз і зауважте, що rockylinux-test-9 більше не має IP-адреси:

```
lxc list
```

```
+-------------------+---------+----------------------+------+-----------
+-----------+
|       NAME        |  STATE  |         IPV4         | IPV6 |   TYPE    |
SNAPSHOTS |
+-------------------+---------+----------------------+------+-----------
+-----------+
| rockylinux-test-8 | RUNNING | 192.168.1.114 (eth0) |      | CONTAINER |
0         |
+-------------------+---------+----------------------+------+-----------
+-----------+
| rockylinux-test-9 | RUNNING |                      |      | CONTAINER |
0         |
+-------------------+---------+----------------------+------+-----------
+-----------+
| ubuntu-test       | RUNNING | 10.146.84.181 (eth0) |      | CONTAINER |
0         |
+-------------------+---------+----------------------+------+-----------
+-----------+
```

Як бачите, наш контейнер Rocky Linux 8.x отримав IP-адресу від інтерфейсу LAN, тоді як контейнер Rocky Linux 9.x ні.

Щоб детальніше продемонструвати проблему, вам потрібно запустити `dhclient` у контейнері Rocky Linux 9.0. Це покаже нам, що профіль macvlan *є* фактично застосованим:

```
lxc exec rockylinux-test-9 dhclient
```

Інший список контейнерів тепер показує наступне:

```
+-------------------+---------+----------------------+------+-----------
+-----------+
|       NAME        |  STATE  |         IPV4         | IPV6 |   TYPE    |
SNAPSHOTS |
+-------------------+---------+----------------------+------+-----------
+-----------+
| rockylinux-test-8 | RUNNING | 192.168.1.114 (eth0) |      | CONTAINER |
0         |
+-------------------+---------+----------------------+------+-----------
+-----------+
| rockylinux-test-9 | RUNNING | 192.168.1.113 (eth0) |      | CONTAINER |
0         |
+-------------------+---------+----------------------+------+-----------
+-----------+
| ubuntu-test       | RUNNING | 10.146.84.181 (eth0) |      | CONTAINER |
0         |
+-------------------+---------+----------------------+------+-----------
+-----------+
```

Це мало статися з простою зупинкою та запуском контейнера, але цього не відбувається. Припускаючи, що ви хочете щоразу використовувати призначену DHCP IP-адресу, ви можете виправити це за допомогою простого запису crontab. Для цього нам потрібно отримати доступ оболонки до контейнера, ввівши:

```
lxc exec rockylinux-test-9 bash
```

Далі визначимо повний шлях до `dhclient`. Для цього, оскільки цей контейнер створено з мінімального зображення, вам потрібно спочатку встановити `which`:

```
dnf install which
```

потім запустіть:

```
which dhclient
```

який має повернути:

```
/usr/sbin/dhclient
```

Далі змінимо crontab користувача root:

```
crontab -e
```

І додайте цей рядок:

```
@reboot    /usr/sbin/dhclient
```

Введена команда crontab використовує *vi*. Щоб зберегти зміни та вийти, використовуйте `SHIFT`+`:`+`wq`.

Вийдіть із контейнера та перезапустіть rockylinux-test-9:

```
lxc restart rockylinux-test-9
```

Інший список показує, що контейнеру призначено адресу DHCP:

```
+-------------------+---------+----------------------+------+-----------
+-----------+
|       NAME        |  STATE  |         IPV4         | IPV6 |   TYPE    |
SNAPSHOTS |
+-------------------+---------+----------------------+------+-----------
+-----------+
| rockylinux-test-8 | RUNNING | 192.168.1.114 (eth0) |      | CONTAINER |
0         |
+-------------------+---------+----------------------+------+-----------
+-----------+
| rockylinux-test-9 | RUNNING | 192.168.1.113 (eth0) |      | CONTAINER |
0         |
+-------------------+---------+----------------------+------+-----------
+-----------+
| ubuntu-test       | RUNNING | 10.146.84.181 (eth0) |      | CONTAINER |
0         |
```

```
+-------------------+---------+----------------------+------+-----------
+-----------+
```

8.2.2 Rocky Linux 9.0 macvlan – виправлення статичної IP-адреси

Для статичного призначення IP-адреси все стає ще заплутанішим. Оскільки `network-scripts` тепер застаріло в Rocky Linux 9.x, єдиний спосіб зробити це – через статичне призначення, і через те, як контейнери використовують мережу, ви не зможете щоб встановити маршрут за допомогою звичайного оператора `ip route`. Проблема полягає в тому, що інтерфейс, призначений під час застосування профілю macvlan (у цьому випадку eth0), не можна керувати за допомогою Network Manager. Виправлення полягає в перейменуванні мережевого інтерфейсу контейнера після перезапуску та призначенні статичної IP-адреси. Ви можете зробити це за допомогою сценарію та запустити (ще раз) у crontab root. Зробіть це за допомогою команди `ip`.

Для цього вам потрібно знову отримати доступ оболонки до контейнера:

```
lxc exec rockylinux-test-9 bash
```

Далі ми створимо сценарій bash у `/usr/local/sbin` під назвою «static»:

```
vi /usr/local/sbin/static
```

Вміст цього сценарію простий:

```
#!/usr/bin/env bash

/usr/sbin/ip link set dev eth0 name net0
/usr/sbin/ip addr add 192.168.1.151/24 dev net0
/usr/sbin/ip link set dev net0 up
/usr/sbin/ip route add default via 192.168.1.1
```

Що ми тут робимо?

- ви перейменовуєте eth0 на нове ім'я, яким ми можемо керувати ("net0")
- ви призначаєте новий статичний IP, який ми виділили для нашого контейнера (192.168.1.151)
- ви відкриваєте новий інтерфейс "net0"
- вам потрібно додати маршрут за замовчуванням для нашого інтерфейсу

Зробіть наш сценарій виконуваним за допомогою:

```
chmod +x /usr/local/sbin/static
```

Додайте це до root-файлу crontab для контейнера з часом @reboot:

```
@reboot     /usr/local/sbin/static
```

Нарешті, вийдіть з контейнера та перезапустіть його:

```
lxc restart rockylinux-test-9
```

Зачекайте кілька секунд і знову виведіть список контейнерів:

```
lxc list
```

Ви повинні побачити успіх:

```
+-------------------+---------+----------------------+------+-----------
+-----------+
|       NAME        |  STATE  |         IPV4         | IPV6 |   TYPE    |
SNAPSHOTS |
+-------------------+---------+----------------------+------+-----------
+-----------+
| rockylinux-test-8 | RUNNING | 192.168.1.114 (eth0) |      | CONTAINER |
0         |
+-------------------+---------+----------------------+------+-----------
+-----------+
| rockylinux-test-9 | RUNNING | 192.168.1.151 (eth0) |      | CONTAINER |
0         |
+-------------------+---------+----------------------+------+-----------
+-----------+
```

```
| ubuntu-test       | RUNNING | 10.146.84.181 (eth0) |      | CONTAINER |
0         |
+-------------------+---------+----------------------+------+-----------
+-----------+
```

8.3 Ubuntu macvlan

На щастя, у реалізації Network Manager в Ubuntu стек macvlan НЕ зламано. Це набагато простіше розгорнути!

Як і у випадку з вашим контейнером rockylinux-test-9, вам потрібно призначити профіль нашому контейнеру:

```
lxc profile assign ubuntu-test default,macvlan
```

Щоб дізнатися, чи призначає DHCP адресу контейнеру, зупиніть і знову запустіть контейнер:

```
lxc restart ubuntu-test
```

Потім знову перерахуйте контейнери:

```
+-------------------+---------+----------------------+------+-----------
+-----------+
|       NAME        |  STATE  |         IPV4         | IPV6 |   TYPE    |
SNAPSHOTS |
+-------------------+---------+----------------------+------+-----------
+-----------+
| rockylinux-test-8 | RUNNING | 192.168.1.114 (eth0) |      | CONTAINER |
0         |
+-------------------+---------+----------------------+------+-----------
+-----------+
| rockylinux-test-9 | RUNNING | 192.168.1.151 (eth0) |      | CONTAINER |
0         |
+-------------------+---------+----------------------+------+-----------
+-----------+
| ubuntu-test       | RUNNING | 192.168.1.132 (eth0) |      | CONTAINER |
0         |
+-------------------+---------+----------------------+------+-----------
+-----------+
```

Успішно!

Налаштування статичної IP-адреси трохи відрізняється, але зовсім не складно. Вам потрібно змінити файл .yaml, пов’язаний із з’єднанням контейнера (`10-lxc.yaml`). Для цього статичного IP ви будете використовувати 192.168.1.201:

```
vi /etc/netplan/10-lxc.yaml
```

Змініть те, що є, на таке:

```
network:
  version: 2
  ethernets:
    eth0:
      dhcp4: false
      addresses: [192.168.1.201/24]
      gateway4: 192.168.1.1
      nameservers:
        addresses: [8.8.8.8,8.8.4.4]
```

Збережіть зміни та вийдіть з контейнера.

Перезапустіть контейнер:

```
lxc restart ubuntu-test
```

Коли ви знову перерахуєте свої контейнери, ви побачите нашу нову статичну IP-адресу:

```
+-------------------+---------+----------------------+------+-----------
+-----------+
|       NAME        |  STATE  |         IPV4         | IPV6 |   TYPE    |
SNAPSHOTS |
+-------------------+---------+----------------------+------+-----------
+-----------+
| rockylinux-test-8 | RUNNING | 192.168.1.114 (eth0) |      | CONTAINER |
0         |
+-------------------+---------+----------------------+------+-----------
+-----------+
| rockylinux-test-9 | RUNNING | 192.168.1.151 (eth0) |      | CONTAINER |
0         |
+-------------------+---------+----------------------+------+-----------
+-----------+
| ubuntu-test       | RUNNING | 192.168.1.201 (eth0) |      | CONTAINER |
0         |
```

```
+-------------------+---------+----------------------+------+-----------
+-----------+
```

Успішно!

У прикладах, використаних у цій главі, було навмисно обрано жорсткий контейнер для налаштування та два менш складних. У списку зображень доступно багато інших версій Linux. Якщо у вас є улюблений, спробуйте встановити його, призначити шаблон macvlan і налаштувати IP-адреси.

9. Розділ 7: Параметри конфігурації контейнера

У цій главі вам потрібно буде виконувати команди як непривілейований користувач ("lxdadmin", якщо ви слідкували за цим із самого початку цієї книги).

Існує велика кількість варіантів налаштування контейнера після його встановлення. Перш ніж їх побачити, давайте розглянемо команду `info` для контейнера. У цьому прикладі ви будете використовувати контейнер ubuntu-test:

```
lxc info ubuntu-test
```

Це показує щось на зразок наступного:

```
Name: ubuntu-test
Location: none
Remote: unix://
Architecture: x86_64
Created: 2021/04/26 15:14 UTC
Status: Running
Type: container
Profiles: default, macvlan
Pid: 584710
Ips:
  eth0:    inet    192.168.1.201    enp3s0
  eth0:    inet6    fe80::216:3eff:fe10:6d6d    enp3s0
  lo:    inet    127.0.0.1
  lo:    inet6    ::1
Resources:
  Processes: 13
  Disk usage:
    root: 85.30MB
  CPU usage:
    CPU usage (in seconds): 1
  Memory usage:
    Memory (current): 99.16MB
    Memory (peak): 110.90MB
  Network usage:
    eth0:
      Bytes received: 53.56kB
      Bytes sent: 2.66kB
      Packets received: 876
```

```
      Packets sent: 36
    lo:
      Bytes received: 0B
      Bytes sent: 0B
      Packets received: 0
      Packets sent: 0
```

Там є багато корисної інформації, від застосованих профілів до використовуваної пам’яті, місця на диску тощо.

9.0.1 Кілька слів про конфігурацію та деякі параметри

За замовчуванням LXD виділить для контейнера необхідну системну пам’ять, дисковий простір, ядра ЦП тощо. Але що, якщо ми хочемо бути більш конкретними? Це цілком можливо.

Однак для цього є компроміси. Наприклад, якщо ви призначаєте системну пам’ять, а контейнер не використовує її всю, ви зберегли її від іншого контейнера, якому вона насправді може знадобитися. Однак може статися й зворотне. Якщо контейнер займає багато пам’яті, він може перешкоджати іншим контейнерам отримати достатньо, тим самим знижуючи їх продуктивність.

Просто майте на увазі, що кожна ваша дія для налаштування контейнера *може* мати негативні наслідки в іншому місці.

Замість того, щоб переглядати всі параметри конфігурації, скористайтеся автозаповненням вкладки, щоб побачити доступні параметри:

```
lxc config set ubuntu-test
```

і TAB.

Це покаже вам усі параметри для налаштування контейнера. Якщо у вас виникли запитання щодо роботи одного з параметрів конфігурації, перейдіть до офіційної документації для LXD і виконайте пошук за параметром конфігурації або весь рядок у Google, наприклад `lxc config set limits.memory`, і подивіться на результати пошуку.

Ми розглянемо кілька найбільш використовуваних параметрів конфігурації. Наприклад, якщо ви хочете встановити максимальний обсяг пам’яті, який може використовувати контейнер:

```
lxc config set ubuntu-test limits.memory 2GB
```

Це говорить про те, що поки пам’ять доступна для використання, іншими словами, є 2 Гб вільної пам’яті, тоді контейнер фактично може використовувати більше 2 Гб якщо вона доступна. Іншими словами, це м’яке обмеження.

```
lxc config set ubuntu-test limits.memory.enforce 2GB
```

Це говорить про те, що контейнер ніколи не може використовувати більше 2 Гб пам’яті, незалежно від того, доступна вона зараз чи ні. У цьому випадку це жорстке обмеження.

```
lxc config set ubuntu-test limits.cpu 2
```

Це говорить про обмеження кількості ядер cpu, які може використовувати контейнер, до 2.

Примітка

Коли цей документ було переписано для Rocky Linux 9.0, репозиторій ZFS для 9 був недоступний. З цієї причини всі наші тестові контейнери були створені з використанням «dir» у процесі ініціалізації. Ось чому в наведеному нижче прикладі показано пул зберігання «dir» замість «zfs».

Пам’ятаєте, коли ми налаштовували наш пул сховищ у розділі про ZFS? Ми назвали пул «сховище», але ми могли назвати його як завгодно. Якщо ми хочемо поглянути на це, ми можемо використати цю команду, яка однаково добре працює для будь-якого іншого типу пулу (як показано для dir):

```
lxc storage show storage
```

Це показує наступне:

```
config:
  source: /var/snap/lxd/common/lxd/storage-pools/storage
```

```
description: ""
name: storage
driver: dir
used_by:
- /1.0/instances/rockylinux-test-8
- /1.0/instances/rockylinux-test-9
- /1.0/instances/ubuntu-test
- /1.0/profiles/default
status: Created
locations:
- none
```

Це показує, що всі наші контейнери використовують наш пул сховищ dir. Під час використання ZFS ви також можете встановити дискову квоту для контейнера. Ось як це виглядатиме встановлення дискової квоти розміром 2 Гб для контейнера ubuntu-test.

```
lxc config device override ubuntu-test root size=2GB
```

Як було сказано раніше, ви повинні економно використовувати параметри конфігурації, якщо у вас немає контейнера, який хоче використовувати значно більше, ніж його частка ресурсів. LXD, здебільшого, добре керуватиме середовищем самостійно.

Є, звичайно, ще багато варіантів, які можуть зацікавити деяких людей. Ви повинні провести власне дослідження, щоб з’ясувати, чи є щось із цього цінним у вашому середовищі.

10. Розділ 8: контейнер Snapshots

У цій главі вам потрібно буде виконувати команди як непривілейований користувач ("lxdadmin", якщо ви слідкуєте за цим з початку цієї книги).

Снепшот контейнери разом із снепшот сервером (про який ми розповімо пізніше) є, мабуть, найважливішим аспектом роботи робочого сервера LXD. Снепшоти забезпечують швидке відновлення. Доцільно використовувати їх як захист від збоїв під час оновлення основного програмного забезпечення, яке працює на певному контейнері. Якщо під час оновлення щось трапиться, що призведе до поломки цієї програми, ви просто відновите знімок, і ви відновите роботу з лише кількома секундами простою.

Автор використав контейнери LXD для загальнодоступних серверів PowerDNS, і процес оновлення цих програм став набагато безтурботнішим, оскільки ви можете спочатку зробити снепшот контейнера, перш ніж продовжити.

Ви навіть можете зробити снепшот контейнера під час його роботи.

10.1 Процес снепшот

Ми почнемо з отримання снепшот контейнера ubuntu-test за допомогою цієї команди:

```
lxc snapshot ubuntu-test ubuntu-test-1
```

Тут ми називаємо снепшот «ubuntu-test-1», але його можна називати як завгодно. Щоб переконатися, що у вас є снепшот, виконайте «lxc info» контейнера:

```
lxc info ubuntu-test
```

Ви вже дивилися на інформаційний екран. Якщо прокрутити вниз, ви побачите:

```
Snapshots:
  ubuntu-test-1 (taken at 2021/04/29 15:57 UTC) (stateless)
```

Успішно! Наш снепшот на місці.

Тепер перейдіть до контейнера ubuntu-test:

```
lxc exec ubuntu-test bash
```

І створіть порожній файл за допомогою команди *touch*:

```
touch this_file.txt
```

Тепер вийдіть з контейнера.

Перш ніж ми відновимо контейнер, як він був до створення файлу, найбезпечніший спосіб відновити контейнер, особливо якщо було багато змін, це спочатку зупинити його:

```
lxc stop ubuntu-test
```

Потім відновіть його:

```
lxc restore ubuntu-test ubuntu-test-1
```

Потім знову запустіть контейнер:

```
lxc start ubuntu-test
```

Якщо ви знову повернетесь у контейнер і подивитесь, наш «this_file.txt», який ми створили, тепер зник.

Якщо снепшот більше не потрібен, його можна видалити:

```
lxc delete ubuntu-test/ubuntu-test-1
```

Попередження

Ви завжди повинні видаляти снепшоти, коли контейнер запущено. Чому? Команда *lxc delete* також працює для видалення всього контейнера. Якби ми випадково натиснули Enter після «ubuntu-test» у наведеній вище команді, І якщо контейнер було зупинено, контейнер було б видалено. Попередження не надається, він просто виконує те, що ви просите.

Однак якщо контейнер запущено, ви отримаєте таке повідомлення:

```
Error: The instance is currently running, stop it first or pass --force
```

Тому завжди видаляйте снепшоти під час роботи контейнера.

У наступних розділах ви:

- налаштуєте процес автоматичного створення snapshots
- налаштуєте термін дії снепшота, щоб він зникав через певний проміжок часу
- налаштуєте автоматичне оновлення снепшотів на снепшот сервері

11. Глава 9: Сервер Snapshot

У цій главі використовується комбінація привілейованого (root) користувача та непривілейованого (lxdadmin) користувача залежно від завдань, які ми виконуємо.

Як зазначалося на початку, snapshot сервер для LXD має бути дзеркалом робочого сервера всіма можливими способами. Причина полягає в тому, що вам може знадобитися перенести його на робочий стан у разі апаратної несправності, а наявність не лише резервних копій, але й швидкого способу запуску робочих контейнерів зводить до мінімуму панічні телефонні дзвінки та текстові повідомлення системних адміністраторів. ЦЕ ЗАВЖДИ ДОБРЕ!

Таким чином, процес створення snapshot сервера точно схожий на робочий сервер. Щоб повністю імітувати налаштування нашого робочого сервера, виконайте всі **розділи 1-4** ще раз на snapshot сервері, а коли завершите, поверніться до цього місця.

Ви повернулись!! Вітаємо, це має означати, що ви успішно завершили базову інсталяцію snapshot сервера.

11.1 Налаштування зв’язку між основним і snapshot сервером

Перш ніж продовжити, потрібно трохи прибрати. По-перше, якщо ви працюєте у робочому середовищі, ви, ймовірно, маєте доступ до DNS-сервера, який можна використовувати для налаштування IP-адреси для розпізнавання імен.

У нашій лабораторії ми не маємо такої розкоші. Можливо, у вас такий самий сценарій. З цієї причини ви збираєтеся додати IP-адреси та імена серверів до файлу /etc/hosts на основному сервері та снепшот сервері. Вам потрібно буде зробити це як ваш root (або *sudo*) користувач.

У нашій лабораторії основний сервер LXD працює на 192.168.1.106, а сервер знімків LXD працює на 192.168.1.141. SSH на кожному сервері та додайте наступне до файлу /etc/hosts:

```
192.168.1.106 lxd-primary
192.168.1.141 lxd-snapshot
```

Далі вам потрібно дозволити весь трафік між двома серверами. Для цього ви збираєтеся змінити правила `firewalld`. Спочатку на сервері lxd-primary додайте цей рядок:

```
firewall-cmd zone=trusted add-source=192.168.1.141 --permanent
```

а на сервері знімків додайте це правило:

```
firewall-cmd zone=trusted add-source=192.168.1.106 --permanent
```

потім перезавантажте:

```
firewall-cmd reload
```

Далі, як наш непривілейований користувач (lxdadmin), вам потрібно встановити довірчі відносини між двома машинами. Це робиться за допомогою наступного виконання на lxd-primary:

```
lxc remote add lxd-snapshot
```

Це відображає сертифікат, який потрібно прийняти. Прийміть його, і з’явиться запит на введення пароля. Це «пароль довіри», який ви встановлюєте під час виконання кроку ініціалізації LXD. Сподіваємось, ви надійно зберігаєте всі ці паролі. Коли ви введете пароль, ви отримаєте це:

```
Client certificate stored at server:  lxd-snapshot
```

Не завадить мати це і в зворотному порядку. Наприклад, також установіть довірчі відносини на сервері lxd-snapshot. Таким чином, за потреби, сервер lxd-snapshot може надсилати знімки назад на основний сервер lxd-primary. Повторіть кроки та замініть у «lxd-primary» на «lxd-snapshot».

11.1.1 Перенесення вашого першого snapshot

Перш ніж ви зможете перенести свій перший snapshot, вам потрібно створити будь-які профілі на lxd-snapshot, які ви створили на lxd-primary. У нашому випадку це профіль "macvlan".

Вам потрібно буде створити це для lxd-snapshot. Поверніться до розділу 6 і створіть профіль "macvlan" у lxd-snapshot, якщо потрібно. Якщо ваші два сервери мають однакові назви батьківського інтерфейсу (наприклад, "enp3s0"), ви можете скопіювати профіль "macvlan" до lxd-snapshot без його повторного створення:

```
lxc profile copy macvlan lxd-snapshot
```

Коли всі зв’язки та профілі налаштовано, наступним кроком є фактичне надсилання знімка з lxd-primary до lxd-snapshot. Якщо ви точно слідкували за цим, ви, ймовірно, видалили всі свої знімки. Створіть інший знімок:

```
lxc snapshot rockylinux-test-9 rockylinux-test-9-snap1
```

Якщо ви запустите команду «info» для `lxc`, ви побачите знімок у нижній частині нашого списку:

```
lxc info rockylinux-test-9
```

Унизу буде показано щось на зразок цього:

```
rockylinux-test-9-snap1 at 2021/05/13 16:34 UTC) (stateless)
```

Тож, тримаємо пальці! Давайте спробуємо перенести наш snapshot:

```
lxc copy rockylinux-test-9/rockylinux-test-9-snap1 lxd-snapshot:rockylinux-
test-9
```

Ця команда говорить, що всередині контейнера rockylinux-test-9 ви хочете надіслати snapshot rockylinux-test-9-snap1 до lxd-snapshot і назвати його rockylinux-test-9.

Через короткий час копіювання буде завершено. Хочете дізнатися напевно? Створіть `список lxc` на сервері lxd-snapshot. Що має повернути наступне:

```
+-------------------+---------+------+------+-----------+-----------+
|    NAME           |  STATE  | IPV4 | IPV6 |   TYPE    | SNAPSHOTS |
+-------------------+---------+------+------+-----------+-----------+
| rockylinux-test-9 | STOPPED |      |      | CONTAINER | 0         |
+-------------------+---------+------+------+-----------+-----------+
```

Успішно! Спробуйте запустити його. Оскільки ми запускаємо його на сервері lxd-snapshot, вам потрібно спершу зупинити його на сервері lxd-primary, щоб уникнути конфлікту IP-адрес:

```
lxc stop rockylinux-test-9
```

А на сервері lxd-snapshot:

```
lxc start rockylinux-test-9
```

Якщо припустити, що все це працює без помилок, зупиніть контейнер на lxd-snapshot і запустіть його знову на lxd-primary.

11.2 Вимкнути boot.autostart для контейнерів

Snapshots, скопійовані в lxd-snapshot, не працюватимуть під час міграції, але якщо у вас виникне подія живлення або потрібно перезавантажити сервер знімків через оновлення чи щось інше, у вас виникне проблема. Ці контейнери намагатимуться запуститися на сервері знімків, створюючи потенційний конфлікт IP-адрес.

Щоб усунути це, потрібно налаштувати переміщені контейнери так, щоб вони не запускалися після перезавантаження сервера. Для нашого щойно скопійованого контейнера rockylinux-test-9 ви зробите це за допомогою:

```
lxc config set rockylinux-test-9 boot.autostart 0
```

Зробіть це для кожного знімка на сервері lxd-snapshot. «0» у команді переконається, що `boot.autostart` вимкнено.

11.3 Автоматизація snapshot процесу

Чудово, що ви можете створювати миттєві snapshots, коли вам це потрібно, і іноді вам *потрібно* створити миттєвий snapshot вручну. Ви навіть можете вручну скопіювати його в lxd-snapshot. Але в інших випадках, особливо для багатьох контейнерів, які працюють на вашому первинному сервері lxd, **останнє**, що ви хочете зробити, це провести півдня, видаляючи snapshots на snapshot сервері знімків, створюючи нові snapshots та надсилаючи їх на snapshot сервер. Для основної частини ваших операцій ви захочете автоматизувати процес.

Перше, що вам потрібно зробити, це запланувати процес автоматизованого створення снепшоту на lxd-primary. Ви зробите це для кожного контейнера на сервері lxd-primary. Після завершення він подбає про це в майбутньому. Це можна зробити за допомогою наступного синтаксису. Зверніть увагу на схожість із записом crontab для позначки часу:

```
lxc config set [container_name] snapshots.schedule "50 20 * * *"
```

Це означає, що щодня о 20:50 робіть знімок назви контейнера.

Щоб застосувати це до нашого контейнера rockylinux-test-9:

```
lxc config set rockylinux-test-9 snapshots.schedule "50 20 * * *"
```

Ви також хочете налаштувати назву знімка, щоб мати значення до нашої дати. LXD скрізь використовує UTC, тому наш найкращий вибір, щоб відстежувати речі, це встановити ім’я знімка з датою та міткою часу в більш зрозумілому форматі:

```
lxc config set rockylinux-test-9 snapshots.pattern "rockylinux-
test-9{{ creation_date|date:'2006-01-02_15-04-05' }}"
```

ЧУДОВО, але ви точно не хочете отримувати новий знімок щодня, не позбувшись старого, чи не так? Ви б заповнили диск знімками. Щоб виправити це, виконайте:

```
lxc config set rockylinux-test-9 snapshots.expiry 1d
```

12. Розділ 10: автоматизація snapshots

У цьому розділі вам потрібно мати права root або вміти використовувати `sudo`, щоб стати root.

Автоматизація процесу створення знімків значно полегшує роботу.

12.1 Автоматизація snapshot процесу

Виконайте цей процес на lxd-primary. Перше, що вам потрібно зробити, це створити сценарій, який запускатиметься cron у /usr/local/sbin під назвою "refresh-containers":

```
sudo vi /usr/local/sbin/refreshcontainers.sh
```

Сценарій досить мінімальний:

```
#!/bin/bash
# This script is for doing an lxc copy --refresh against each container,
copying
# and updating them to the snapshot server.

for x in $(/var/lib/snapd/snap/bin/lxc ls -c n --format csv)
        do echo "Refreshing $x"
        /var/lib/snapd/snap/bin/lxc copy --refresh $x lxd-snapshot:$x
        done
```

Зробіть його виконуваним:

```
sudo chmod +x /usr/local/sbin/refreshcontainers.sh
```

Змініть право власності на цей сценарій на свого користувача та групу lxdadmin:

```
sudo chown lxdadmin.lxdadmin /usr/local/sbin/refreshcontainers.sh
```

Налаштуйте crontab для користувача lxdadmin, щоб запустити цей сценарій, у цьому випадку о 22:00:

```
crontab -e
```

Ваш запис виглядатиме так:

```
00 22 * * * /usr/local/sbin/refreshcontainers.sh > /home/lxdadmin/refreshlog
2>&1
```

Збережіть зміни та вийдіть.

Це створить журнал у домашньому каталозі lxdadmin під назвою «refreshlog», який дасть вам інформацію про те, чи працював ваш процес, чи ні. Дуже важливо!

Автоматизована процедура іноді дає збій. Зазвичай це трапляється, коли не вдається оновити певний контейнер. Ви можете вручну повторно запустити оновлення за допомогою такої команди (припускаючи, що rockylinux-test-9 тут є нашим контейнером):

```
lxc copy --refresh rockylinux-test-9 lxd-snapshot:rockylinux-test-9
```

13. Додаток A - налаштування робочої станції

Хоча ця процедура не є частиною розділів про сервер LXD, ця процедура допоможе тим, хто хоче лабораторне середовище, напівпостійну ОС і програму, що працює на робочій станції або ноутбуці Rocky Linux.

13.1 Передумови

- вам зручно працювати в командному рядку
- ви вільно використовуєте редактор командного рядка, наприклад `vi` або `nano`
- готові встановити `snapd` для встановлення LXD
- потреба в стабільному середовищі тестування, яке використовується щодня або за потреби
- можете стати root або мати привілеї `sudo`

13.2 Інсталяція

З командного рядка встановіть репозиторій EPEL:

```
sudo dnf install epel-release
```

Коли встановлення завершиться, виконайте оновлення:

```
sudo dnf upgrade
```

Встановіть `snapd`

```
sudo dnf install snapd
```

Увімкніть службу `snapd`

```
sudo systemctl enable snapd
```

Перезавантажте ноутбук або робочу станцію

Встановіть оснащення для LXD:

```
sudo snap install lxd
```

13.3 Ініціалізація LXD

Якщо ви переглянули розділи про робочий сервер, це майже те саме, що процедура ініціалізації робочого сервера.

```
sudo lxd init
```

Відкриється діалогове вікно запитань і відповідей.

Ось запитання та наші відповіді щодо сценарію з невеликими поясненнями, де це необхідно:

```
Бажаєте використовувати кластеризацію LXD? (yes/no) [default=no]:
```

Якщо вас цікавить кластеризація, проведіть додаткові дослідження щодо цього у контейнерах Linux тут.

```
Бажаєте налаштувати новий пул зберігання? (yes/no) [default=yes]:
Name of the new storage pool [default=default]: storage
```

За бажанням ви можете прийняти значення default.

```
Ім’я сховища для використання (btrfs, dir, lvm, ceph) [default=btrfs]: dir
```

Зауважте, що `dir` дещо повільніший, ніж `btrfs`. Якщо у вас є передбачливість залишити диск порожнім, ви можете використовувати цей пристрій (приклад: /dev/sdb) для пристрою `btrfs`, а потім вибрати `btrfs`, але лише якщо ваш хост-комп’ютер має операційну систему, яка підтримує `btrfs`. Rocky Linux і будь-які клони RHEL не підтримуватимуть `btrfs` — у всякому разі, поки ні. `dir` добре працюватиме в лабораторному середовищі.

```
Бажаєте підключитися до сервера MAAS? (yes/no) [default=no]:
```

Metal As A Service (MAAS) виходить за рамки цього документа.

```
Бажаєте створити новий міст локальної мережі? (yes/no) [default=yes]:
Як мав би називатися новий міст? [default=lxdbr0]:
Яку адресу IPv4 слід використовувати? (CIDR subnet notation, “auto” or “none”)
[default=auto]:
Яку адресу IPv6 слід використовувати? (CIDR subnet notation, “auto” or “none”)
[default=auto]: none
```

Якщо ви хочете використовувати IPv6 у своїх контейнерах LXD, ви можете ввімкнути цю опцію. Це залежить від вас.

```
Бажаєте, щоб сервер LXD був доступний через мережу? (yes/no) [default=no]: yes
```

Це необхідно для створення snapshot робочої станції. Тут дайте відповідь "yes".

```
Address to bind LXD to (not including port) [default=all]:
Port to bind LXD to [default=8443]:
Trust password for new clients:
Again:
```

Цей пароль довіри означає, як ви підключатиметеся до snapshot сервера або повертатиметеся із snapshot сервера. Встановіть це з тим, що має сенс у вашому оточенні. Збережіть цей запис у безпечному місці, наприклад у менеджері паролів.

```
Бажаєте, щоб застарілі кешовані зображення оновлювалися автоматично? (yes/no)
[default=yes]
Чи бажаєте ви, щоб надруковано передзапуск YAML "lxd init"? (yes/no)
[default=no]:
```

13.4 Права користувача

Наступне, що вам потрібно зробити, це додати свого користувача до групи lxd. Знову ж таки, для цього вам потрібно буде використовувати `sudo` або бути адміністратором:

```
sudo usermod -a -G lxd [username]
```

де [username] — ваш користувач у системі.

На цьому етапі ви внесли багато змін. Перш ніж йти далі, перезавантажте машину.

13.5 Перевірка інсталяції

Щоб переконатися, що `lxd` запущено та що ваш користувач має привілеї, у підказці оболонки виконайте:

```
lxc list
```

Зауважте, що ви не використали тут `sudo`. Ваш користувач має можливість вводити ці команди. Ви побачите щось на зразок цього:

```
+------------+---------+----------------------+------+-----------+-----------+
|    NAME    |  STATE  |         IPV4         | IPV6 |   TYPE    | SNAPSHOTS |
+------------+---------+----------------------+------+-----------+-----------+
```

Якщо так, то виглядаєте добре!

13.6 Решта

З цього моменту ви можете легко використовувати розділи з нашого «LXD Production Server», щоб продовжити. Однак у налаштуванні робочої станції є деякі речі, на які нам потрібно звертати менше уваги. Ось рекомендовані розділи, які вам допоможуть:

- Розділ 5 - Налаштування та керування зображеннями
- Розділ 6 - Профілі
- Розділ 8 - Snapshots контейнера

13.7 Додатково

- Посібник для початківців LXD, який допоможе вам почати продуктивно використовувати LXD.
- Офіційний огляд і документація LXD

13.8 Висновок

LXD — це потужний інструмент, який можна використовувати на робочих станціях або серверах для підвищення продуктивності. На робочій станції він чудово підходить для лабораторних тестів, але також може зберігати напівпостійні екземпляри операційних систем і програм у їхньому приватному просторі.

https://docs.rockylinux.org/