

[Главная](#)[О авторе](#)

## Linux контейнеры - установка LXC в Debian GNU/Linux, подробная шпаргалка по командам

Расскажу о том что такое Linux контейнеры (LXC) и где они применяются, как быстро установить поддержку этого механизма в Debian GNU/Linux.

Опишу важные настройки в конфигурационном файле контейнеров. Приведу список часто используемых команд (шпаргалку) - LXC cheat sheet. Кратко и по сути.



Содержание:

1. Что такое LXC и для чего это нужно
2. Быстрая установка LXC в Debian GNU/Linux
3. Список директорий, используемых механизмом LXC
4. Конфигурационный файл контейнера и его опции
5. DOWNLOAD IMAGES LIST
6. LIST & INFO
7. MONITOR
8. CREATE
9. START
10. AUTOSTART
11. STOP
12. WAIT
13. FREEZE / UNFREEZE
14. EXECUTE
15. CONSOLE
16. ATTACH

### Категории публикаций:

- Интернет технологии (3)
  - Поисковая оптимизация (SEO) (2)
    - Сервисы и партнерки (0)
    - Хостинг и домены (1)
    - Автоматизация (2)
- Программирование (30)
  - PHP (6)
  - Python (4)
  - Микроконтроллеры (18)
- Операционные системы (14)
  - UNIX FreeBSD (0)
  - Linux (13)
  - Windows (1)
- Информационная безопасность (1)
  - Радиоэлектроника (22)
  - Новости из мира IT (2)
  - Железячки (26)
  - Компьютерные игры (4)
  - Самосовершенствование (11)
  - Здоровье и долголетие (1)
  - Боевые искусства (0)
  - Полезное и необходимое (3)
  - Разное (3)

### Как поддержать проект:

→ ПОМОЩЬ, [DONATE](#)

### Подписаться на рассылку по Email:

- ☒ Публикации  
☐ Комментарии

### Популярные публикации:

Усилитель низкой частоты (УНЧ) на микросхеме TDA7250  
RPi.GPIO - работа с входами, выходами и прерываниями в

- 17. COPY
- 18. SNAPSHOT
- 19. CHECKPOINT
- 20. DESTROY
- 21. CGROUP
- 22. DEVICE
- 23. Диагностика и решение проблем
- 24. В завершение

## Что такое LXC и для чего это нужно

LXC - сокращение от **LinuXContainers**, мощная система управления контейнерами, использующая общее ядро Linux и механизмы разделения/ограничения пользовательского пространства (имен, ресурсов) для запуска служб и программ. Является свободным программным обеспечением.

Механизм LXC контейнеров построен на основе системы CGroups (Control Groups), которая содержится в ядре Linux и позволяет выделять для работы изолированные пространства имен, ограничивать ресурсы процессов и т.п.

По своей сути LXC - это набор программных средств и утилит, которые позволяют управлять встроенными возможностями ядра Linux по созданию и управлению изолированных окружений. Нечто похожее к LXC есть также в ОС FreeBSD - это называется Jails (клетки, изолированные комнаты для процессов).

Исходя из вышесказанного следует ограничение - запуск LXC возможен только на операционных системах класса [GNU/Linux](#).

В отличие от популярного Docker, который является системой контейнеризации ориентированной на изоляцию отдельных служб и приложений (микросервисы, microservices), LXC - это контейнеризация окружения полноценной ОС, базирующейся на Linux ядре хоста, в которой может работать вся инфраструктура ОС с множеством запущенных программ и служб.

LXC контейнер содержит миниатюрную копию файловой системы ОС с базовым набором программ и скриптов, поэтому в некоторых случаях такие образы могут занимать не мало места.

Например чистый контейнер с Debian Stretch (9) для архитектуры amd64 будет занимать примерно 370 МБ. Образ Alpine GNU/Linux amd64 - примерно 44 МБ.

Как видим, размеры готовых контейнеров не такие уж и большие, тем не менее после установки какого-то необходимого ПО они могут быть значительно выше, что справедливо для любой системы контейнеризации.

LXC контейнеры отлично подойдут если вы собираетесь запустить несколько изолированных друг от друга виртуальных серверов для сетевых многопользовательских игр, систем распределения трафика, socks5-серверов и т.п.

Также это хороший выбор при тестировании и разработке программного обеспечения, обеспечивающий минимальные затраты по ресурсам и времени для поднятия необходимого изолированного окружения.

При необходимости в контейнер можно пробросить одну или несколько директорий и даже отдельных файлов, также можно выполнить проброс устройств с хоста внутрь контейнера (например сделать доступной в контейнере USB-вебкамера, подключенную к хосту и т.п.).

Raspberry Pi, примеры на Python  
Изучаем GPIO в Raspberry Pi, эксперимент со светодиодом и кнопкой

Обзор полезных Live CD/DVD/Flash на основе Linux, как записать и протестировать

Работа с регистрами AVR микроконтроллера на Си, битовые операции

DBeaver - свободный менеджер баз данных (MySQL, PostgreSQL, Firebird, SQLite, Oracle)

Самодельный роутер и мини-сервер на Raspberry Pi - Часть 2 (программы)

Как сделать звукосниматель для акустической гитары

CPU стресс-тест в Linux, как нагрузить все ядра микропроцессора

Ремонт усилителя Радиотехника У-101, модуль УМЗЧ на микросхеме TDA7250

### Интересный опрос:

Откуда вы узнали о моем сайте?

- ☐ Gogole
- ☐ Yandex
- ☐ Youtube
- ☐ Twitter
- ☐ Из форума
- ☐ От друга
- ☐ Другое

Проголосовать

Результаты голосования

Хочу отметить удобство клонирования и резервирования контейнеров, доступны снимки (snapshots). Например, чтобы сделать резервную копию контейнера достаточно его остановить и упаковать директорию с файлами (rootfs, config) в архив.

Все контейнеры можно держать вместе как на одной машине, так и на отдельных связанных сетью хостах. Конфигурация и управление ими - очень простые.

А еще эта технология поддерживает возможность создания вложенных изолированных контейнеров, то есть в одном из готовых контейнеров вы можете установить LXC и создать внутри еще несколько вложенных изолированных друг от друга контейнеров.

LXC - отличный инструмент!

## Быстрая установка LXC в Debian GNU/Linux

Установка среды изоляции и управления LXC очень проста, все необходимое для работы уже содержится в официальных пакетах для GNU/Linux, поэтому нужно выполнить лишь несколько команд и провести простую настройку.

Установка свежего пакета lxc:

```
1 sudo apt-get update
2 sudo apt-get install lxc
```

Установка пройдет быстро, размер закупаемых пакетов составит всего лишь несколько мегабайт.

Проверка конфигурации и доступных возможностей:

```
1 sudo lxc-checkconfig
```

Теперь, если с конфигурацией все хорошо, осталось настроить и включить постоянный сетевой мост для контейнеров. Сперва откроем в редакторе файл 'lxc-net':

```
1 sudo nano /etc/default/lxc-net
```

Добавим в этот файл строчку:

```
1 USE_LXC_BRIDGE="true"
```

Сохраним файл и выйдем из редактора (CTRL+X, Y).

Изменим шаблон с опциями настройки по умолчанию для контейнеров, отредактируем файл default.conf:

```
1 sudo nano /etc/lxc/default.conf
```

Добавляем строчки конфигурирования сети и безопасности:

В LXC 3.0 (для Debian 10)

```
1 lxc.apparmor.profile = generated
2 lxc.apparmor.allow_nesting = 1
3
4 lxc.net.0.type = veth
5 lxc.net.0.link = lxcbr0
6 lxc.net.0.flags = up
7 lxc.net.0.hwaddr = 00:16:3e:xx:xx:xx
```

```
1 lxc.network.type = veth
2 lxc.network.link = lxcbr0
3 lxc.network.flags = up
4 lxc.network.hwaddr = 00:16:3e:xx:xx:xx
```

Сохраняем файл и выходим.

Перезапустим сетевую службу для LXC:

```
1 sudo systemctl restart lxc-net.service
```

Проверим создан ли интерфейс виртуального сетевого моста - lxcbr0:

```
1 ip a
```

Если интерфейс присутствует в списке - значит сетевой мост настроен, можно приступать к работе с LXC контейнерами!

IP-адрес и маска сети хостовой машины по умолчанию - 10.0.3.1/24, контейнеры будут получать IP-адреса из подсети 10.0.3.0/24.

Хочу обратить внимание на то, что данной конфигурации вполне достаточно для работы из под привилегированного пользователя (root), но не достаточно для выполнения команд от имени обычного пользователя.

При запуске lxc-команд от пользователя root и от обычного пользователя - размещение контейнеров и файлов с настройками LXC будут отличаться!

О настройке и запуске контейнеров от имени непривилегированного пользователя можно узнать на страничке [2].

## Список директорий, используемых механизмом LXC

Для привилегированного пользователя (root):

- /var/lib/lxc - директория где хранятся контейнеры и их настройки;
- /etc/lxc - настройки LXC;
- /etc/default/lxc-net - настройки сетевой подсистемы;
- /var/cache/lxc - скачиваемые образы файловых систем ОС;
- /var/lib/lxcsnaps - тут хранятся снапшоты;
- /var/log/lxc - логи;
- /usr/share/lxc/templates/ - шаблоны.

Для НЕ привилегированного пользователя:

- ~/.local/share/lxc - директория где хранятся контейнеры и их настройки;
- ~/.config/lxc - настройки LXC;
- ~/.cache/lxc - скачиваемые образы файловых систем ОС;
- ~/.local/share/lxcsnaps - тут хранятся снапшоты;

## Конфигурационный файл контейнера и его опции

Конфигурационные файлы контейнеров носят имя 'config' и хранятся в директориях соответствующих контейнеров. Например, путь к конфигурационному файлу контейнера 'container1' для привилегированного пользователя - '/var/lib/lxc/container1/config'.

Конфигурационный файл любого из созданных контейнеров можно открыть для редактирования в любом удобном вам текстовом редакторе.

Пример загрузки конфигурационного файла для контейнера 'container1' в консольном редакторе 'nano':

```
1 nano /var/lib/lxc/container1/config
```

Опция для ограничения доступно контейнеру памяти:

```
1 # Memory limit = 256M
2 lxc.cgroup.memory.limit_in_bytes = 256M
```

Опция для ограничения количества ядер CPU:

```
1 # CPU Cores
2 lxc.cgroup.cpuset.cpus = 1
```

Монтируем файл /etc/rinetd.conf из хостовой машины в контейнер:

```
1 lxc.mount.entry = /etc/rinetd.conf etc/rinetd.conf none bind,optional,create
```

Примечание: путь назначения (в контейнере) не должен начинаться с бек-слеша, это путь относительно корня контейнера.

Монтируем файл блочного устройства /dev/sde внутри контейнера, создаем файл если он не существует в точке назначения:

```
1 lxc.mount.entry = /dev/sde dev/sde none bind,optional,create
```

Монтируем директорию /media/zzz на хосте в директорию /var/lib/mysql/zzz внутри контейнера:

```
1 lxc.mount.entry=/media/zzz var/lib/mysql/zzz none bind,create
```

Монтирование директории /dev/mapper/lvmfs-home из другой файловой системы в контейнер:

```
1 lxc.mount.entry = /dev/mapper/lvmfs-home home ext4 default
```

Настройки автозапуска контейнера (включение, задержка, порядок запуска):

```
1 lxc.start.auto = 1
2 lxc.start.delay = 15
3 lxc.start.order = 50
```

Указываем группу для контейнера:

```
1 lxc.group = ph0en1x_servers
```

Перед стартом контейнера 'container1' выполнить 'хук' (hook) - запустить скрипт 'prepare-start.sh', который лежит в директории контейнера (рядом с конфигурационным файлом):

```
1 lxc.hook.pre-start = /var/lib/lxc/container1/prepare-start.sh
```

Список других полезных хуков для LXC:

--	--	--



Опция	Описание	Исполнение скрипта
lxc.hook.pre-start	Перед стартом контейнера (окончание загрузки консоли, монтирования файловых систем)	Хост
lxc.hook.pre-mount	Перед монтированием rootfs	Контейнер
lxc.hook.mount	После монтирования файловых систем, перед pivot_root	Контейнер
lxc.hook.autodev	После процедуры монтирования и других хуков связанных с монтированием, перед pivot_root	Контейнер
lxc.hook.start	Прямо перед инициализацией INIT	Контейнер
lxc.hook.stop	После остановки контейнера	Хост
lxc.hook.post-stop	После остановки контейнера	Хост
lxc.hook.clone	Когда контейнер клонирован	Хост
lxc.hook.destroy	Когда контейнер уничтожен	Хост
lxc.network.script.up lxc.net.[i].script.up	После создания сетевого интерфейса	Хост
lxc.network.script.down lxc.net.[i].script.down	Перед уничтожением сетевого интерфейса	Хост

Список сетевых настроек контейнера:

Опция	Описание	Примеры значений
lxc.network.type lxc.net.[i].type	Тип сетевой виртуализации	none, empty, veth, vlan, macvlan, phys
lxc.network.link lxc.net.[i].link	Сетевой интерфейс на Хосте	eth0, wlan0...
lxc.network.flags lxc.net.[i].flags	Действие, выполняемое на машине	up
lxc.network.hwaddr lxc.net.[i].hwaddr	MAC-адрес сетевого интерфейса	00:16:3e:11:22:33
lxc.network.mtu lxc.net.[i].mtu	Maximum Transfer Unit (MTU)	1500
lxc.network.name lxc.net.[i].name	Имя сетевого интерфейса	
lxc.network.ipv4 lxc.net.[i].ipv4	IPv4 адрес интерфейса	10.0.3.55/24
lxc.network.ipv4.gateway lxc.net.[i].ipv4.gateway	IPv4 адрес шлюза	10.0.3.1/24
lxc.network.ipv6 lxc.net.[i].ipv6	IPv6 адрес интерфейса	
lxc.network.ipv6.gateway	IPv6 адрес шлюза	

lxc.net.[i].ipv6.gateway		
--------------------------	--	--

Примечание: опции формата "lxc.net.[i]." - для LXC версии 3.0. Где "[i]" - цифра, номер сетевого интерфейса. Например: "lxc.net.0.ipv4" - для первого сетевого интерфейса.

Более детальную информацию о конфигурационном файле для контейнера в LXC можно узнать из [3] или же используя встроенную в GNU/Linux справочную систему man:

1	<code>man lxc.container.conf</code>
---	-------------------------------------

### DOWNLOAD IMAGES LIST

Как узнать список доступных к скачиванию образов в LXC? - достаточно запустить команду создания контейнера с опцией "-t download" без указания архитектуры, имени и версии образа. На экран будет выведен весь список хранящихся удаленно образов, программа попросит вас выбрать дистрибутив.

Пример команды:

1	<code>lxc-create --template download --name test1</code>
---	--

Вот часть из вывода команды:

1	Setting up the GPG keyring
2	Downloading the image index
3	
4	---
5	DIST      RELEASE ARCH      VARIANT BUILD
6	---
7	alpine    3.10      amd64      default 20190921_13:00
8	alpine    3.10      arm64      default 20190921_13:00
9	.....
10	debian    sid        s390x      default 20190922_05:24
11	debian    stretch amd64      default 20190922_05:24
12	debian    stretch arm64      default 20190922_05:24
13	debian    stretch armel      default 20190922_05:24
14	debian    stretch armhf      default 20190922_05:55
15	debian    stretch i386        default 20190922_05:24
16	debian    stretch ppc64el      default 20190922_05:24
17	debian    stretch s390x        default 20190922_05:24
18	.....
19	voidlinux            current i386        default 20190921_17:10
20	---
21	
22	Distribution:

Для создания контейнера на основе Debian - вводим слово 'debian', на запрос релиза - например, 'stretch', а на запрос архитектуры, например 'amd64'.

Если контейнер не нужно создавать, то для отмены можно нажать CTRL+C.

### LIST & INFO

Узнаем версию установленной подсистемы LXC:

```
1 lxc-info --version
```

Список существующих (в директории хранения по умолчанию) контейнеров (только имена):

```
1 lxc-ls
```

Список существующих контейнеров включая информацию (статус, автозапуск, группы, IPv4/6):

```
1 lxc-ls -f
```

Список контейнеров, работающих из директории (тома) '/media/ssd1':

```
1 lxc-ls -f -P /media/ssd1
2 lxc-ls -f --lxcpath=/media/ssd1
```

Информация о контейнере 'container1':

```
1 lxc-info -n container1
```

Пример вывода этой команды:

```
1 Name:          container1
2 State:         RUNNING
3 PID:          32293
4 IP:           10.0.3.152
5 CPU use:      0.35 seconds
6 BlkIO use:    16.00 KiB
7 Memory use:   17.21 MiB
8 KMem use:     3.71 MiB
9 Link:         vethVQSF7J
10 TX bytes:    1.58 KiB
11 RX bytes:    1.28 KiB
12 Total bytes: 2.85 KiB
```

Также узнать используемый контейнером размер памяти можно следующей командой:

```
1 cat /sys/fs/cgroup/memory/lxc/container1/memory.usage_in_
```

Заменив в вышеприведенной команде 'memory.usage\_in\_bytes' на 'memory.stat' можно узнать очень подробную информацию по использованию памяти в указанном контейнере.

## MONITOR

Мониторинг состояний контейнера 'container1':

```
1 lxc-monitor -n container1
```

Мониторинг нескольких указанных контейнеров:

```
1 lxc-monitor -n 'container1|container2'
```

Мониторинг по шаблону для имени (начинается со слова 'container' либо с буквы 'X'):

```
1 lxc-monitor -n '[container|X].*'
```



## CREATE

Создать контейнер 'container1', предлагая перед этим выбрать дистрибутив:

```
1 lxc-create -t download -n container1
```

Создать контейнер с образом ОС Debian и релизом Stretch:

```
1 lxc-create -n container1 -t debian -- -r stretch
```

Создать контейнер с ОС Debian, релизом Stretch и архитектурой amd64:

```
1 lxc-create -t download -n container1 -- -d debian -r stretch
```

Создать контейнер с Debian Jessie (8) с ограниченной файловой системой размером 10ГБ:

```
1 lxc-create -B loop --fssize 10G -t debian -n container1
```

Создать контейнер, разместив его содержимое в директории '/media/ssd1/container1':

```
1 lxc-create -P /media/ssd1/ -n container1 -t debian -- -r stretch
2 lxc-create --lxcpath==/media/ssd1 -n container1 -t debian
```

## START

Запуск контейнера 'container1'

```
1 lxc-start -n container1
```

Запуск контейнера, хранящегося в директории '/media/ssd1':

```
1 lxc-start -P /media/ssd1 -n container1
```

Запуск в фоновом режиме (daemon):

```
1 lxc-start -n container1 -d
```

Запуск с записью лога в файл /var/log/lxc.log:

```
1 lxc-start --logfile=/var/log/lxc.log -d -n container1
```

Запуск + запись лога в файл + установка приоритета логирования 'DEBUG':

```
1 lxc-start -d -l DEBUG --logfile=debug.log -n container1
```

Список значений для опции '-l': FATAL, CRIT, WARN, ERROR, NOTICE, INFO, DEBUG.

Запуск контейнера на переднем плане (автоматическое подключение к консоли):

```
1 lxc-start --foreground -n container1
```

## AUTOSTART

Запуск контейнеров, у которых в конфигурационном файле установлена опция 'lxc.start.auto'.

Список контейнеров с опцией автозапуска:

```
1 lxc-autostart --list
```

Запуск всех контейнеров имеющих возможность автозапуска:

```
1 lxc-autostart --all
```

Вывести список контейнеров в группе 'socks5':

```
1 lxc-autostart --list --group socks5
```

Запуск всех контейнеров в группе 'socks5':

```
1 lxc-autostart --group socks5
```

Запуск групп контейнеров в порядке их указания:

```
1 lxc-autostart -g "lamp,socks5,mangos"
```

## STOP

---

Остановка контейнера 'container1':

```
1 lxc-stop -n container1
```

Принудительно завершить работу контейнера (без корректной остановки, KILL):

```
1 lxc-stop -k -n container1
```

## WAIT

---

Ожидаем пока контейнер 'container1' не окажется в статусе STOPPED:

```
1 lxc-wait -n container1 -s 'STOPPED'
```

Ждем когда контейнер получит любой из статусов - RUNNING или STARTING:

```
1 lxc-wait -n container1 -s 'RUNNING|STARTING'
```

Список возможных статусов контейнера:

- RUNNING;
- STOPPED;
- STARTING;
- STOPPING;
- ABORTING;
- FREEZING;
- FROZEN.

## FREEZE / UNFREEZE

---

Заморозить процессы в контейнере 'container1':

```
1 lxc-freeze -n container1
```

Разморозить этот же контейнер:

```
1 lxc-unfreeze -n container1
```

Узнать статус заморозки у контейнера:

```
1 cat /sys/fs/cgroup/freezer/lxc/container1/freezer.state
```

## EXECUTE

Запуск приложения 'app1.bin' в контейнере 'container1':

```
1 lxc-execute -n container1 app1.bin
```

## CONSOLE

Подключение текущего терминала к консоли контейнера 'container1':

```
1 lxc-console -n container1
```

Для отключения используйте комбинацию клавиш CTRL+A, потом нажмите Q.

## ATTACH

Запуск процесса 'passwd' в запущенном контейнере 'container1' для установки пароля суперпользователя:

```
1 lxc-attach -n container1 passwd
```

Запуск командного интерпретатора BASH внутри контейнера и доступ к нему в текущем терминале:

```
1 lxc-attach -n container1 /bin/bash
```

Выйти из запущенного в контейнере интерпретатора команд можно введя команду 'exit' или же нажав комбинацию клавиш CTRL+D.

Запуск в контейнере команды 'df' с ключем '-h' (вывод статистики файловых систем контейнера):

```
1 lxc-attach -n container1 -- df -h
```

Разделитель '--' указывает на то что следующие параметры не нужно интерпретировать как параметры команды 'lxc-attach', а отнести их к команде для выполнения внутри указанного контейнера.

## COPY

Копирование контейнера под именем 'container1' в новый контейнер под именем 'container1\_COPY':

```
1 lxc-copy -n container1 --newname container1_COPY
```

Копирование контейнера 'lamp-server', размещенного в директории /media/ssd1 в контейнер под названием 'lamp-srv-clone' (будет размещен в той же директории):

```
1 lxc-copy -n lamp-server -P /media/ssd1 --newname lamp-srv
```

## SNAPSHOT

Вывести список снэпшотов для контейнера 'container1':

```
1 lxc-snapshot -n container1 -L
```

Создать снимок для этого же контейнера:

```
1 lxc-snapshot -n container1
```

Восстановить состояние контейнера 'container1' из снимка 'snap0':

```
1 lxc-snapshot -n container1 -r snap0
```

Создать (восстановить) контейнер 'container1-from-snap0' из снимка 'snap0', созданного для контейнера 'container1':

```
1 lxc-snapshot -n container1 -r snap0 container1-from-snap0
```

## CHECKPOINT

Сохранить состояние контейнера на диск (в директории '/media/checkpoint') с последующей возможностью восстановления этого состояния:

```
1 lxc-checkpoint -n container1 -D /media/checkpoint
```

Восстановление состояния контейнера:

```
1 lxc-checkpoint -r -n container1 -D /media/checkpoint
```

## DESTROY

Уничтожить контейнер 'container1':

```
1 lxc-destroy -n container1
```

Примечание: перед уничтожением контейнер должен быть [остановлен](#).

## CGROUP

Получить значение лимита памяти в Байтах для контейнера 'container1':

```
1 lxc-cgroup -n container1 memory.limit_in_bytes
```

Примечание: если значение с лимитом памяти для контейнера не было установлено то команда выведет очень большое целое число.

Установка лимита памяти для контейнера - 256МБ:

```
1 lxc-cgroup -n container1 memory.limit_in_bytes 256000000
```

Установка лимита памяти для контейнера - 512МБ:

```
1 lxc-cgroup -n container1 memory.limit_in_bytes 512M
```

Другие значения параметров для возможной установки:

- `cpu.shares` - приоритет использования CPU, пример: `A = 2000`, `B = 1000` - контейнер B будет иметь в 2 раза меньший приоритет по использованию ЦПУ чем контейнер A;
- `cpuset.cpus` - количество и/или номера ядер CPU для использования в контейнере, примеры: `0,1,2` или `0-3`;
- `memory.memsw.limit_in_bytes` - суммарный лимит памяти (ОЗУ + файл подкачки);
- `blkio.weight` - приоритет использования блочной системы ввода-вывода (IO).

Более подробную информацию по CGroups вы можете узнать по ссылке [3].

## DEVICE

Подключение (проброс) устройства с файлом '/dev/video0' (напрмиер веб-камера) в контейнер 'container1':

```
1 lxc-device -n container1 add /dev/video0
```

Подключение сетевого адаптера с интерфейсом 'veth0' (на хост-машине) к контейнеру и назначение в нем нового имени интерфейса eth1:

```
1 lxc-device -n container1 add veth0 eth1
```

## Диагностика и решение проблем

Может случиться так, что в какой-то момент (после высокой загрузки сервера, перевода ноутбука в спящий режим и т.п.) созданный LXC контейнер потерял свой IP-адрес или же и вовсе создаваемые контейнеры перестали получать сетевые настройки.

В такой ситуации, первым делом нужно проверить работу сервисов на хост-машине. Смотрим присутствует ли в системе сетевой интерфейс (мост, bridge) для LXC-контейнеров и каковы его настройки:

```
1 ip a show lxcbr0
```

Если с этим интерфейсом все хорошо, то в результате исполнения команды увидим IP-адрес моста и другие параметры:

```
1 5: lxcbr0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qd:
2 link/ether 00:16:3e:00:00:00 brd ff:ff:ff:ff:ff:ff
3 inet 10.0.3.1/24 scope global lxcbr0
4 valid_lft forever preferred_lft forever
5 inet6 fe80::216:3eff:fe00:0/64 scope link
6 valid_lft forever preferred_lft forever
```

Для получения контейнерами сетевых настроек от хостовой машины механизмом LXC используется служба "dnsmasq", проверяем запущена ли она:

```
1 ps ax | grep "[d]nsmasq"
```

Если служба работает и присутствует в процессах, то в выводе вышеуказанной команды мы увидим строку, где будет указан PID процесса и его параметры, например (для удобства чтения я перенес параметры в отдельные строчки):

```
1 16734 ? S 0:00 dnsmasq -u dnsmasq --strict-o
2 --pid-file=/run/lxc/dnsmasq.pid --listen-address 10.0.3.1
3 --dhcp-range 10.0.3.2,10.0.3.254 --dhcp-lease-max=253 --c
4 --except-interface=lo --interface=lxcbr0
5 --dhcp-leasefile=/var/lib/misc/dnsmasq.lxcbr0.leases --df
```

Как видим, здесь указан интерфейс и IP-адрес сетевого моста, на котором работает служба, а также пул адресов, доступный для присваивания LXC-контейнерам (2-254).

Теперь переходим к диагностике сети внутри контейнера, приведенные ниже команды должны выполняться в консоли запущенного LXC-контейнера.

Смотрим список доступных и активных сетевых интерфейсов внутри контейнера:

```
1 ip a
```

При настройках по умолчанию, как правило, в списке должны присутствовать минимум 2 сетевых интерфейса:

- `lo` - loopback (локальная петля), локальный сетевой интерфейс;
- `eth0` - сетевой интерфейс, связанный с хост-машиной.

```
1 1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state
2     link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
3     inet 127.0.0.1/8 scope host lo
4         valid_lft forever preferred_lft forever
5     inet6 ::1/128 scope host
6         valid_lft forever preferred_lft forever
7 17: eth0@if18: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500
8     link/ether 00:16:3e:53:02:cd brd ff:ff:ff:ff:ff:ff
9     inet 10.0.3.139/24 brd 10.0.3.255 scope global eth0
10        valid_lft forever preferred_lft forever
11     inet6 fe80::216:3eff:fe53:2cd/64 scope link
12        valid_lft forever preferred_lft forever
```

Если у сетевого интерфейса `eth0` нет IP-адреса, то возможно что причина таится где-то в самом контейнере и его службах.

Посмотрим получил ли контейнер список DNS-серверов от сервиса `dnsmasq`, запущенного на хост-машине, выполним команду внутри контейнера:

```
1 cat /etc/resolv.conf
```

В случае если все сконфигурировано правильно, в выводе получим строку:

```
1 nameserver 10.0.3.1
```

Если же полученных настроек IP-адреса и DNS-сервера в контейнере нет, то можем заставить его сделать повторную попытку получения этих сетевых настроек по протоколу DHCP от хост-машины:

```
1 dhclient -r
```

Теперь можем пропинговать какой-то сайт и проверить корректно получены ли сетевые настройки:

```
1 ping reddit.com
```

Исправность работы сетевого механизма LXC-системы проще всего проверить создав какой-нибудь новый контейнер. Если новосозданные контейнеры не получают сетевых настроек - проблема, скорее всего, на стороне хост-машины, в каком-то звене сетевой подсистемы LXC.

Очень вероятно что проблему удастся решить перезапуском сетевой службы LXC на хост-машине:

```
1 systemctl restart lxc-net.service
```

Для перезапуска всего механизма LXC:

```
1 systemctl restart lxc.service
```



Если и это не помогло, то может быть что где-то сбились основные настройки сети LXC, стоит перепроверить все параметры, которые были задействованы [на этапе установки и настройки механизма LXC](#).

Еще не помешает проверить правила файрвола [IPTables](#), которые могут блокировать сетевые пакеты на пути от хост-машины к контейнерам.

В более сложных случаях возможна даже не работоспособность самой службы "dnsmasq", нужно изучить ее логи и на основе этих данных устранить проблему.

Еще одна проблема, которая может возникать - невозможность получения GPG ключей от доверенных серверов по причине их перегрузки или неработоспособности.

Результатом станет то, что скачать шаблон и создать из него контейнер не получится, будет выдана следующая ошибка:

```
1 Setting up the GPG keyring
2 ERROR: Unable to fetch GPG key from keyserver
3 lxc-create: test1: lxccontainer.c: create_run_template: 1
4 lxc-create: test1: tools/lxc_create.c: main: 327 Failed to
```

Решить эту проблему можно экспортировав в окружение специальную переменную, в которой указать адрес сервера с которого должны браться ключи.

```
1 # Set environment variable
2 # Also you can add this command in your ~/.bashrc
3 export DOWNLOAD_KEYSERVER="pgp.mit.edu"
4
5 # Run commands
6 lxc-create ...arguments...
```

Список некоторых доверенных серверов, для пробы:

1. [pgp.mit.edu](https://pgp.mit.edu)
2. [keyserver.ubuntu.com](https://keyserver.ubuntu.com)

Также при запуске команды 'lxc-create' можно отключить проверку цифровой подписи для скачиваемого шаблона, для этого достаточно добавить опцию '--no-validate'. Но я так делать не рекомендую!

## В завершение

В дополнение к приведенной выше шпаргалке по командам могу сказать что документация по ним всегда есть под рукой - это система man pages. Пример вызова справки по команде lxc-snapshot:

```
1 man lxc-snapshot
```

LXC - очень мощный и относительно не сложный в использовании инструмент, позволяющий запускать в изолированной среде исполнения как отдельные приложения, так и целые связки из приложений. При этом каждое приложение будет работать в полноценной среде выбранной операционной системы Debian, Ubuntu, CentOS, Alpine и т.п.

Открытый исходный код, свобода распространения и простая установка из репозитория GNU/Linux делают эти контейнеры очень привлекательными для использования в тестировании и разработке программ и информационных систем.

Используя вышеприведенные команды можно построить небольшой [менеджер для удобного управления Linux-контейнерами](#) на одном из популярных языков программирования или и вовсе на SH/Bash-скриптах.

Полезные ссылки и литература:

1. <https://linuxcontainers.org/>
2. [Debian Wiki - LXC](#);
3. [kernel.org - CGroups v.1](#)
4. [lxc.container.conf manpage](#)
5. [Ivanov K. - "Containerization with LXC", 2017.](#)

2 5684 Linux

Тематика: [LXC](#) [Linux](#) [virtualization](#)

Также может заинтересовать:

- [Самодельный роутер и мини-сервер на Raspberry Pi - Часть 2 \(программы\)](#)
- [Компиляция и сборка Wargus \(Warcraft 2\) на Debian 9 GNU/Linux, пошаговое руководство](#)
- [Скрипт фаервола IPTables для ipv4 и ipv6 с автозагрузкой в Linux \(systemd\)](#)
- [Установка ISPManager Lite на сервер с операционной системой Linux](#)
- [Установка Firebird Server Classic 1.5 на Linux Ubuntu 12 \(64-бита\)](#)

#### Комментарии к публикации (2):



#1 **shai**

18 Май 2019 20:19

+2 +

Thanks !! 🙏



#2 **ph0en1x**

03 Декабрь 2020 11:41

0 +

Добавлен пример конфигурационного файла для настройки постоянного сетевого моста под GNU/Linux Debian 10 (Buster).

Текст комментария

Смайлики :)

Имя, никнейм

Почтовый адрес E-Mail

Символы с картинки

Отправить комментарий



© 2012-2022, Ph0en1x.net - информационная безопасность, SEO, программирование, UNIX и Linux, радиоэлектроника и саморазвитие.  
Все права защищены. Копирование информации разрешается при условии установки активной ссылки на оригинал.

🕒 188	266
🕒 328	455