

Виртуализация Linux: создание безопасной и простой рабочей среды

В этой главе

- Определение правильной технологии виртуализации.
- Использование менеджеров репозитория Linux.
- Создание эффективных сред с использованием VirtualBox.
- Построение контейнеров с помощью LXC.
- Профессиональное управление виртуальными машинами.

Виртуализация — это самая важная технология, стоящая за почти всеми недавними улучшениями в предоставлении сервисов и продуктов. Она сделала не только возможными, но и необходимыми целые отрасли промышленности: от облачных вычислений до самоуправляемых автомобилей. Заинтересовались? Вот два факта о виртуализации, которые вам нужно знать с самого начала.

- Linux абсолютно доминирует в виртуальном пространстве.
- Виртуализация облегчает изучение любой технологии.

Эта глава дает представление о доминирующих технологиях виртуализации уровня предприятия, используемых в настоящее время. Но что еще важнее, здесь вы также узнаете, как задействовать виртуальную среду, чтобы безопасно получать

в ней навыки администрирования Linux. Почему эта довольно сложная технология так рано появляется в книге? Потому что так вам будет намного легче проработать остальные главы.

Нужна свежая, чистая операционная система (ОС), чтобы попробовать что-то новое? Создайте ее за несколько секунд. Сделали ошибку в конфигурации и она заблокировала вам доступ к машине? Нет проблем. Уничтожьте эту конфигурацию и запустите новую. Вы узнаете, как использовать диспетчеры пакетов Linux для загрузки и установки всего необходимого программного обеспечения (например, VirtualBox и LXC) и управления им.

2.1. Что такое виртуализация

Когда вам понадобился новый сервер, чтобы запустить веб-сервер или хранилище документов с совместным доступом для вашей компании или ее клиентов, вам пришлось провести исследование, запросить одобрение бюджета, согласовать, заказать, безопасно разместить, обеспечить и затем запустить совершенно новую машину. Этот процесс от начала до конца мог занять месяцы (поверьте мне, я с таким сталкивался). И если бы растущие потребности сервиса грозили превысить возможности сервера, вам пришлось бы повторить все сначала в надежде правильно угадать соотношение «производительность/требования».

Стандартный сценарий предполагает, что компания предоставляет несколько взаимозависимых сервисов, каждый из которых работает на собственном оборудовании. Представьте себе фронтенд-веб-сервер, развернутый вместе с базой данных в бэкенде. Однако спустя время вы сталкиваетесь с ситуацией, когда один сервер используется недостаточно эффективно, а другой (как правило, рядом с первым в стойке) не справляется с нагрузкой. Но представьте, что вы можете безопасно разделить память, хранилище и сетевые ресурсы одного высокопроизводительного сервера между несколькими сервисами. Вообразите себе возможность выделять экземпляры виртуальных серверов из этого физического сервера, назначая им только необходимый уровень ресурсов, а затем мгновенно настраивая ресурсы для удовлетворения меняющихся потребностей.

А теперь представьте, что вы можете эффективно упаковать дюжины этих виртуальных компьютеров, работающих под управлением нескольких операционных систем, на одном сервере с «голым железом» так, чтобы абсолютно ничего не проистекало. Представьте себе, что эти виртуальные машины (ВМ) могут автоматически распространяться на другие физические серверы при заполнении первых. Подумайте о том, как удобна возможность уничтожить виртуальную машину, которая вышла из строя или нуждается в обновлении, и заменить ее так быстро, что пользователи даже не поймут, что что-то изменилось. Представили себе это (надеюсь, получилось что-то вроде рис. 2.1)? Это и есть *виртуализация*.

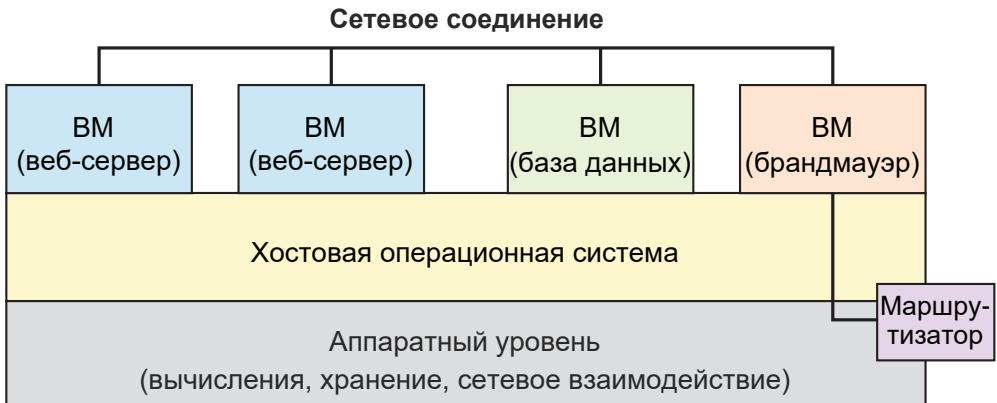


Рис. 2.1. Виртуальные машины — клиенты аппаратного хоста с возможностью их соединения между собой и с большей сетью через внешний маршрутизатор

Этот подход настолько привлекателен, что теперь доминирует в мире корпоративных вычислений. Я сомневаюсь, что на данный момент осталось много локальных или облачных серверных приложений, которые не работают с какой-либо технологией виртуализации. И Linux — та операционная система, на которой работает большинство этих виртуальных рабочих нагрузок.

Amazon Web Services (AWS), кстати, позволяют пользователям арендовать мощности на (Linux-) серверах, где размещены миллионы виртуальных машин, на которых, в свою очередь, работает бесчисленное множество полезных программ, включая большинство наиболее популярных онлайн-сервисов. На рис. 2.2 показано, как экземпляр VM AWS Elastic Compute Cloud (EC2) служит центром для целого комплекса инструментов хранения, баз данных и сетевого взаимодействия.

Не волнуйтесь, если некоторые из этих деталей AWS немного непонятны — мы не будем изучать их в этой книге. Но если вам захочется больше узнать о Amazon Web Services, вы всегда можете прочитать мою книгу *Learn Amazon Web Services in a Month of Lunches* (Manning, 2017). А что насчет виртуализации? У меня есть книга и на эту тему: *Teach Yourself Linux Virtualization and High Availability* (LULU Press, 2017).

Следующий короткий раздел может показаться немного трудным, но он обеспечит некоторой информацией тех из вас, кто хочет понимать, как все работает, так сказать, за кадром. При успешной виртуализации используется какое-то изолированное пространство на физическом компьютере, куда можно установить гостевую ОС, а затем «убедить» ее, что она одна на своем собственном компьютере. Гостевые операционные системы могут совместно использовать сетевые подключения, чтобы их администраторы могли удаленно входить в систему (о чем я расскажу в главе 3)

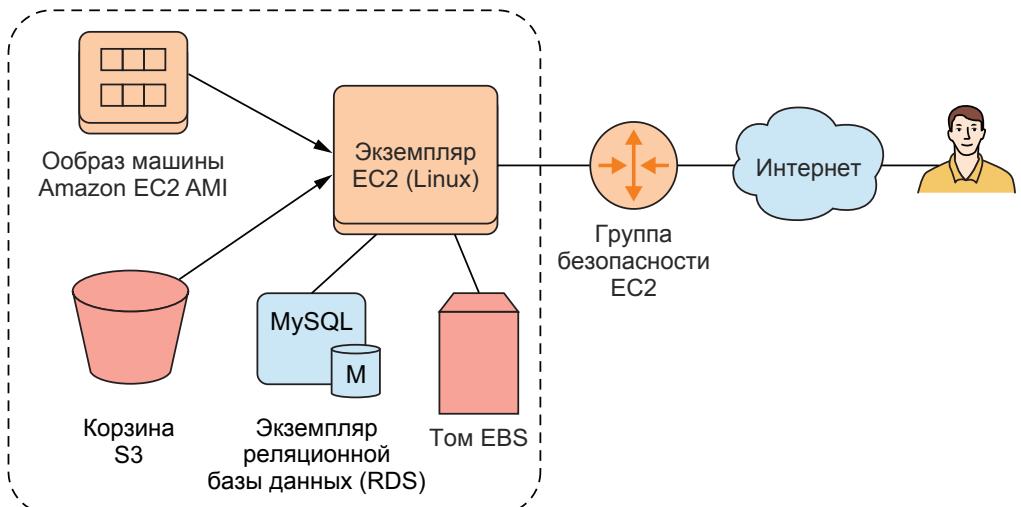


Рис. 2.2. Типичная рабочая схема при облачных вычислениях, сосредоточенная вокруг экземпляров виртуальных машин AWS Elastic Cloud Compute (EC2) на Amazon Web Services

и выполнять свою работу точно так же, как на традиционных машинах. Те же общие сетевые подключения позволяют вам использовать виртуальные машины для предоставления публичных сервисов, таких как сайты. Вообще говоря, в настоящее время существует два средства виртуализации.

- **Гипервизоры** – в той или иной степени управляют оборудованием хост-системы, предоставляя каждой гостевой ОС необходимые ей ресурсы (рис. 2.3). Гостевые машины запускаются как системные процессы, но с виртуализированным доступом к аппаратным ресурсам. Например, серверы AWS давно построены на технологии гипервизора Xen с открытым исходным кодом (хотя недавно они начали переключать некоторые из своих серверов на платформу KVM все с тем же открытым исходным кодом). Другие важные гипервизорные платформы включают VMware ESXi, KVM и Microsoft Hyper-V.
- **Контейнеры** – чрезвычайно легкие виртуальные серверы, которые вместо того, чтобы работать как полноценные операционные системы, совместно используют ядро своей хостовой машины (рис. 2.4). Контейнеры могут быть построены на базе текстовых сценариев, созданы и запущены за считанные секунды, а также легко доступны для совместного использования в сетях. Самая известная контейнерная технология сейчас, вероятно, Docker. Проект Linux Container (LXC), с которым мы будем работать в этой главе, послужил изначальным источником вдохновения для Docker.

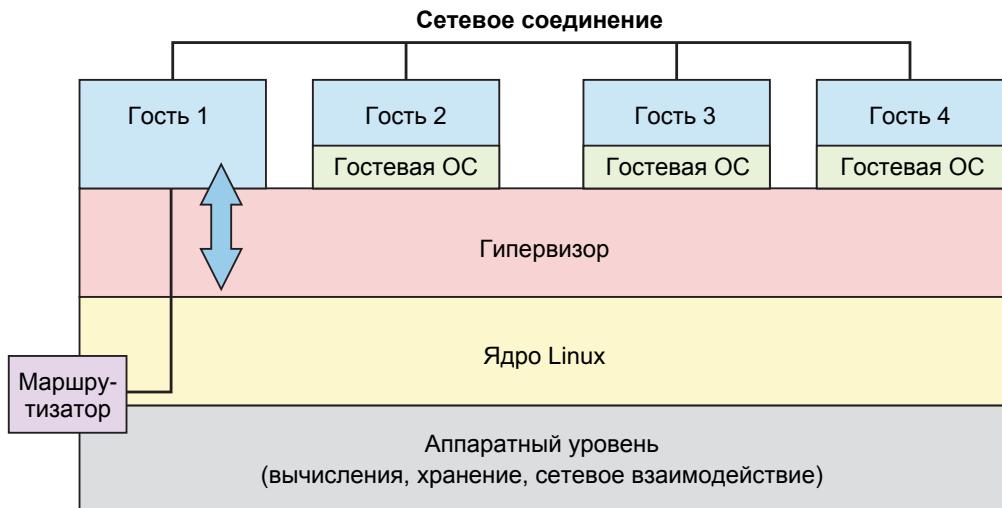


Рис. 2.3. Архитектура гипервизора типа 2, демонстрирующая целые операционные системы, установленные на каждом госте, с некоторыми специальными административными обязанностями, делегированными гостю 1

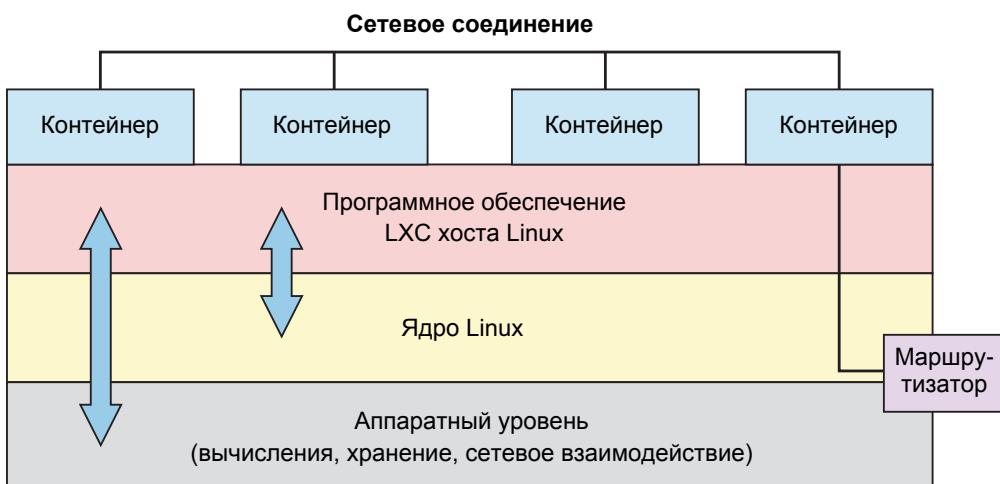


Рис. 2.4. Архитектура LXC, иллюстрирующая возможность доступа между средой LXC и как ядром Linux, так и аппаратным уровнем под ним

Ни одна технология не подходит для всех проектов идеально. Но если вы решите прочитать эту главу до конца, то узнаете, как и зачем использовать две технологии виртуализации: VirtualBox (гипервизор типа 2) и, как я упоминал ранее, LXC (менеджер контейнеров).

Проектные соображения

Я бы не хотел, чтобы вы закончили читать эту книгу, не получив некоторых базовых рекомендаций по выбору технологий виртуализации как минимум, поэтому вот несколько соображений.

- Полномасштабные гипервизоры, такие как Xen и KVM (через интерфейс управления, такой как Libvirt), обычно используются для развертываний на уровне предприятия, так как имеют большой парк виртуальных машин Linux.
- VirtualBox (и VMware Player) идеально подходит для тестирования работающих операционных систем и экспериментов с ними, по одной или две за раз, без необходимости их установки на настоящие ПК. При этом относительно высокие накладные расходы делают их непривлекательными для большинства производственных сред.
- Контейнерные технологии, такие как LXC и Docker, попроще и могут быть подготовлены и запущены в считанные секунды. Контейнеры LXC особенно хорошо подходят для обкатки новых технологий и безопасного создания стеков программного обеспечения ОС. В настоящее время Docker – это технология, управляющая бесчисленными динамическими, интегрированными парками контейнеров в рамках обширной архитектуры микросервисов. (Я немного больше расскажу о микросервисах в главе 9.)

2.2. Работа с VirtualBox

Вы можете многое сделать с помощью программного обеспечения с открытым исходным кодом VirtualBox компании Oracle. Вы можете установить его на любой ОС (включая Windows), работающей на любом портативном компьютере или ноутбуке, или использовать как ВМ практически для любой ОС.

Установка VirtualBox в среде Windows

Хотите попробовать все это на ПК с Windows? Зайдите на сайт VirtualBox (www.virtualbox.org/wiki/Downloads) и загрузите исполняемый архив. Щелкните кнопкой мыши на файле, который скачали, и выполните несколько шагов по настройке (можно оставить значения по умолчанию). Наконец, программа установки спросит вас, можно ли сбросить сетевые интерфейсы и хотите ли вы установить VirtualBox. Ответьте утвердительно.

VirtualBox предоставляет среду, в которой вы можете запустить столько виртуальных компьютеров, сколько в состоянии обрабатывать ваша физическая система. И это особенно полезный инструмент для безопасного тестирования и получения новых навыков администрирования, что и является нашей основной целью. Но, прежде чем это произойдет, вам нужно знать, как работает загрузка и установка программного обеспечения в Linux.

2.2.1. Работа с менеджерами пакетов Linux

Успешно установить VirtualBox на машине под управлением Ubuntu просто. Для этого потребуются две команды:

```
# apt update  
# apt install virtualbox
```

ПРИМЕЧАНИЕ

Как вы помните, приглашение # означает, что для этой команды требуются права администратора, которые обычно можно получить, предварительно указав команду с помощью sudo.

Что происходит в нашем примере? Все вращается вокруг менеджера пакетов программного обеспечения, который называется Advanced Package Tool (APT, более известный как apt). В мире Linux менеджеры пакетов подключают компьютеры к огромным онлайн-хранилищам тысяч программных приложений, большинство из которых являются бесплатными или имеют открытый исходный код. У менеджера, который по умолчанию устанавливается с Linux, есть несколько заданий.

- Поддерживать локальный индекс для отслеживания репозиториев и их содержимого.
- Отслеживать состояние всего программного обеспечения, установленного на вашей локальной машине.
- Гарантировать, что все доступные обновления применяются к установленному программному обеспечению.
- Гарантировать, что программные зависимости (другие программные пакеты или параметры конфигурации, требуемые пакетом, который вы устанавливаете) соблюдаются для новых приложений перед их установкой.
- Поддерживать установку и удаление пакетов программного обеспечения.

На рис. 2.5 показаны некоторые элементы текущих отношений между онлайн-хранилищем программного обеспечения и менеджером пакетов, работающим на компьютере с Linux.

Система работает отлично, и по различным причинам за пределами мира Linux ничего подобного не наблюдается. Дело в том, что используемый вами менеджер будет зависеть от вашего конкретного дистрибутива Linux. В общем, если ваш дистрибутив относится к семейству Debian/Ubuntu, то вам стоит использовать APT. Члены семейства Red Hat будут использовать менеджер RPM и Yum (или его новую замену DNF). В табл. 2.1 перечислены варианты операционных систем для каждого из менеджеров.

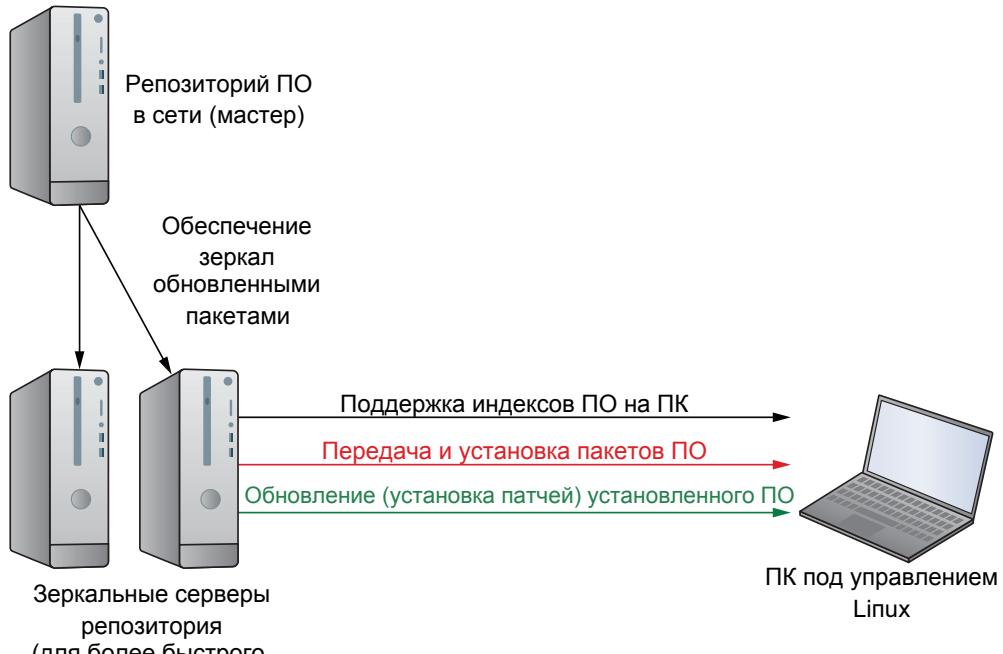


Рис. 2.5. Отношения между основными репозиториями программного обеспечения, зеркальными серверами загрузки и Linux, запущенной на компьютере конечного пользователя

Таблица 2.1. Менеджеры пакетов и дистрибутивы

Менеджер пакетов	Дистрибутив
APT	Debian
	Ubuntu
	Mint
	Kali Linux
RPM	Red Hat Enterprise Linux
	CentOS
	Fedora
YaST	SUSE Linux
	OpenSUSE

Помимо использования менеджера пакетов для установки программного обеспечения из удаленных репозиториев, вам может потребоваться загрузить программное обеспечение с сайта. Часто можно найти пакеты, которые были отформатированы их разработчиками для работы с APT или Yum после установки из командной строки с помощью бэкенд-утилит. Например, вы хотите использовать Skype. Перейдя

на страницу загрузки (рис. 2.6), вы сможете загрузить файл DEB или RPM пакета Skype для Linux. Выбирайте DEB, если используете дистрибутив на основе Debian и APT, или RPM, если работаете в системе RedHat с менеджером Yum.

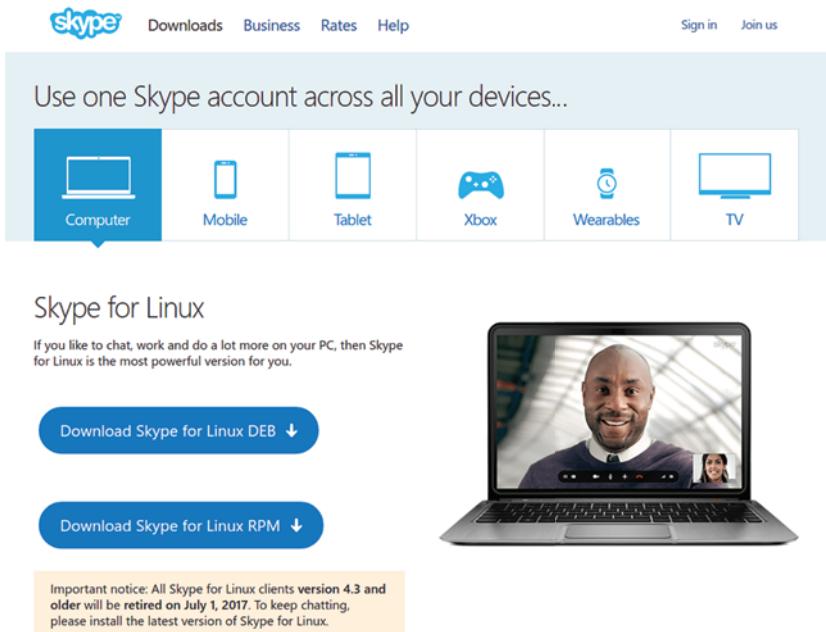


Рис. 2.6. Страница загрузки Skype для Linux. Обратите внимание на разные ссылки для менеджеров пакетов Debian (APT) и RPM (Yum)

Работа с менеджером пакетов Debian

Как только вы загрузили файл, можно установить его из оболочки командной строки, используя инструмент `dpkg`. Укажите флаг `-i` (для установки). Вам нужно убедиться, что вы запускаете команду `dpkg` из каталога, в котором находится файл `skypeforlinux-64`. В этом примере предполагается, что вы сохранили пакет в каталоге `Downloads` вашей учетной записи пользователя:

```
$ cd /home/<username>/Downloads
# dpkg -i skypeforlinux-64.deb
```

Команда `dpkg` должна позаботиться обо всех программных зависимостях. Но если этого не случилось, ее вывод обычно содержит достаточно информации, чтобы понять, что происходит.

Что значит «-64»

Linux, как и другие операционные системы для процессорной архитектуры x86, выпускается как в 64-разрядной, так и в 32-разрядной версии. Подавляющее большинство компьютеров, произведенных и проданных за последнее десятилетие, используют более быстрые 64-разрядные процессоры. Поскольку все еще существует частично устаревшее и ориентированное на такую разработку оборудование, вам иногда потребуется запускать 32-разрядную версию и вы захотите, чтобы установленное вами программное обеспечение работало с ней.

Вы можете проверить разрядность процессора самостоятельно, запустив команду `arch` в оболочке командной строки. Вряд ли вы другим образом узнаете, что работаете на старом оборудовании (это Linux делает особенно хорошо).

Установка VirtualBox для менеджера пакетов RPM

Немного раньше я упоминал короткие команды `apt update` и `apt install virtualbox`. Что они сделали? Для пояснения я установлю то же самое программное обеспечение VirtualBox на машину, на которой запущен дистрибутив Fedora Linux. Поскольку я буду использовать менеджер пакетов DNF от Red Hat, потребуется несколько дополнительных действий, и это хорошо, потому что так мы сможем увидеть, как выполняется этот процесс. Сам процесс немного сложнее, поэтому в табл. 2.2 перечислены нужные шаги.

Таблица 2.2. Процесс установки VirtualBox на Fedora

Задание	Команда
Добавить репозиторий	<code>wget http://download.virtualbox.org/virtualbox/rpm/fedora/virtualbox.repo</code>
Обновить индекс	<code>dnf update</code>
Установить зависимости	<code>dnf install patch kernel-devel dkms</code>
Установить пакет	<code>dnf install VirtualBox-5.1</code>

ПРИМЕЧАНИЕ

Эти шаги были протестированы в версии Fedora 25 и отлично иллюстрируют процесс управления пакетами. Однако все это может работать проще в более поздних релизах Fedora.

Возвращаясь к Ubuntu: APT распознал слово `virtualbox` в команде `install`, потому что пакет `VirtualBox` является частью онлайн-репозитория, с которым APT уже знаком. Однако оказывается, что RedHat и ее дочерние системы (такие как CentOS и Fedora) не настолько «мудры», по крайней мере не при первичной установке, поэтому мне понадобилось вручную добавить репозиторий `virtualbox` в Yum.

Из предыдущей главы вы узнали, что файлы конфигурации стороннего программного обеспечения часто хранятся в структуре каталогов `/etc/`, и в этом отношении yum/DNF ничем не различаются. Информация о репозитории хранится по адресу `/etc/yum.repos.d/`, поэтому вы должны перейти в этот каталог. Там вы используете программу `wget` (обычно устанавливаемую по умолчанию) для загрузки файла `.repo`. Вот как это сделать:

```
$ cd /etc/yum.repos.d/
# wget http://download.virtualbox.org/virtualbox/rpm/fedora/virtualbox.repo
```

Установка программного обеспечения в Linux

Конкретные инструкции по установке программного обеспечения в системе Linux, включая такие подробности, как точный URL-адрес, который я использовал ранее, почти всегда доступны в Интернете. Вы можете найти их либо на собственных сайтах разработчиков программного обеспечения, либо в тематических статьях. Интернет – ваш главный помощник.

При поиске убедитесь, что указали дистрибутив Linux, релиз и архитектуру в поисковой системе. Я нашел информацию о конкретных пакетах, необходимых для этого проекта, в моей любимой поисковой системе, что советую сделать и вам.

Наличие файла `.repo` в нужном каталоге не принесет большой пользы, пока вы не сообщите RPM, что произошло. Это можно сделать, выполнив команду `update`. Она также проверяет индекс локального репозитория по своим онлайн-аналогам, чтобы узнать, есть ли что-то новое, о чем вам следует знать. Независимо от того, каким менеджером вы пользуетесь, всегда полезно обновить информацию о репозитории перед установкой нового программного обеспечения:

```
# dnf update
Importing GPG key 0x98AB5139: ←
  Userid      : "Oracle Corporation (VirtualBox archive signing key)
  <info@virtualbox.org>" ←
  Fingerprint: 7B0F AB3A 1389 0743 5925 D9C9 5442 2A4B 98AB 5139 ←
  From        : https://www.virtualbox.org/download/oracle_vbox.asc ←
Is this ok [y/N]: y
Fedora 25 - x86_64 - VirtualBox          120 kB/s | 33 kB   00:00
Dependencies resolved.
Nothing to do.
Complete!
```

Всегда транзакции с репозиториями шифруются с использованием ключей GPG

Ссылки VirtualBox говорят о том, что я использую этот хост Fedora как виртуальную машину в VirtualBox

Следующий шаг – установка всех программных зависимостей, которые VirtualBox должен будет правильно использовать. *Зависимость* – это программное обеспечение, которое должно быть уже установлено на вашем компьютере для работы нового пакета. Возвращаясь к Ubuntu, следует сказать, что APT негласно заботится о соблюдении зависимостей; Yum также решает многие фоновые задачи. Но когда возникает необходимость в настройке зависимостей вручную, нужную справочную информацию легко найти в Интернете на сайтах, которые обсуждались ранее. Вот сокращенный вариант того, как может выглядеть процесс установки зависимостей:

```
# dnf install patch kernel-devel dkms
Last metadata expiration check: 0:43:23 ago on Tue Jun 13 12:56:16 2017.
[...]
Dependencies resolved.
```

```
=====
Package      Arch    Version           Repository  Size
=====
Installing:
dkms          noarch  2.3-5.20170523git8c3065c.fc25   updates      81 k
kernel-devel  x86_64  4.11.3-202.fc25            updates      11 M
patch         x86_64  2.7.5-3.fc24             fedora     125 k
Transaction Summary
=====
Install 3 Packages
Total download size: 12 M
Installed size: 43 M
Is this ok [y/N]: y
↓
Подтвердите операцию,
введя у перед запуском
Downloading Packages:
(1/3): dkms-2.3-5.20170523git8c3065c.fc25.noarc 382 kB/s | 81 kB   00:00
(2/3): patch-2.7.5-3.fc24.x86_64.rpm            341 kB/s | 125 kB  00:00
(3/3): kernel-devel-4.11.3-202.fc25.x86_64.rpm  2.4 MB/s | 11 MB   00:04
Total                                         1.8 MB/s | 12 MB   00:06
[...]
Running transaction
Installing : kernel-devel-4.11.3-202.fc25.x86_64          1/3
Installing : dkms-2.3-5.20170523git8c3065c.fc25.noarch  2/3
Installing : patch-2.7.5-3.fc24.x86_64                3/3
Verifying  : patch-2.7.5-3.fc24.x86_64                1/3
Verifying  : kernel-devel-4.11.3-202.fc25.x86_64      2/3
Verifying  : dkms-2.3-5.20170523git8c3065c.fc25.noarch 3/3
Installed: ↓
Краткий обзор успешной операции
dkms.noarch 2.3-5.20170523git8c.fc25 kernel-devel.x86_64 4.11.3-202.fc25
patch.x86_64 2.7.5-3.fc24
Complete!
```

Это лишь малая часть всего процесса установки, но вы наконец готовы установить VirtualBox на свою машину под управлением операционной системы RedHat, CentOS или Fedora. Номер версии, который указан в этом примере, взят

из онлайн-руководства, использованного ранее. Естественно, к тому времени, когда вы попробуете повторить мои действия, версия может стать уже далеко не 5.1.

DNF явно устраивают ранее установленные зависимости

```
# dnf install VirtualBox-5.1
Last metadata expiration check: 0:00:31 ago on Tue Jun 13 13:43:31 2017.
```

→ Dependencies resolved.

Package	Arch	Version	Repository	Size
Installing:				
SDL	x86_64	1.2.15-21.fc24	fedora	213 k
VirtualBox-5.1	x86_64	5.1.22_115126_fedora25-1	virtualbox	68 M
python-libs	x86_64	2.7.13-2.fc25	updates	6.2 M
qt5-qtx11extras	x86_64	5.7.1-2.fc25	updates	30 k
Transaction Summary				
Install 4 Packages				
[...]	Возвращает список всех пакетов, которые			
Is this ok [y/N]: y	будут установлены во время этой транзакции			
[...]				
Creating group 'vboxusers'. VM users must be member				
↳ of that group!	Обратите внимание, что процесс создал новую			
[...]	системную группу. Я расскажу о группах в главе 9			
Installed:				
SDL.x86_64 1.2.15-21.fc24				
VirtualBox-5.1.x86_64 5.1.22_115126_fedora25-1				
python-libs.x86_64 2.7.13-2.fc25				
qt5-qtx11extras.x86_64 5.7.1-2.fc25				
Complete!				

Дополнения VirtualBox

Вы должны знать, что Oracle предоставляет пакет расширений для VirtualBox. Этот пакет добавляет такие возможности, как поддержка USB-устройств и шифрование диска, а также предлагает некоторые альтернативы существующим параметрам загрузки. Не забывайте об этих инструментах на тот случай, если у вас возникнут трудности после установки того или иного стандартного пакета.

Вы также можете добавить дополнительную интеграцию файловой системы и устройств между виртуальными машинами VirtualBox и их хостом через образ CD-ROM Vbox Guest Additions. Так вы получите возможность пользоваться общим буфером обмена и выполнять перетаскивание. Если дополнения Vbox еще недоступны на вашем хосте, установите пакет расширений в Ubuntu с помощью следующей команды:

```
sudo apt install virtualbox-guest-additions-iso
```

Затем добавьте CD-ROM в качестве виртуального оптического привода на работающую виртуальную машину. Ищите в Интернете документацию, касающуюся любых дополнительных пакетов, которые могут понадобиться для работы в вашей операционной системе.

Прежде чем вы по-настоящему перейдете к использованию инструментов виртуализации, таких как VirtualBox, я должен оставить вам хотя бы одну или две подсказки для отслеживания других пакетов репозитория, которые могут вам понадобиться. АРТ-системы позволяют напрямую искать доступные пакеты с помощью команды `apt search`.

В этом примере выполняется поиск пакетов, которые могут помочь вам контролировать состояние системы, а затем используется команда `apt show` для отображения полной информации о пакете:

```
$ apt search sensors  
$ apt show lm-sensors
```

Если программа `aptitude` установлена, она выступает частично графической оболочкой, в которой вы можете отслеживать как доступные, так и уже установленные пакеты, а также управлять ими. Если вы не можете жить без мыши, обратите внимание на `Synaptic` — полноценный менеджер пакетов с графическим интерфейсом для ПК. В `Yum` также полностью реализованы инструменты поиска:

```
$ yum search sensors  
$ yum info lm_sensors
```

2.2.2. Определение виртуальной машины (ВМ)

Я не уверен, что вы когда-нибудь собирали настоящий компьютер из комплектующих, но это могло бы помочь. Настройка новой виртуальной машины в `VirtualBox` выполняется примерно так же. Единственное существенное отличие состоит в том, что вместо того, чтобы, зажав между зубами фонарик, вручную вставлять планки ОЗУ и жесткий диск в компьютер, вы щелчком кнопкой мыши предоставляем возможность `VirtualBox` определять «аппаратные» характеристики виртуальной машины.

Выбрав пункт `New` (Создать) в интерфейсе `VirtualBox`, вы сообщите виртуальной машине, что готовы дать описательное имя. Как видно на рис. 2.7, программное обеспечение должно правильно автоматически заполнить поля `Type` (Тип) и `Version` (Версия). Тип и версия, которые вы здесь указываете, относятся не к настоящей, хостовой операционной системе, а используются для соответствующих настроек эмулируемого оборудования.

В следующем окне выделите оперативную память для своей виртуальной машины. Если не планируете что-то особенно требовательное, как, например, размещение ряда контейнеров или использование загруженного веб-сервера, то подойдет размер по умолчанию (768 Мбайт). Вы, конечно, можете выделить больше оперативной памяти, если это необходимо, но не забудьте оставить достаточно для вашего хост-компьютера и любых других виртуальных машин, которые уже могут в ней быть. Если на вашем хосте только 4 Гбайт физической памяти, вы, вероятно, не захотите отдавать половину этого объема своей виртуальной машине.

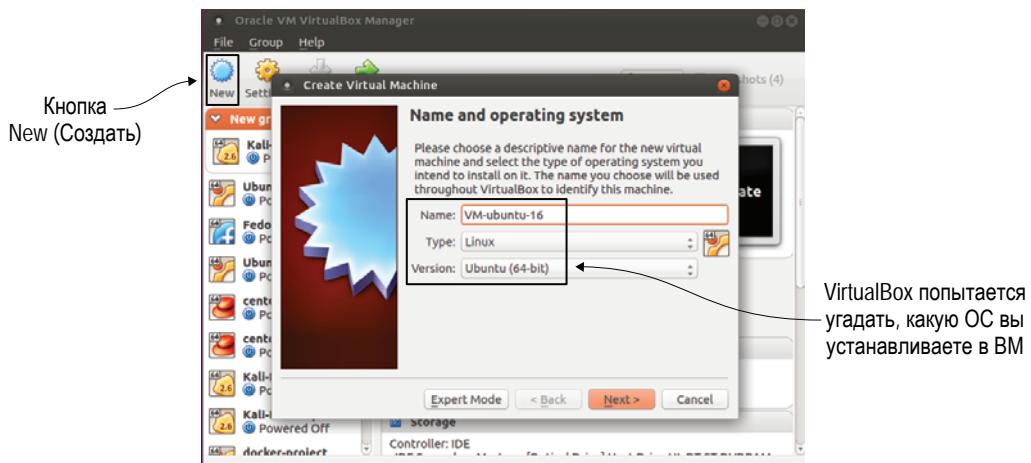


Рис. 2.7. Диалоговое окно Create Virtual Machine (Создание виртуальной машины): VirtualBox попытается угадать вашу ОС и ее версию, чтобы позже предложить правильный выбор по умолчанию

Помните об этих ограничениях, если решите запустить несколько виртуальных машин одновременно, что будет полезно для парочки проектов, о которых вы узнаете позже в этой книге. Даже если каждая виртуальная машина использует только объем памяти по умолчанию, две или три запущенные одновременно могут использовать всю оперативную память, необходимую для нормальной работы хоста.

Теперь мастер установки VirtualBox запрашивает, хотите вы создать новый виртуальный диск для своей виртуальной машины или использовать уже существующий (рис. 2.8). Что за компьютер без жесткого диска? Возможно, вы захотите разделить один диск между двумя виртуальными машинами, но в этом упражнении, я полагаю, вы предпочтете начать с нуля. Установите переключатель в положение *Create a virtual hard disk now* (Создать виртуальный жесткий диск сейчас).

В следующем окне (рис. 2.9) вы можете выбрать формат файла на жестком диске для вашей ВМ. Если не планируете в конечном итоге экспорттировать диск для использования с какой-либо другой средой виртуализации, то формат VirtualBox Disk Image (VDI), указанный по умолчанию, будет нормально работать.

Я никогда не сожалел о том, что выбрал параметр *Dynamically allocated* (Динамически распределяемый) (рис. 2.10), чтобы указать, как виртуальный диск будет занимать пространство на хосте. Здесь *динамическое распределение* означает, что пространство на диске хоста будет выделяться виртуальной машине только по мере необходимости. Если виртуальная машина незначительно использует диск, то на хосте будет выделено меньше места.

Диск фиксированного размера, с другой стороны, сразу займет весь объем, независимо от того, сколько он фактически использует. Единственное преимущество фиксированного размера – производительность. Обычно я использую виртуальные машины VirtualBox только для тестирования и экспериментов, поэтому я обхожусь динамическим диском.



Рис. 2.8. Окно настройки жесткого диска. Обратите внимание на то, что в данном случае переключатель установлен в положение *Use an existing virtual hard disk file* (Использовать существующий файл виртуального жесткого диска)



Рис. 2.9. Виртуальные жесткие диски могут быть созданы с применением нескольких форматов. VDI подходит для виртуальных машин, которые будут использоваться только в VirtualBox



Рис. 2.10. Динамические виртуальные диски будут занимать столько места на устройствах хоста, сколько им нужно

Поскольку эмулятор знает, что вы используете Linux, и поскольку Linux эффективно задействует пространство для хранения, VirtualBox, вероятно, предложит вам только 8 Гбайт общего размера диска (рис. 2.11). Если у вас нет глобальных планов насчет ВМ (например, вы не собираетесь выполнять трудоемкие операции с крупными базами данных), этого, вероятно, будет достаточно. С другой стороны, если бы вы выбрали Windows в качестве виртуальной ОС, то по умолчанию было бы предложено 25 Гбайт — и на то есть веская причина: Windows не стесняется в требованиях к ресурсам. Это отличная иллюстрация того, почему Linux так хорошо подходит для виртуальных сред.

ПРИМЕЧАНИЕ

При желании вы также можете изменить название и расположение, которое VirtualBox будет использовать для вашего диска, в окне File location and size (Местоположение и размер файла).

Когда вы закончите, нажмите Create (Создать), и новая виртуальная машина появится в списке виртуальных машин в левой части менеджера VirtualBox. Но вы еще не закончили: вы создали только машину. Теперь вам понадобится ОС, чтобы она начала подавать признаки жизни.

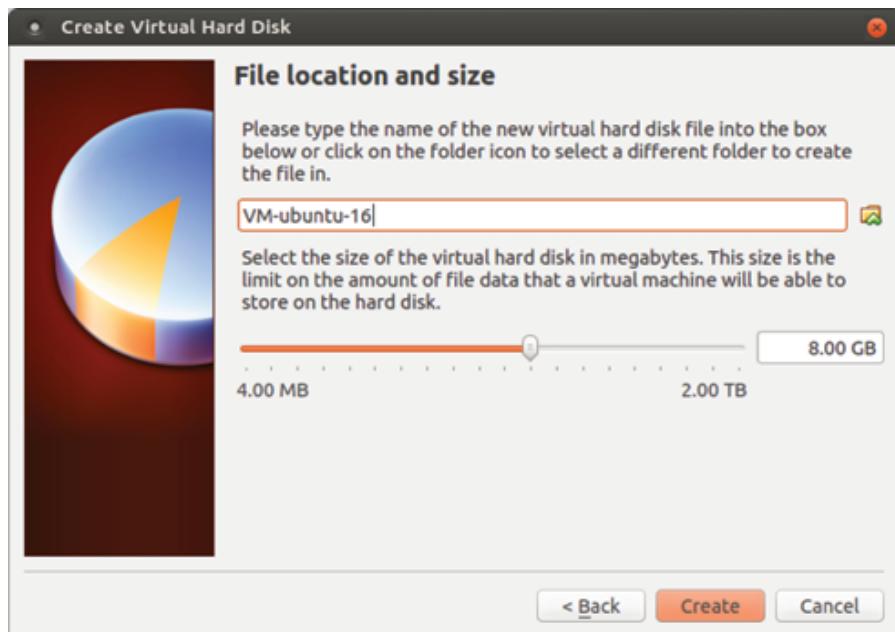


Рис. 2.11. При необходимости размер вашего виртуального диска можно расширить до 2 Тбайт или даже всего свободного пространства на хосте

2.2.3. Установка операционной системы

Теперь, когда вы определили профиль виртуального оборудования своей новой виртуальной машины, вам нужно сделать еще кое-что.

1. Скачать файл (в формате ISO) образа дистрибутива Linux, который вы хотите использовать.
2. Загрузить новую виртуальную машину, указав виртуальный привод DVD, в который «вставлен» скачанный ISO-образ.
3. Пройти стандартный процесс установки ОС.
4. Загрузить виртуальную машину и запустить установленную ранее ОС.

После того как вы выберете дистрибутив, вам нужно скачать файл в формате ISO, содержащий файлы ОС и программу установки. Подходящий файл обычно можно найти в Интернете по названию дистрибутива, добавив слово **скачать**. В случае с Ubuntu вы также можете открыть страницу <https://ubuntu.com/> и перейти на вкладку **Downloads** (Загрузки), как показано на рис. 2.12. Обратите внимание, что доступны различные типы Ubuntu. Если вы собираетесь использовать эту виртуальную машину для задач администрирования, то небольшая и быстрая версия **Server**, вероятно, будет лучшим выбором, чем **Desktop**.

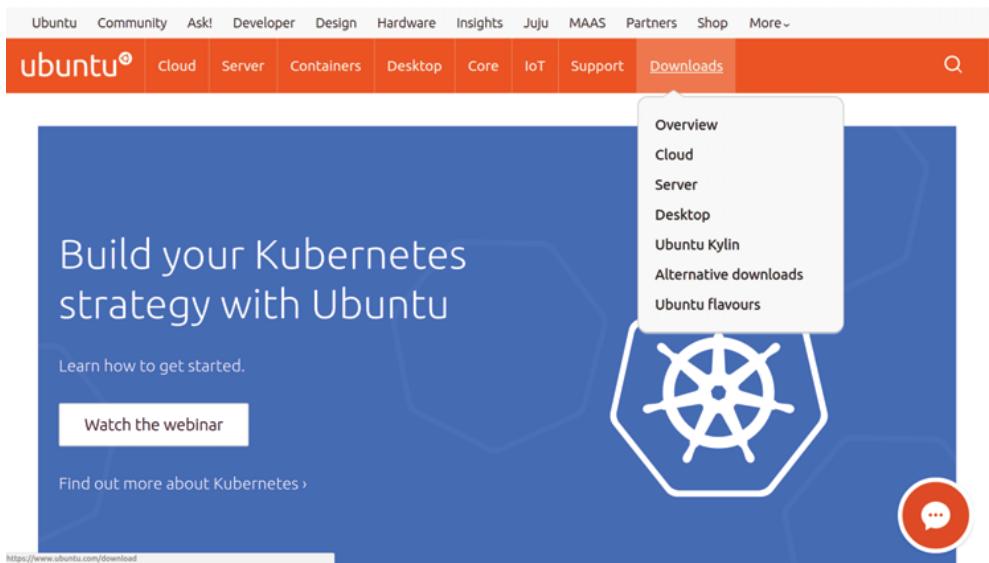


Рис. 2.12. Меню Downloads (Загрузки) на сайте [ubuntu.com](https://www.ubuntu.com/). Обратите внимание на набор версий, которые предлагает Ubuntu.

Файлы большого размера при скачивании иногда могут повреждаться. Если хотя бы один байт в вашем ISO-файле был изменен, есть вероятность, что установка не пройдет должным образом. Вы наверняка не хотите тратить время и силы на обнаружение проблемы со скачиванием в процессе установки, поэтому советую сразу посчитать контрольную сумму (или *хеш*) загруженного файла ISO, чтобы подтвердить, что скачанный файл идентичен оригиналу. Для этого вам необходимо получить соответствующую контрольную сумму SHA или MD5, которая представляет собой длинную строку, выглядящую примерно так:

4375b73e3a1aa305a36320ffd7484682922262b3

Вы можете получить эту строку там же, откуда скачали свой файл ISO. В случае с Ubuntu нужно перейти на веб-страницу по адресу <http://releases.ubuntu.com/>, выбрать каталог, соответствующий скачанной версии, а затем щелкнуть на одной из ссылок на контрольную сумму (например, [SHA1SUMS](#)). Вы должны сравнить соответствующую строку на данной странице с результатами команды, запущенной в каталоге, куда вы скачали ISO-файл. Это выглядит следующим образом:

```
$ shasum ubuntu-16.04.2-server-amd64.iso
```

Если они совпадают, все хорошо. Если не совпадают (и вы дважды проверили, чтобы убедиться, что используете правильную версию), то вам, возможно, придется заново загрузить файл ISO.

ПРИМЕЧАНИЕ

Имейте в виду, что хеши бывают разных видов. В течение многих лет алгоритм MD5SUM был доминирующим, но SHA256 (с более длинными 256-битными хешами) набирает популярность. На практике для больших файлов образов ОС оба варианта, вероятно, равнозначны.

Как только ваш файл ISO будет готов, вернитесь в VirtualBox. Только что созданная вами ВМ должна быть выделена на левой панели. Нажмите зеленую кнопку **Start** (Пуск) в верхней части приложения. Вам будет предложено выбрать файл ISO в вашей файловой системе для использования в качестве виртуального DVD-привода. Естественно, выбирайте тот файл, который только что скачали. Новая ВМ прочитает этот DVD и запустит установку ОС.

В основном процесс установки должен пройти нормально; однако поиск решений для каждой из множества возможных мелких проблем потребует еще пары полных глав. Если у вас возникли сложности, можете обратиться к документации и руководствам, которые доступны для любого дистрибутива Linux, или задать свой вопрос на форуме Manning в ветке книги *Linux in Action* и запросить помочь у сообщества.

Даже если установка прошла успешно, все еще может быть несколько задач, которые нужно выполнить, прежде чем вы сможете нормально загрузиться в свою виртуальную машину. Выделив виртуальную машину, щелкните кнопкой мыши на желтом значке **Settings** (Настройки). Здесь вы можете поэкспериментировать со средой вашей виртуальной машины и настройками оборудования. Например, выбрав раздел **Network** (Сеть), вы можете определить сетевое подключение. Если вы хотите, чтобы у вашей виртуальной машины был полный доступ к Интернету через сетевой интерфейс хост-компьютера, выберите пункт **Bridged Adapter** (Мост) в раскрывающемся списке **Attached to** (Присоединен к), как показано на рис. 2.13, а затем укажите название адаптера вашего хоста.

ПРИМЕЧАНИЕ

Не всегда нужно выбирать вариант **Bridged Adapter** (Мост). Иногда это может представлять угрозу безопасности. Фактически вариант **NAT Network** (Сеть NAT) предлагает более распространенный способ предоставления виртуальной машине доступа в Интернет. Поскольку во многих упражнениях этой книги требуется соединение нескольких виртуальных машин (что сложно реализовать с помощью NAT), я пока продолжу с мостом.

Возможно, вам никогда не понадобится последующая информация, но предупрежден — значит вооружен. В некоторых случаях для правильной загрузки виртуальной машины необходимо извлечь DVD из дисковода так же, как вы делаете при реальной физической установке. Для этого нужно перейти в раздел **Storage** (Хранилище). Щелкните кнопкой мыши на первом диске в списке, а затем на значке **Remove Disk** (Удалить диск) внизу (рис. 2.14). Убедитесь, что случайно не удалили свой жесткий диск!

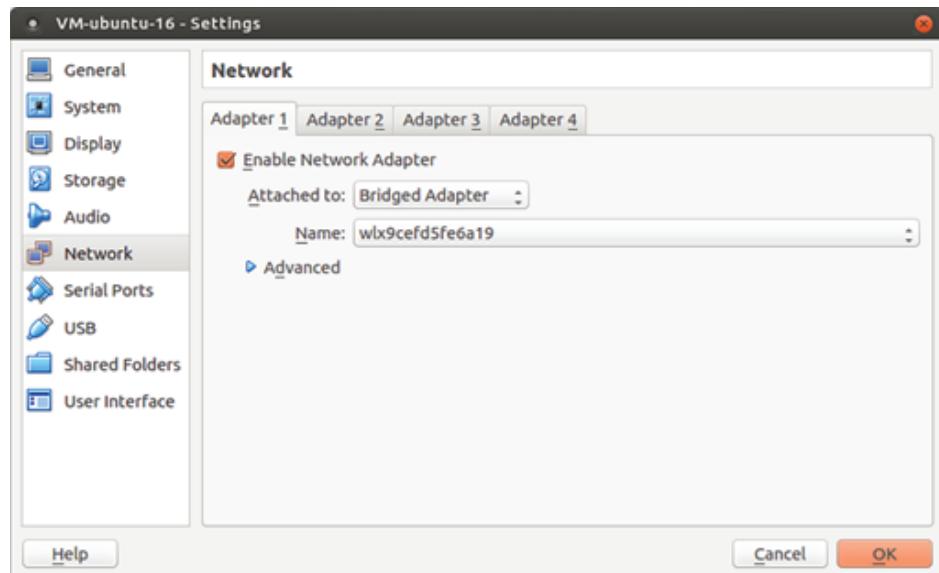


Рис. 2.13. Вкладка Network (Сеть) диалогового окна Settings (Настройки), где вы можете определить, какой тип сетевого интерфейса (или интерфейсов) использовать для своей виртуальной машины

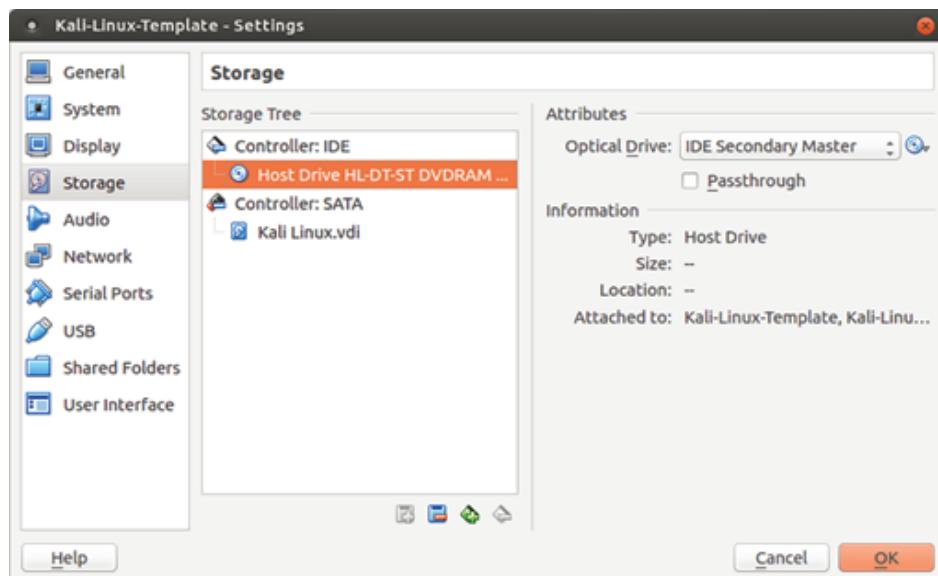


Рис. 2.14. Удалите виртуальный диск, щелкнув правой кнопкой мыши на его ссылке и выбрав пункт Delete (Удалить). Возможно, вам придется сделать это, чтобы убедиться, что виртуальная машина загружается на правильный диск

Возможно, вам иногда потребуется смонтировать DVD (или файл ISO), чтобы VirtualBox мог его распознать. Щелкнув на значке + при выделенной строке Controller: IDE (Контроллер: IDE), вы можете выбрать файл, который будет использоваться в качестве виртуального оптического привода.

2.2.4. Клонирование и совместное использование виртуальной машины VirtualBox

Информация из этого раздела не относится напрямую к теме главы, но кому не нравятся бесплатные бонусы? Я расскажу вам о двух связанных между собой хитростях: как клонировать виртуальные машины VirtualBox, чтобы ускорить запуск новых копий, и как использовать командную строку для совместного применения виртуальных машин в сети.

Клонирование виртуальных машин для быстрого запуска

Одним из наиболее очевидных преимуществ работы с виртуальными машинами является возможность быстрого доступа к свежей, чистой среде ОС. Но если для доступа к этой среде требуется пройти полный процесс установки, то не будет особой выгоды в скорости, пока вы не научитесь клонировать ВМ. Почему бы не сохранить исходную виртуальную машину в чистом состоянии после установки и создавать идентичный клон всякий раз, когда вы хотите выполнить какую-то работу?

Это легко. Посмотрите еще раз на приложение VirtualBox. Выберите (остановленную) виртуальную машину, которую хотите использовать в качестве главной копии, откройте меню Machine (Машина) и выберите команду Clone (Клонировать). Подтвердите название, которое хотите дать своему клону. Затем, после нажатия кнопки Next (Далее), выберите, хотите вы создать Full Clone (Полный клон) (то есть для новой виртуальной машины будут созданы совершенно новые копии файлов) или Linked Clone (Связанный клон) (то есть новая ВМ будет использовать все базовые файлы хозяина, при этом поддерживая ваш новый проект отдельно).

ПРИМЕЧАНИЕ

Если вы выберете вариант Linked Clone (Связанный клон), процесс будет происходить намного быстрее и машина займет гораздо меньше места на жестком диске. Единственным недостатком будет то, что вы не сможете позже перенести этот конкретный клон на другой компьютер. Таков выбор.

Теперь нажмите кнопку Clone (Клонировать), и новая виртуальная машина появится на панели программы. Запустите ее как обычно, а затем войдите в систему, используя те же учетные данные, которые определили на исходной ВМ.

Управление виртуальными машинами из командной строки

VirtualBox поставляется с собственной оболочкой командной строки, которая вызывается с помощью команды `vboxmanage`. Зачем нам оболочка командной строки? Помимо прочего, она позволяет работать на удаленных серверах, что может значительно увеличить возможности потенциальных проектов. Чтобы увидеть, как работает команда `vboxmanage`, используйте инструкцию `list vms` для отображения всех виртуальных машин, доступных в настоящее время в вашей системе:

```
$ vboxmanage list vms
"Ubuntu-16.04-template" {c00d3b2b-6c77-4919-85e2-6f6f28c63d56}
"centos-7-template" {e2613f6d-1d0d-489c-8d9f-21a36b2ed6e7}
"Kali-Linux-template" {b7a3aea2-0cfb-4763-9ca9-096f587b2b20}
"website-project" {2387a5ab-a65e-4a1d-8e2c-25ee81bc7203}
"Ubuntu-16-1xd" {62bb89f8-7b45-4df6-a8ea-3d4265dfcc2f}
```

Команда `vboxmanage clonevm` выполнит такое же клонирование, как описанное раньше для графического интерфейса. Здесь я делаю клон шаблона виртуальной машины Kali Linux, называя копию `newkali`:

```
$ vboxmanage clonevm --register Kali-Linux-template --name newkali
```

Вы можете убедиться, что это сработало, еще раз запустив команду `vboxmanage list vms`.

Это подходит, пока мне нужно использовать новую виртуальную машину только здесь, на моем локальном компьютере. Но предположим, что я хочу, чтобы другие члены моей команды могли запустить точную копию этой виртуальной машины, возможно, для того, чтобы протестировать то, над чем я работал. Для этого мне нужно преобразовать виртуальную машину в какой-то стандартизованный файловый формат. Вот как я могу экспорттировать локальную виртуальную машину в файл, используя открытый формат виртуализации (Open Virtualization Format):

```
$ vboxmanage export website-project -o website.ova
0%...10%...20%...30%...40%...50%...60%...70%...80%...90%...100%
Successfully exported 1 machine(s).
```

Флаг `-o` указывает выходное имя файла, в данном случае `website.ova`

Теперь вам необходимо скопировать файл OVA на компьютер вашего коллеги. Имейте в виду, что файл по всем параметрам не будет маленьким и удобным. Если вам не хватает пропускной способности сети для передачи нескольких гигабайт, рассмотрите возможность его копирования через USB-накопитель. Но если вы выберете копирование через сеть, лучшим инструментом для работы будет Secure Copy (`scp`). Вот как это может работать:

```
$ scp website.ova username@192.168.0.34:/home/username
```

Если весь этот код команды `scp` кажется немного странным, не беспокойтесь: помошь уже в пути. Команда `scp` будет полностью рассмотрена в главе 3, в части, посвященной OpenSSH. Тем временем команда `scp` будет работать только в том случае, если оболочка OpenSSH установлена на обоих компьютерах, у вас есть

авторизованный доступ к учетной записи на удаленном компьютере и она доступна с вашего локального компьютера.

Как только передача завершена, остается только импортировать с удаленного компьютера виртуальную машину в VirtualBox этой машины. Команда проста:

```
$ vboxmanage import website.ova
```

Убедитесь, что операция импорта успешно завершена, для чего выполните команду `list vms`. И попробуйте запустить виртуальную машину:

```
$ vboxmanage list vms  
"website" {30ec7f7d-912b-40a9-8cc1-f9283f4edc61}
```

Если вам не нужен удаленный доступ, вы также можете расшарить виртуальную машину из графического интерфейса. Выделив в VirtualBox ВМ, которой вы хотите поделиться, выберите пункт меню **File ▶ Export Appliance** (Файл ▶ Экспортировать оборудование).

2.3. Работа с контейнерами Linux (LXC)

VirtualBox отлично подходит для выполнения операций, когда требуется доступ к ядру Linux (например, как если бы вы использовали функции безопасности наподобие SELinux, как вы увидите в главе 9), когда нужны сеансы рабочего стола с графическим интерфейсом пользователя или во время тестирования таких операционных систем, как Windows. Но если вам нужен быстрый доступ к чистой среде Linux и вы не ищете какой-либо специальной версии релиза, то лучше всего подойдет LXC.

ПРИМЕЧАНИЕ

Как и любая сложная система, LXC может не поддерживать некоторые аппаратные архитектуры. Если у вас возникли проблемы с запуском контейнера, вероятно, это проблема с совместимостью. В Интернете, как всегда, можно получить более точную информацию.

Насколько быстро работают контейнеры LXC? Вы скоро увидите сами. Но, поскольку они умело разделяют многие системные ресурсы как с хостом, так и с другими контейнерами, они работают как настоящие отдельные серверы, используя только минимальное место для хранения и память.

2.3.1. Начало работы с LXC

Установить LXC на рабочую станцию Ubuntu? Легко, вот рецепт:

```
# apt update  
# apt install lxc
```

А как насчет CentOS? Все в общем понятно, но нужно проделать немного большую работу. Как бы там ни было, попробуйте CentOS, но я не гарантирую

успеха. Сначала нужно добавить новый репозиторий Extra Packages for Enterprise Linux (EPEL), а затем установить LXC вместе с некоторыми зависимостями:

```
# yum install epel-release
# yum install lxc lxc-templates libcap-devel \
libcgroup busybox wget bridge-utils lxc-extra libvirt
```

Символ обратной косой черты (\) может использоваться для удобного разбиения длинной команды на несколько строк в оболочке командной строки

Теперь все готово и вы можете приступить к делу. Базовые принципы работы с LXC на самом деле довольно просты. Я собираюсь показать вам три или четыре команды, которые вам понадобятся, чтобы все это заработало, а затем открою один секрет, который поразит вас, как только вы поймете, как организована LXC.

2.3.2. Создание вашего первого контейнера

Почему бы не продолжить и не создать свой первый контейнер? Ключ -n задает имя, которое вы будете использовать для контейнера, а -t указывает LXC построить контейнер по шаблону Ubuntu:

```
# lxc-create -n myContainer -t ubuntu
```

Процесс создания может занять несколько минут, но вы увидите подробный вывод и, в конце концов, уведомление об успехе

ПРИМЕЧАНИЕ

Возможно, будут ссылки на альтернативный набор команд LXC, связанных с относительно новым диспетчером контейнеров LXD. LXD все еще использует неявно инструменты LXC, но с помощью немного другого интерфейса. Например, с применением LXD предыдущая команда будет выглядеть следующим образом: lxc launch ubuntu:16.04 myContainer. Допустимы оба набора команд.

На самом деле доступно довольно много шаблонов, как вы можете видеть в коде, относящемся к каталогу `/usr/share/lxc/templates/`:

```
$ ls /usr/share/lxc/templates/
lxc-alpine    lxc-centos   lxc-fedora      lxc-oracle    lxc-sshd
lxc-altnlinux lxc-cirros   lxc-gentoo     lxc-plamo    lxc-ubuntu
lxc-archlinux lxc-debian   lxc-openmandriva lxc-slackware lxc-ubuntu-cloud
lxc-busybox   lxc-download lxc-opensuse   lxc-sparclinux
```

ВНИМАНИЕ

Не все эти шаблоны гарантированно работают в исходном состоянии. Некоторые представляются как экспериментальные или находятся в процессе разработки. Использование шаблона Ubuntu на хосте Ubuntu, вероятно, окажется безопасным выбором. Как я уже отмечал, с самого начала набор LXC всегда работал лучше всего на хостах Ubuntu. Ваш опыт может быть другим, если вы имеете дело с иными дистрибутивами.

Если вы решили создать, скажем, контейнер CentOS, то следует записать последние несколько строк вывода, поскольку они содержат информацию о пароле, который вы используете для входа в систему:

```
# lxc-create -n centos_lxc -t centos ←
```

В этом примере я вызвал контейнер centos_lxc и использовал шаблон centos

[...]

The temporary root password is stored in:

'/var/lib/lxc/centos_lxc/tmp_root_pass' ←

Пароль администратора
находится в каталоге,
названном в честь контейнера

Вы войдете в систему, используя логин *администратора* и пароль, содержащийся в файле tmp_root_pass. Если, с другой стороны, ваш контейнер задействует шаблон Ubuntu, то вы будете указывать ubuntu как имя пользователя и пароль. Естественно, если вы планируете применять этот контейнер для чего-либо серьезного, то лучше изменить этот пароль:

```
$ passwd
Changing password for ubuntu.
(current) UNIX password:
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
```

Кстати, это действительно команда `passwd`, а не `password`. Предполагаю, что создатель программы `passwd` не любил печатать. Теперь введите `lxc-ls --fancy` для проверки состояния вашего контейнера:

```
# lxc-ls --fancy
NAME      STATE   AUTOSTART GROUPS IPV4    IPV6
myContainer STOPPED 0          -     -     -
```

Контейнер создан, и теперь, пожалуй, пора начать. Как и прежде, `-n` указывает имя контейнера, который вы хотите запустить; `-d` означает «отсоединение», то есть вы не хотите автоматически запускать интерактивную сессию при запуске контейнера. В интерактивных сессиях нет ничего плохого. Но в этом случае выполнение команды `lxc-start` без `-d` будет означать, что единственный способ выйти из сессии — закрыть контейнер, а это может быть не совсем то, чего вы хотите:

```
# lxc-start -d -n myContainer
```

Список ваших контейнеров теперь должен выглядеть подобно этому:

```
# lxc-ls --fancy
NAME      STATE   AUTOSTART GROUPS IPV4    IPV6
myContainer RUNNING 0          -     10.0.3.142 -
```

Состояние контейнера
теперь RUNNING

На этот раз контейнер работает и получил IP-адрес (10.0.3.142). Вы можете использовать данный адрес для входа в систему в сеансе защищенной оболочки,

но сначала прочитайте главу 3. На данный момент вы можете запустить сеанс оболочки администратора в работающем контейнере с помощью команды `lxc-attach`:

```
# lxc-attach -n myContainer
root@myContainer:/#
```

Обратите внимание на информацию
в новой командной строке

Вы можете потратить пару минут на осмотр окружения. Например, `ip addr` выведет список сетевых интерфейсов контейнера. Здесь интерфейсу `eth0` был присвоен IP-адрес `10.0.3.142`, который соответствует значению IPV4, полученному при выполнении `lxc-ls --fancy` ранее:

```
root@myContainer:/# ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue
  ↳ state UNKNOWN group default qlen 1
    link/loopback 00:00:00:00:00 brd 00:00:00:00:00:00
      inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
      inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
10: eth0@if11: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue
  ↳ state UP group default qlen 1000
    link/ether 00:16:3e:ab:11:a5 brd ff:ff:ff:ff:ff:ff link-netnsid 0
      inet 10.0.3.142/24 brd 10.0.3.255 scope global eth0
        valid_lft forever preferred_lft forever
      inet6 fe80::216:3eff:feab:11a5/64 scope link
        valid_lft forever preferred_lft forever
```

В данном случае `eth0` —
это обозначение
основного сетевого
интерфейса контейнера

IP-адрес контейнера (10.0.3.142)
и маска сети CIDR (/24)

Когда вы закончите осматривать свой новый контейнер, можете либо набрать `exit`, чтобы выйти, оставив контейнер работающим:

```
root@myContainer:/# exit
exit
```

либо завершить его работу, используя команду `shutdown -h now`. Но, прежде чем сделать это, выясним, насколько быстрыми являются контейнеры LXC. Флаг `-h`, который я добавил к команде `shutdown`, означает *остановку*. Если бы я вместо этого указал `r`, а не полностью завершил работу, то контейнер перезагрузился бы. Запустим `reboot`, а затем попробуем снова войти в систему, чтобы узнать, сколько времени потребуется, чтобы контейнер «поднялся на ноги»:

```
root@myContainer:/# shutdown -r now
# lxc-attach -n myContainer
```

Как все прошло? Могу поспорить, что к тому времени, когда вы повторно введете команду `lxc-attach`, контейнер `myContainer` уже проснется и будет готов к работе. Знаете ли вы, что нажатие клавиши `↑` в Bash приведет к вызову предыдущей команды? Использование этого трюка ускорит запрос на вход в систему. В моем случае заметной задержки не было. Контейнер закрывается и полностью перезагружается менее чем за две секунды!

ПРИМЕЧАНИЕ

Контейнеры LXC также удобны в отношении использования системных ресурсов. В отличие от моего опыта работы с виртуальными машинами VirtualBox, когда запуск одновременно трех машин уже начинает серьезно влиять на производительность моей рабочей станции с 8 Гбайт ОЗУ, я могу запускать все виды контейнеров LXC без замедления работы.

Что вы говорите? Как насчет того секрета, который я обещал раскрыть? Отлично. Я вижу, вы внимательны. Итак, вернемся к оболочке командной строки на хост-компьютере (в отличие от контейнера): вам нужно будет войти в оболочку администратора с помощью команды `sudo su`. С этого момента, пока вы не введете команду `exit`, вы будете работать в `sudo`:

```
$ sudo su  
[sudo] password for username:  
#
```

Теперь перейдите в каталог `/var/lib/lxc/` и просмотрите его содержимое. Вы должны увидеть каталог с именем вашего контейнера. Если у вас есть другие контейнеры в системе, у них также будут собственные каталоги:

```
# cd /var/lib/lxc  
# ls  
myContainer
```

Перейдите в каталог контейнера и просмотрите его содержимое. Там будет файл с именем `config` и каталог с именем `rootfs` (`fs` означает файловую систему):

```
# cd myContainer  
# ls  
config rootfs
```

Не стесняйтесь взглянуть на `config`: именно здесь определены основные значения среды для контейнера. Как только вам станет удобнее работать с LXC, вы, вероятно, захотите использовать этот файл для настройки поведения ваших контейнеров. Но вот что представляет собой каталог `rootfs`, который я действительно хотел бы показать вам:

```
# cd rootfs  
# ls  
Bin dev home lib64 mnt proc run srv tmp var  
boot etc lib media opt root sbin sys usr
```

Все эти подкаталоги, которые заполняют `rootfs`, вам знакомы? Все они являются частью стандарта иерархии файловых систем Linux (FHS). Это корневой (`/`) каталог контейнера, но внутри файловой системы хоста. Пока у вас есть права администратора на хосте, вы сможете просматривать эти каталоги и редактировать любые файлы, которые вам нужны, даже если контейнер не работает.

Обладая таким доступом, вы сможете делать все что угодно, но есть одна возможность, которая вполне может однажды спасти вашу (профессиональную) жизнь.

Предположим, вы в контейнере. Теперь ничто не мешает вам перемещаться по файловой системе, исправлять испорченные файлы конфигурации и возвращаться к работе. Вот скажите мне, что это не круто! Но будет еще лучше.

Это правда, что экосистема Docker получила дополнительную функциональность и стала более сложной с тех пор, как несколько лет назад эта технология вышла из-под эгиды LXC. Однако с точки зрения внутреннего устройства Docker все еще построена на базовой структурной парадигме, которую сразу опознают все, кто знаком с LXC. Это означает, что, если у вас нет проблем с самой быстро развивающейся технологией виртуализации десятилетия, у вас уже есть джокер в игре.

Резюме

- Гипервизоры, такие как VirtualBox, предоставляют среду, в которой виртуальные операционные системы могут безопасно получать доступ к аппаратным ресурсам, тогда как облегченные контейнеры совместно используют ядро программного обеспечения своего хоста.
- Менеджеры пакетов Linux, такие как APT и RPM (Yum), наблюдают за установкой и администрированием программного обеспечения из курируемых онлайн-репозиториев, используя регулярно обновляемый индекс, который отражает состояние удаленного репозитория.
- Для запуска виртуальной машины в VirtualBox необходимо определить ее виртуальную аппаратную среду, загрузить образ ОС и установить ОС на свою виртуальную машину.
- Вы можете легко клонировать, совместно использовать и администрировать виртуальные машины VirtualBox из командной строки.
- Контейнеры LXC созданы на базе предопределенных, распространяемых шаблонов.
- Данные LXC хранятся в файловой системе хоста, что упрощает администрирование контейнеров.

Ключевые понятия

- *Виртуализация* — это совместное использование несколькими процессами вычислительных, сетевых ресурсов и ресурсов хранения, что позволяет запускать каждый из них так, как если бы это был отдельный физический компьютер.
- *Гипервизор* — это программное обеспечение, которое работает на хост-компьютере и предоставляет системные ресурсы гостевому уровню, позволяя запускать и администрировать полнофункциональные гостевые виртуальные машины.
- *Контейнер* — это виртуальная машина, которая запускается поверх ядра операционной системы хоста (и совместно использует его). Контейнеры чрезвычайно легко запускать и уничтожать в зависимости от краткосрочной необходимости.