Solving for Technology

Как быстро освоить полезные новые навыки в быстро меняющемся мире технологии

Дэвид Клинтон

Bootstrap IT

Solving for Technology

Как быстро освоить полезные новые навыки в быстро меняющемся мире технологии

Дэвид Клинтон

ISBN: 978-1-387-51766-4

Copyright:

© 2018 David Clinton and Bootstrap IT

Содержание

Глава 1. Управление изменением 5

1.1 Почему ваша карьера нуждается в ориентации на будущее (*future-proofing*) 5

1.2 Что вы должны изучить? 6

1.2.1 Облачные вычисления 7

1.2.2 Web-разработка 7

1.2.3 База данных 8

1.2.4 Big data 9

1.2.5 Docker 9

1.2.6 Blockchain 10

1.2.7 Безопасность и шифрование 10

1.2.8 DevOps 11

Глава 2. Подумай хорошенько 12

2.1 Выявление проблемы, требующей решения 12

2.2 Получение максимальной отдачи от вашего обучения 13

2.3 Оставаться вдохновленным 14

2.3.1 Не бойтесь менять подход к обучению 14

2.3.2 Изучайте по частям 14

2.4 Ведение сбалансированного, здорового образа жизни 14

2.5 Пример из практики 14

Глава 3. Рабочие привычки (богатых и знаменитых) 16

3.1 Дисциплина 16

3.2 Не бойтесь экспериментировать… и ошибаться 16

3.3 Делайте заметки 18

3.3.1 Документирование 18

3.3.2 Тест 18

3.3.3 Автоматизация инфраструктуры с помощью скриптов 19

3.4 Пример из практики 22

Глава 4. Первые шаги 23

4.1 Общая картина 23

4.2 Установка программы 24

4.2.1 Пакетные менеджеры программного обеспечения 24

4.2.2 Альтернативные источники 25

4.3 Интегрированная среда разработки 25

4.4 Тестовый запуск «Hello world» 25

4.5 Примеры из практики 27

Глава 5. Создание виртуального рабочего пространства 28
5.1 Начало работы с VirtualBox 28
5.1.1 Установка VirtualBox 28
5.1.2 Определение виртуальной машины 29
5.1.3 Загрузка операционной системы 31
5.1.4 Установка операционной системы 32
5.1.5 Клонирование виртуальных машин для быстрого старта 33
5.1.6 Управление виртуальными машинами из командной строки 34
5.2 Работа с LXC 35
5.2.1 Начало работы с LXC 35
5.3 Запуск кода на удаленном сервере 38
5.4 Пример из жизни 38
Глава 6. Использование учебных ресурсов 39
6.1 Книги 39
6.2. Кураторские курсы 40
6.2.1 Какие есть хорошие сайты для онлайн обучения? 40
6.3 Курсы без куратора 42
6.4 Высшее образование 42
6.5 Открытый интернет 44
6.5.1 Научитесь составлять строки интеллектуального поиска 44
6.5.2 Используйте образцы общедоступного кода 45
6.6 Бесплатные решения 45
6.7 Тематическое исследование 46
Глава 7. Приложение 1: Основы работы в сети 47
7.1 Введение в адресацию TCP/IP 47
7.1.1 Адресация NAT 48
7.2 Введение в Domain Name System (DNS) 50
Глава 8. Приложение 2: базовое администрирование Linux 52
8.1 Дистрибутивы Linux 52
8.2 Работа с привилегиями root 53
8.3 Менеджер пакетов 54
8.4 SSH: безопасное удаленное подключение 55
8.4.1 Безопасная передача файлов 56
8.5 Файловая система Linux 56
8.5.1 Стандарт иерархии файловой системы Unix 57
8.5.2 Популярные команды в Linux 57
8.5.3 Разрешения на объект 58

8.5.4 Помощь....59
Глава 9. Приложение 3: Amazon Web Services....60
9.1 Перенос рабочих нагрузок....61
9.2 Рефакторинг и оптимизация облачных рабочих нагрузок....62
9.3 Serverless....63
Глава 10. Завершая все это....64

Глава 1. Управление изменением

Вы, возможно, привыкнете к этому. По всей вероятности, темпы изменений технологий никогда не замедлятся. Будь то новые устройства, новые платформы или новые парадигмы программирования и инструменты операционной системы. Нет смысла предполагать, что ваших текущих навыков будет достаточно, чтобы оставаться на плаву. Вам постоянно придется совершенствовать ваши навыки и приобретать новые.

Чтобы разобраться с этой проблемой, неплохо было бы начать с двух вопросов:

- В контексте вашей текущей работы и долгосрочных карьерных целей, насколько вы подвержены влиянию технологий?
- Какие именно навыки вам стоить освоить, чтобы добавить их в своё резюме?

Я постараюсь ответить на эти вопросы в этой главе. Остальная часть книги будет посвящена решению других проблем: в первом разделе (главы 2-4) будут рассмотрены некоторые методологии обучения, которые вы можете использовать, во втором разделе (главы 5-6) вы познакомитесь с некоторыми полезными инструментами обучения, а в третьем разделе (приложения 1-3) вы получите базовые знания о Linux, AWS сервисах и TCP/IP.

1.1 Почему ваша карьера нуждается в ориентации на будущее (*future-proofing*)

Итак, «насколько вы подвержены технологическим изменениям»? Ну, это зависит от того, как вы зарабатываете себе на жизнь. Вот несколько примеров:

- Если вы работаете в сфере разработки ПО, вам обязательно нужно следить за тенденциями рынка. Вы можете быть профессионалом в области BASIC или Pascal, но в настоящий момент рынок нуждается в разработчиках на PHP и JavaScript. Основываясь на ваших текущих навыках, уверены ли вы, что ваш нынешний работодатель найдет для вас работу через год или два?
- Если вы работаете администратором локальных серверов на своём предприятии, готовьтесь к изменениям. Большинство рабочих нагрузок, обслуживаемых локально, в ближайшем будущем будут перенесены на облачную платформу, например, такую как Amazon Web Services. И поскольку виртуализация продолжает изменять способ работы серверов, унаследованная инфраструктура, которая остается локальной, потребует меньше персонала для ее обслуживания.
- Если вы хотите начать свою карьеру и пытаетесь найти лучший способ продать свои навыки, вам понадобится глубоко проработанный профиль. Поверьте, резюме, в котором в разделе «Навыки» указано «Javascript-разработчик», звучит далеко не так хорошо, как «Java-разработчик с навыками работы с Angular и опытом в микросервисах на основе Docker».

Суть в том, что то, не получится использовать навыки, который были актуальны несколько лет назад и вам необходимо постоянно улучшать старые и получать новые навыки, для того чтобы оставаться востребованным специалистом.

1.2 Что вы должны изучить?

Не ждите того, что я буду рассказывать вам чему учиться. Я лишь могу предложить несколько идей, но разобраться вам придется самостоятельно.

Прежде всего, обратите внимание на тенденции в сфере IT. Какие платформы становятся все более популярными? Куда вкладывают большие деньги IT гиганты, такие как Amazon и Google? На специалистов каких языков программирования растет спрос?

Отличным инструментом для исследования является Google Trends. На рис. 1.1 показаны результаты поиска для Kubernetes (открытое программное обеспечение для автоматизации развёртывания, масштабирования и управления контейнеризированными приложениями) и vSphere (платформа виртуализации VMware) за последние пять лет. Как вы можете видеть, vSphere никуда не исчезнет в ближайшее время, но рост интереса к Kubernetes определенно растет.

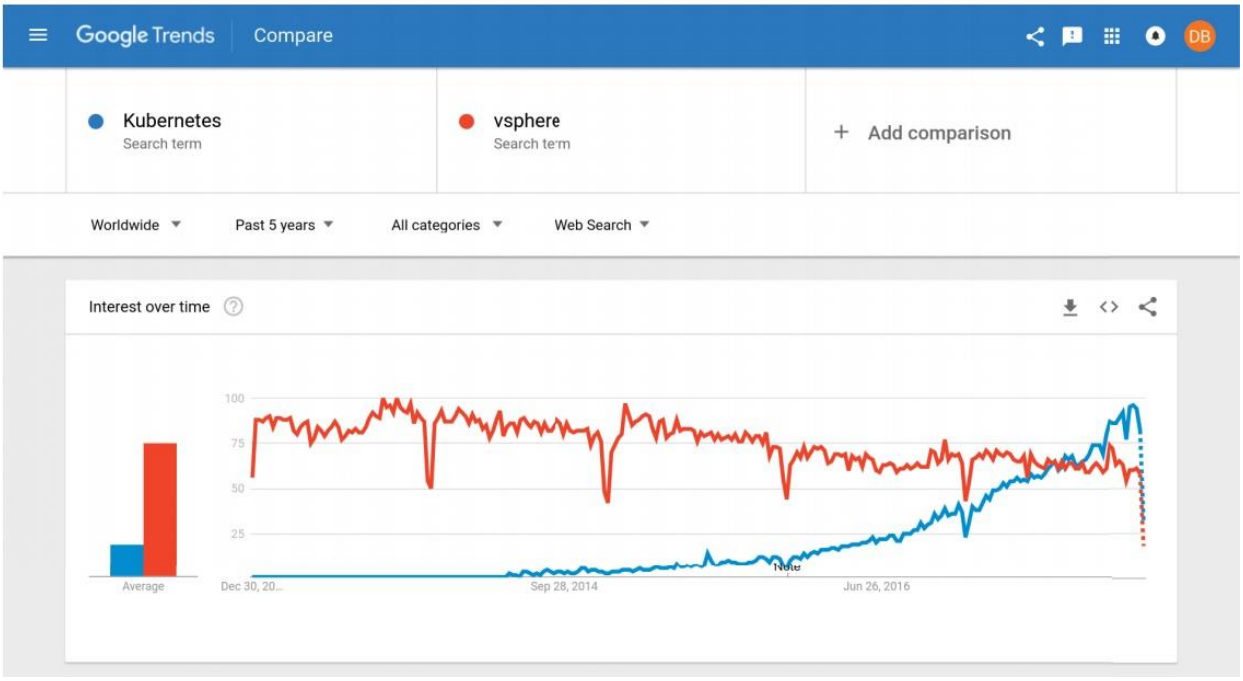

Рисунок 1.1. Пример поиска Google Trends, сравнивающий интерес с течением времени на двух популярных платформах виртуализации.

Однако точно оценить тенденции может быть непросто. Например, тысячи людей могут выразить желание узнать о новом языке программирования, но это не означает, что компании собираются инвестировать в проекты, построенные на нем. Нет проектов… нету и преимуществ в карьере для вас. И все мы видели технологии, чья внезапная стремительная популярность сменяется столь же внезапным забвением.

Чтобы действовать правильно, вам также нужно регулярно следить за информацией в области технологий, на таких ресурсах как Ars Technica, Tech Crunch, или (моя любимая) ZDNet. Также не помешает быть в курсе событий в мире политики, бизнеса и экономики, каждое из которых может оказать значительное влияние на IT компании.

Помимо постоянного чтения новостей, вам необходимо понимать смежные технологии. Например, AWS - отличное место для создания инфраструктуры для безопасных и

высоконадежных сервисов, но что будет через два или три года? Кто знает… возможно его сместит какая-то другая облачная технология.

Точно так же постарайтесь, по крайней мере, понять структуру одного или двух языков программирования, помимо того, который вы используете для большей части своей работы. Java и Python сегодня великолепны… но как бы выглядели ваши карьерные перспективы, если бы вы писали код на COBOL?

Баланс. Везде должен быть баланс.

Чтобы сделать эту главу еще более полезной, я кратко опишу некоторые технологии и объясню почему стоило бы узнать о них немного больше.

1.2.1 Облачные вычисления

Облачные вычисления — это предоставление по требованию вычислительных ресурсов и ресурсов хранения. Большая часть экономической эффективности облачной модели является результатом того, что плата рассчитывается только за используемые мощности и у вас нет необходимости покупать и обслуживать физическую инфраструктуру для своих задач.

Вместо этого вы фактически “арендуете” оборудования, чтобы справиться с меняющимся спросом на вашем сайте, и платите только за то, что вы действительно используете. Как показано на рис.1.2, при правильной настройке ваши облачные ресурсы будут автоматически масштабироваться в соответствии с спросом, что приведет к минимальным потерям и зачастую к значительной экономии средств. Microsoft (Azure), Google (Google Cloud Platform), и даже Alibaba (Alibaba Cloud) предлагают надежные облачные сервисы, но AWS (Amazon Web Services) остается доминирующим, как с точки зрения доли рынка, так и с точки зрения спектра предоставляемых ими услуг.

Учитывая, насколько быстро корпоративные, государственные и организационные мощности переносятся на облачные платформы, ваш переход на облако - лишь вопрос времени.

1.2.2 Web-разработка

Создание безопасной инфраструктуры, проектирование надежной архитектуры и хорошей бизнес-логики – это то, что происходит “под капотом” (backend). Но не забывайте, что помимо эффективности клиенту важно и удобство сервиса, который он сможет легко использовать для любых своих задач. Обычно под этим подразумевается создание удобного пользовательского интерфейса (frontend).

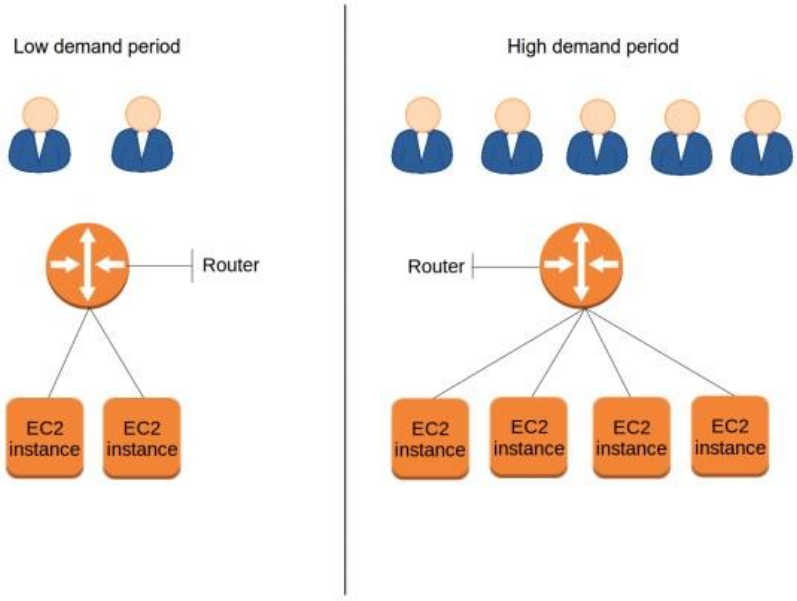

Рисунок 1.2. Эластичные системы могут динамически добавлять или удалять ресурсы для удовлетворения меняющегося спроса.

Вот несколько инструментов, которые вам нужно знать:

- **Hypertext Markup Language (HTML)** понятен веб-браузерам, которые помогают показывать HTML-файлы пользователям в привлекательной и удобной для чтения форме. Базовый HTML нетрудно выучить.
- **Cascading Style Sheets (CSS)** используется вместе с HTML, чтобы точно определить способ визуального оформления и представления элементов HTML. Хотя большинство функций CSS технически могут быть включены в файлы HTML, их разделение на отдельные файлы позволяет создать более эффективный и модульный дизайн документа.
- **JavaScript** это язык, который позволяет вам применять сложные вещи на web-странице — каждый раз, когда на web-странице происходит что-то большее, чем просто её статичное отображение — отображение периодически обновляемого контента, или интерактивных карт, или анимация 2D/3D графики, или прокрутка видео в проигрывателе, и т.д. — можете быть уверены, что скорее всего, не обошлось без JavaScript.

1.2.3 База данных

Большинство приложений генерируют данные, и данные - особенно когда они поступают в больших объемах - должны быть правильно организованы и поддерживаться.

Structured Query Language (SQL) — это стандартизированный синтаксис для управления данными в реляционных СУБД.

Реляционные БД (например, MySQL, MariaDB или Aurora Amazon) хранят структурированные данные, которые обычно представляют объекты реального мира. Скажем, это могут быть сведения о человеке, или о содержимом корзины для товаров в магазине, сгруппированные в таблицах, формат которых задан на этапе проектирования хранилища.

Нереляционные БД (например, MongoDB или Amazon DynamoDB), напротив, обладают гибкими схемами для неструктурированных данных. Они могут храниться по-разному: в колонках, документах, графах или в виде хранилища «ключ-значение».

1.2.4 Big data

Подумайте вот о чем:

- В настоящее время работают миллионы веб-приложений, которые делают свое дело.
- Существуют миллиарды устройств IoT (например, ваш умный холодильник или автомобиль), которые также делают свое дело.
- Каждое приложение и устройство могут легко выводить много мегабайт данных в день.

В совокупности они производят невообразимые объемы данных. Кто всё это может обработать? Кто это может использовать? Аналитики данных, инженеры данных и специалисты по машинному обучению — вот кто. Если у вас есть проект, который взаимодействует с большими потоками данных, подумайте о добавлении в свой портфель Apache Hadoop, Spark и Cassandra.

1.2.5 Docker

Контейнеризация (виртуализация на уровне операционной системы, контейнерная виртуализация) — метод виртуализации, при котором ядро операционной системы поддерживает несколько изолированных экземпляров пространства пользователя вместо одного. Рисунок 1.3 иллюстрирует базовую модель контейнера.

Docker в настоящее время является самой известной технологией контейнеризации в мире. Docker представляет собой систему управления контейнерами. Она позволяет «упаковать» приложение или веб-сайт со всем его окружением и зависимостями в контейнер, которым в дальнейшем можно легко и просто управлять: переносить на другой сервер, масштабировать, обновлять.

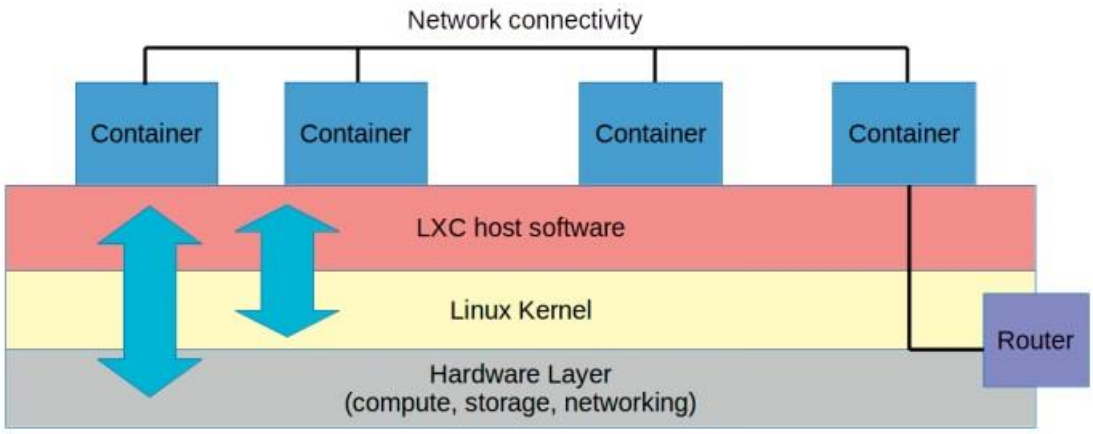

Рисунок 1.3. Контейнеры на физических серверах хоста.

1.2.6 Blockchain

Blockchain — это технология, интерес к которой вырос вместе с популярностью криптовалют. Сегодня ее широко обсуждают не только в мире финансов. Blockchain также используют для хранения и обработки персональных данных и идентификации, в маркетинге, компьютерных играх и других направлениях.

В дословном переводе blockchain — это непрерывная цепочка блоков. В отличие от обычных баз данных, изменить или удалить эти записи нельзя, можно только добавить новые. Blockchain еще называют технологией распределенных реестров, потому что всю цепочку сделок и актуальный список владельцев хранят на своих компьютерах множество независимых пользователей. Даже если один или несколько компьютеров дадут сбой, информация не пропадет.

Blockchain, представленный ниже на рис. 1.4, представляет собой пример использования технологии для проверки транзакции.

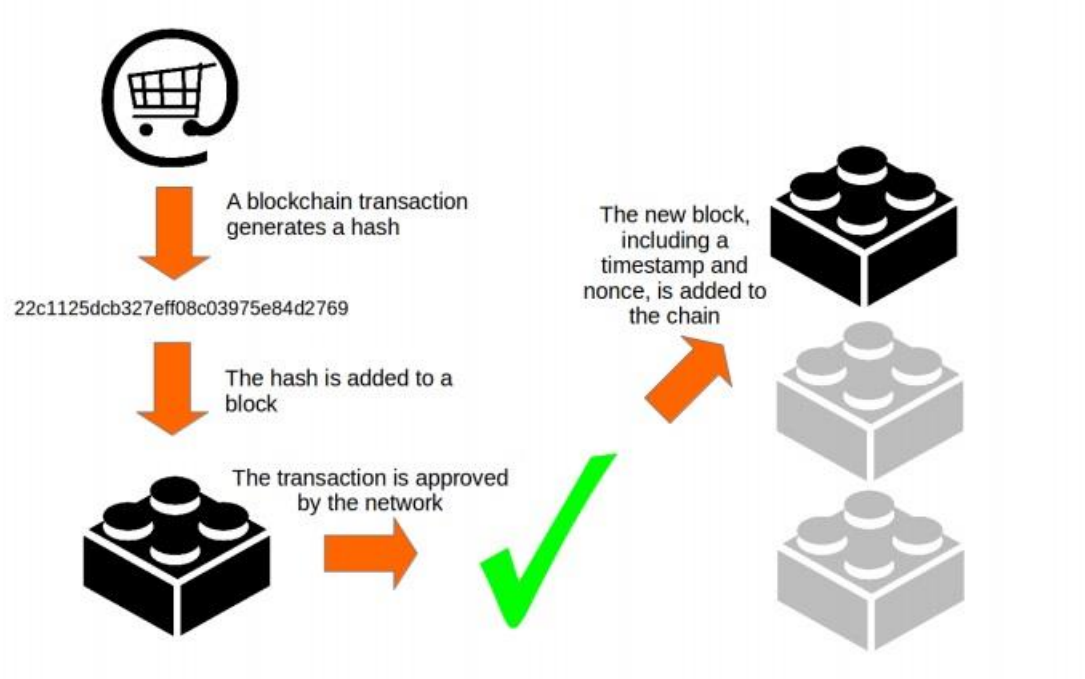

Рисунок 1.4. Пошаговое представление транзакции blockchaina.

1.2.7 Безопасность и шифрование

Практически любая инфраструктура уязвима для атак. От смартфонов до крупных корпоративных центров обработки данных: если нет нескольких уровней защиты, ожидайте сбоев в обслуживании. Честно говоря, даже если существует несколько уровней защиты, сбои в обслуживании все равно будут, просто в меньшем количестве.

Кто-то должен разрабатывать, тестировать и внедрять протоколы безопасности и алгоритмы шифрования. И кто-то должен построить межсетевые экраны и убедить людей использовать правильные пароли и многофакторную аутентификацию. Это работа, которую необходимо выполнить каждой организации, и эта потребность никогда не исчезнет.

1.2.8 DevOps

DevOps - это набор практик, которые работают для автоматизации и интеграции процессов между IT-командами, чтобы они могли создавать, тестировать и выпускать программное обеспечение быстрее и надежнее.

Часто реальные преимущества автоматизации заключаются в создании конвейера CI/CD с помощью, таких инструментов как CodeCommit, CodeBuild, CodeDeploy и CodePipeline от Amazon. На рис. 1.5 показан цикл типичного рабочего процесса DevOps.

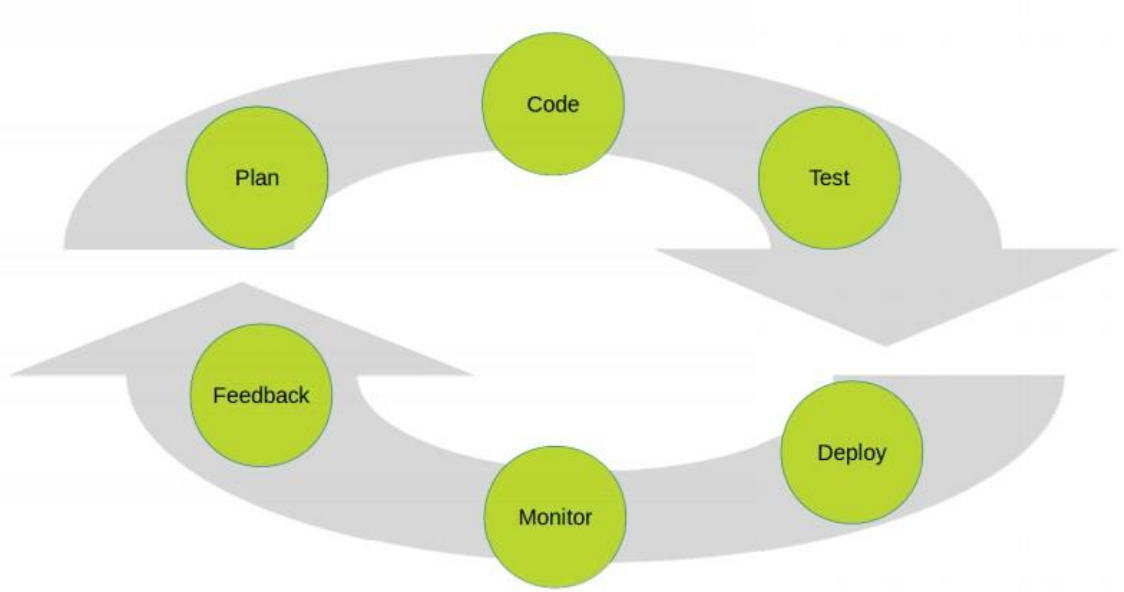

Рисунок 1.5. Цикл разработки с использованием DevOps практик

Глава 2. Подумай хорошенько

В этой главе…

- Выборе технологий, которые соответствуют вашему опыту и профессиональным потребностям.
- Ресурсах для обучения.
- Оптимизации учебного плана.

Есть методы и приемы, специфичные для изучения технологий, чтобы оставаться мотивированным и иметь правильный настрой. Они показаны на рис. 2.1.

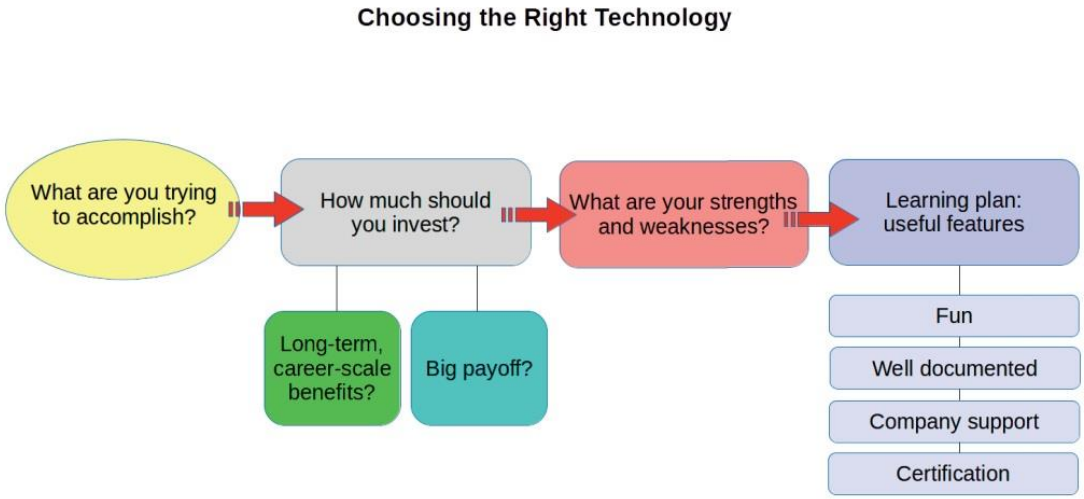

Рисунок 2.1. Поток решений, который может привести к правильному выбору в обучении.

2.1 Выявление проблемы, требующей решения

Прежде чем окунуться с головой в свое следующее учебное приключение, стоит уделить время тому, чтобы задать себе несколько вопросов. Четкое понимание того, почему вы что-то делаете, может улучшить то, как вы это делаете. Идея состоит в том, чтобы разработать для себя план обучения, который будет удобно сочетаться как с тем, где вы находитесь сейчас, так и с тем, где вы хотите быть через год или два.

Во-первых, что именно вы надеетесь получить от этой новой технологии? Чего не хватает для ваших целей, и какой навык исправит?

Являетесь ли вы системным администратором, который пытается лучше понять потребности пользователей системы, которую вы обслуживаете, или разработчиком, который хочет протестировать работу своего приложения?

Понимание ответов на эти вопросы, снизит вероятность того, что вы в конечном итоге потратите время, вкладывая средства во что-то, что не принесет пользы. Но большая «экономия» будет получена, если вы будете знать, насколько глубоко вам следует копать.

Вот очевидный пример. Если вы рассматриваете карьеру в области администрирования Linux, то вам следует искать учебную программу, охватывающую полный спектр инструментов Linux

(например, мою книгу "Linux в действии"). Но если вы заинтересованы только в том, чтобы подобрать инструменты, необходимые для безопасной подготовки и запуска вашего веб-приложения, то вам лучше начать с помощью простого пошагового руководства – например такого, как создание веб-приложения на стеке LAMP.

Вы должны убедиться, что ваш новый учебный проект хорошо соответствует вашим конкретным сильным сторонам и опыту. Амбиции велики и выход из зоны комфорта может быть полезным по множеству причин.

Не знаю, как вы, но я знал множество людей, чьи мечты приводили их к катастрофическим неудачам. Возьмите моего двоюродного брата в качестве примера того, как это делается правильно. Он действительно думал, что хотел стать дантистом, и у него хватило ума и энергии чтобы сделать это. Но, прежде чем принять окончательное решение, он провел неделю сидя в кабинете дантиста и наблюдая за его работой. В конце недели он твердо решил: работа не для него.

Но помните, план должен соответствовать вашей проблеме - он её должен решать.

2.2 Получение максимальной отдачи от вашего обучения

Я хотел бы обсудить еще несколько идей. Вскоре мы перейдем к реальным инструментам обучения и передовым методам, но некоторые вещи просто работают эффективнее, когда они полностью спланированы заранее и образование является одним из них.

Давайте представим, что вы сузили свои возможности в области новых технологий до короткого списка тем, которые хорошо соответствуют вашему опыту, потребностям и пониманию тенденций рынка. Какие фильтры вы теперь можете применить, чтобы сделать лучший выбор?

При прочих равных условиях, эти рассуждения должны показать вам, что не все вещи равнозначны.

- Очень важно чтобы вам нравился сам процесс обучения. Но и другие факторы имеют значение, например, такие как небольшое знакомство с более широкой средой технологии, могут иметь значение. Например, бэкграунд в Linux LXC containers облегчит обучение Docker.

- Не хватает свободного времени, чтобы учиться? Обратитесь к своему боссу: ваша компания может помочь вам в этом: оплатить курсы или выделить время на обучение. Некоторые компании даже активно поощряют такие стремления своих сотрудников, часто выделяя определенный процент рабочих часов на изучение новых навыков. Более того, многие организации имеют корпоративные учетные записи с более устоявшимися платформами онлайн-обучения, такими как Pluralsight.com, где также имеются и мои курсы.

- Хорошая сертификация может помочь вам убедить работодателей или клиентов в том, что вы знаете, что делаете. Тем не менее, сертификаты исторически лучше всего работали в области IT - для задач системного администрирования, сетевого взаимодействия и безопасности - больше, чем для языков программирования.

2.3 Оставаться вдохновленным

Вы знаете, как это бывает. Через несколько дней после начала нового учебного проекта вы обнаруживаете, что застряли в дебрях новой технологии. Ваши захватывающие долгосрочные мечты кажутся далекими, и ваш энтузиазм начинает угасать. Но я никогда не говорил вам, что это будет легко, верно? Готовьтесь к трудным временам и планируйте все для этого.

Вот несколько идей, которые вы можете включить в план.

2.3.1 Не бойтесь менять подход к обучению

Подумайте о том, чтобы нарушить некоторые правила. Не прорабатывайте все темы и не выполняйте все свои действия подряд, переходя от простого к сложному. Присмотритесь к технологии на более высоком уровне абстракции, попытайтесь понять общую идею и концепцию и лишь потом переходите к более мелким и конкретным деталям.

2.3.2 Изучайте по частям

Не пытайтесь решить всю проблему за один раз целиком. Лучше разбейте её на отдельные задачи, а их разделите на конкретные действия. Решайте их последовательно, ведь выполнение каждой отдельно взятой небольшой задачи воодушевит вас больше, чем очередная неудачная попытка сделать всё за один раз.

2.4 Ведение сбалансированного, здорового образа жизни

Вы знаете: нужно хорошо есть и спать, заниматься спортом, время от времени выходить на улицу, поддерживать тесную связь с семьей и друзьями...и звонить своей матери. Задача сложна, но нет ничего сильнее семьи.

2.5 Пример из практики

Я расскажу вам историю, которая поможет наглядно проиллюстрировать то, что мы обсуждали.

Кевин уже пару лет работает старшим разработчиком в небольшой розничной продуктовой сети. До сих пор он в основном отвечал за внутреннюю систему учета запасов. Но его компания теперь стала настолько большой и сложной, что для них имеет смысл напрямую интегрировать свою систему инвентаризации на основе Linux с инфраструктурой цепочки поставок, используемой их поставщиками. Чтобы это сделать, Кевину нужно будет нанять еще несколько разработчиков и назначить им отдельные задачи.

Нас не волнуют все детали, которые Кевину нужно будет рассмотреть, чтобы все заработало. Но его желание узнать, как инструменты DevOps могут сделать работу более продуктивной и эффективной, действительно интересует нас. Обдумывая это, Кевин имеет три различных мотивации для изучения DevOps:

- Расширение его навыков, выходящих за рамки простого написания скриптов и развертывания приложений, делает его более ценным для своего работодателя - особенно с учетом того, что успешное внедрение может сэкономить компании довольно много денег и избавить их от необходимости нанимать нового IT-администратора.

• Добавление хотя бы одной технологии DevOps в его портфолио сделает его более ценным на рынке труда.

В этот момент Кевин сделал мудрый выбор. Он решил сначала провести исследование, чтобы выяснить какие инструменты и платформы существуют и какие из них лучше всего подходят для его нужд.

Учитывая десятки вариантов доступных учебных ресурсов, это непростая задача. Если бы он был действительно умен, я полагаю, он мог бы обратиться к главе 16 из моего Linux in Action.

Во всяком случае, Кевин узнал о методиках непрерывной интеграции и непрерывного развертывания (CI/CD), а также о том, как они могут глубоко интегрироваться с Git (инструмент, с которым он уже был очень хорошо знаком). Он также отметил, что бы он ни выбрал, очень вероятно, что он будет более чем доволен игрой с серверами Linux, которые компания уже использовала.

Затем Кевин сузил варианты своей платформы до AWS Code Deploy, Ansible и простых сценариев Bash. Все они имеют хорошую документацию и большая поддержку сообщества и разработчиков. Как выбрать между ними?

Хотя то, что он узнал, было многообещающим, Кевин быстро исключил внедрение кода просто потому, что знал, что его босс еще не готов серьезно рассмотреть вопрос о переносе инфраструктуры компании в облако Amazon. Хотя, возможно, когда-нибудь…

Оставались Ansible и Bash. Преимущество Ansible заключалось в том, что он был специально создан именно для тех проектов, в которых нуждалась компания, и был выбором, который, вероятно, сделало бы большинство людей. Но процесс обучения был слишком сложным на тот момент.

Но было еще кое-что, что привлекло Кевина к Bash: администрирование Linux. Кевин давно был очарован безопасным, стабильным и доступным интерфейсом операционной системы с открытым исходным кодом, и он всегда стремился узнать о ней все, что мог, при любой возможности.

Добавьте к этому возможность изучить некоторые более глубокие навыки написания сценариев Bash и это звучит как правильно сделанный выбор. Он спросил у своего босса, может ли он использовать некоторое время компании, чтобы быстро освоить соответствующие навыки, и есть ли у компании уже учетная запись в онлайн-сервисе обучения технологиям (как Pluralsight).

Стремясь получить хорошее начало и не желая погружаться в детали, которые не были сейчас необходимы, он запустил виртуальную среду Linux, используя VirtualBox, где он мог свободно экспериментировать, не беспокоясь о том, что нарушит что-то важное. Если что-то произойдет, он просто выключит и удалит виртуальную машину, а через несколько секунд запустит другую. Интересно, как это работает? Будьте терпеливы. В главе 5 будет еще больше.

Кевин использовал свои виртуальные сферы для запуска тестовых расширений и разработки параметров конфигурации по одному за раз. Ему нужно будет только беспокоиться о том, чтобы собрать все это вместе и нажать на кнопку для реального внедрения в самом конце процесса, как только он освоит все шаги.

Глава 3. Рабочие привычки (богатых и знаменитых)

В этой главе …

- Как выйти из тупика в процессе обучения
- Учитесь на своих ошибках
- Записывайте свои достижения

3.1 Дисциплина

Держу пари, вы думали, что я начну этот раздел с искренним призывом к вашей лучшей стороне. Вы знаете, что: каждое утро нужно встать до рассвета, делать зарядку, выходить на пробежку и быть максимально эффективным и мотивированным.

Нет, я начну с малого, но очень важного в работе - перерывах. Почему? Потому что слишком долгая работа над одной сложной проблемой сложной иногда может затруднить творческое мышление. Вы, скорее всего, закончите тем, что безнадежно будете бегать по когнитивным кругам. Решение? Отдохни немного и приходите позже, и посмотрите на проблему свежим взглядом.

Конечно, это не означает, что вы должны тратить все время между настоящим моментом и “позже”, одержимо проверяя свои социальные сети или отслеживая последние новости. Я могу уверенно сказать, на своём личном примере, что такой перерыв обычно не помогает. Скорее, во время перерыва, необходимо делать что-то продуктивное вдали от компьютера или даже сосредоточиться на каком-то другом элементе вашего проекта.

Хорошо. Но где же дисциплина? Что ж, независимо от того, насколько хорошо вы организованы и умны, на самом деле все сводится к последовательности. Конечно, отвлечение от своих проблем может помочь, но независимо от того, насколько трудными будут обстоятельства, если вы не захотите снова броситься в бой, вы не добьетесь многого.

Это правдиво независимо от того, что вы пытаетесь сделать в жизни. Но настойчивость может быть особенно ценна при обучении созданию и внедрению приложений. Вы, вероятно, уже заметили, что просто один раз скомпилировать или загрузить что-то не всегда гарантирует, что в следующий раз это будет работать так же. В этом контексте дисциплина может означать заставлять себя тестировать свой код снова и снова, используя различные параметры, пока вы действительно не поймете, что происходит под капотом...и почему.

3.2 Не бойтесь экспериментировать… и ошибаться

Не ошибается тот, кто ничего не делает. Путь к успеху лежит через множество ошибок. Заставить что-то работать идеально с первого раза – не получитс. Например, если что-то заработало с первого раза, это может означать, что вы действительно не узнали ничего нового о том, как построена ваша технология. И это также означает, что в какой-то момент, когда ваш проект уже находится в производстве, вас ждет настоящая катастрофа.

Так что принимайте неудачи и делайте выводы основываясь на них. Однако принятие неудачи не означает формирования терпимости - или даже извращенного восторга-к боли. Скорее, речь идет

о том, чтобы научиться отслеживать сообщения об ошибках и неожиданные системные события, а также находить и интерпретировать сообщения записей в логах.

Пример? На первый взгляд сообщение типа “ImportError: нет модуля с именем x” может показаться не более чем раздражением, но на самом деле это просто ваш компьютер, вежливо сообщающий вам, что есть необходимый модуль Python, ожидающий установки. Запуск pip install x быстро решит эту проблему.

Ваша ОС будет генерировать огромные объемы данных логов, послушно сообщая обо всем происходящем в системе. Просто чтобы проиллюстрировать, как вы можете получить доступ к некоторым из этих знаний на Linux, эта команда journalctl вернет все последние записи журнала, классифицированные как “ошибка”. Это может быть полезно для отслеживания ошибок, которые, как вы знаете, произошли, без необходимости пробираться через тысячи строк исторических или тривиальных записей.

journalctl -p err –since yesterday

Аналогично, в следующем примере будут возвращены все события, относящиеся к службе веб-сервера Apache, запущенной в Ubuntu. То же самое в системе CentOS или Red Hat будет использовать httpd вместо apache2.

journalctl -u apache2

Избегайте копирования примеров кода, которые вы найдете в Интернете в руководствах по практическим рекомендациям, и просто вставляйте их в свой проект. Для этого есть две веские причины.

- Когда вы сами вводите команду - даже если вы просто читаете ее прямо в онлайн - источнике, - структура и логика синтаксиса станут для вас более очевидными. И когда вы что-то вводите неправильно или пропускаете детали, полученное сообщение об ошибке, уверяю вас, сделает вас умнее.

- Запуск кода или команд, которые вы не до конца понимаете, может привести к нарушению. Например, ваша операционная система или даже ваш счет кредитной карты. Хотя большинство фрагментов кода, которые вы найдете в Интернете, совершенно нормальны, полезно предположить, что тот, который вы собираетесь скопировать, содержит скрытые вредоносные программы, опасные ошибки или элементы, которые просто не могут хорошо сочетаться с чем-то другим, работающим в вашей системе. Всегда читайте и понимайте, что все делает, прежде чем запускать его.

Ввод команд самостоятельно также повышает вероятность того, что по прихоти вы решите поэкспериментировать с небольшими изменениями. Такое любопытство почти всегда приведет к хорошим местам. Если, конечно, команда, о которой вы говорите, не является старой командой администрирования файловой системы Unix/Linux dd. Одно неверное движение с dd (часто называемое “Разрушителем дисков”) может значительно уменьшить ваши долгосрочные изменения для счастья.

Сказав все это, неудавшиеся эксперименты могут привести к нежелательным последствиям. Помимо риска полного сбоя вашей системы, работа с длинным рядом расширений, плагинов и пакетов программного обеспечения может привести к конфликтам и сбоям конфигурации. Это также может затруднить выяснение того, что именно вызвало отдельную неудачу - или даже успех.

На самом деле, через некоторое время вы можете обнаружить, что ваша рабочая станция стала устаревшей. Без надежного компьютера для работы ваша производительность будет снижаться.

Решение? Виртуализация - то, как Кевин использовал VirtualBox в качестве среды в конце главы 2 (и то, что мы опишем более подробно, когда перейдем к главе 5). Таким образом, последствия сбоев и ошибок ограничиваются виртуальной машиной и не должны влиять на вашу физическую рабочую станцию.

3.3 Делайте заметки

Что-то, над чем вы работаете, терпит неудачу, и вы проводите несколько часов, безумно бегая в поисках решения. Наконец, после создания и перестройки среды и перебора десятков комбинаций параметров конфигурации, что-то щелкает, и всё снова работает. Измученный, но радостный, вы выключаете свой ноутбук и отправляетесь спать. Сейчас 3:15 утра. И на этом история заканчивается. Или, по крайней мере, вы думали, что это конец истории. Через несколько месяцев вы столкнетесь с точно такой же проблемой. Ваша первая реакция-облегчение: “Ну, по крайней мере, на этот раз я знаю, как это исправить”. Вы просматриваете старый код, журналы, электронную почту и даже кэш браузера в поисках подсказок, но ничего нет.

Звучит знакомо? Ну, я могу почувствовать вашу боль. Конечно, это случалось со мной, но из-за того, что я теперь одержим полным документированием своих действий, я едва могу вспомнить, когда в последний раз это было.

Вот как это должно работать:

3.3.1 Документирование

Пока вы все еще «экспериментируете», делайте частые копии своего кода или команд. Это может означать публикации в GitHub или сохранение изменений в обычный текстовый файл. Просто убедитесь, что идет полная запись вашего процесса, содержащая достаточно информации, чтобы вы могли ее восстановить. Вы можете увидеть пример заметок, которые я создал для посетителей моего курса Pluralsight «Network Vulnerability Scanning with OpenVAS» здесь: bootstrap-it.com/openvas. В основном это последовательный набор команд Bash.

Что хорошего принесут вам эти заметки, если ваша система выйдет из строя? Итак, убедитесь, что вы создаете резервные копии своих заметок (вместе со всеми другими рабочими файлами). В идеале это должно включать резервное копирование за пределами сайта в надежный облачный сервис, такой как Amazon S3. О, и разве вы не знаете, что в моей книге «Linux в действии» есть две главы - 4 и 5, посвященные именно таким процессам.

3.3.2 Тесты

Технические проекты, как правило, имеют много частей. Поэтому, прежде чем архивировать свои заметки, вы, вероятно, должны убедиться, что они действительно представляют рабочую версию.

Как? Проверьте их. То есть разверните новую среду и примените команды или код в точной последовательности и формате, используемых в примечаниях.

Более того, у вас также есть некоторый базовая информация, с помощью которой вы можете автоматизировать многие общие задачи. Об этом будет дальше.

3.3.3 Автоматизация инфраструктуры с помощью скриптов

Скрипт представляет собой набор символов, которые складываются в список команд. Вот пример простого Bash- скрипта, который установит, а затем перезапустит программное обеспечение веб-сервера Apache на компьютере с Ubuntu Linux:

```
#!/bin/bash

apt-get update

apt-get install apache2

systemctl restart apache2.service
```

Файл, предполагая, что он называется scriptname.sh можно разрешить к исполнению, а затем запустить из командной строки с помощью этих двух команд:

```
$ chmod +x scriptname.sh

$ sudo ./scriptname.sh
```

Разумеется, скрипты могут быть намного длиннее и сложнее, чем в этом примере, и могут быть сделаны для выполнения довольно впечатляющих вещей. И, конечно, скрипты также могут быть написаны и запущены в операционных системах, отличающихся от Linux, с использованием надежных инструментов, таких как, например, PowerShell в Windows.

Я упоминаю эту тему здесь, потому что автоматизация должна вас заинтересовать...и скрипты того или иного типа находятся в центре революции автоматизации.

Визуализируйте это: вы день и ночь трудитесь, изучая, как контейнеры Docker можно использовать для развертывания вашего веб-приложения. Разумно, что вы запускаете серверы VirtualBox для тестирования. Запуск новой виртуальной машины Linux (на основе клонированного образа) не займет много времени, но работа с операционной системой в процессе установки Docker может каждый раз занимать более пяти минут...и было бы предпочтительно не терять это время.

Хотели бы вы сократить это до одной команды, которая будет работать сама по себе и все равно завершится менее чем за минуту? Напишите скрипт. Для этого примера перейдите на страницы документации Docker, выберите инструкции по установке для вашей операционной системы, запишите команды, которые вам нужно будет выполнить, а затем вставьте их в скрипт – и готово. Хотя не помешало бы проверить его в действии, просто чтобы убедиться.

Здесь не место для подробной демонстрации, того, как это работает, но я хотел бы привести несколько примеров, чтобы дать вам представление о том, что возможно.

3.3.3.1 Использование скриптов Dockerfile

Этот файл с именем Dockerfile загрузит образ Ubuntu 16.04, установит Apache, создаст простую веб-страницу (index.html) с некоторым контентом (“Welcome to my website”). Не забудьте открыть 80 порт, чтобы разрешить входящие HTTP-запросы. Неплохо для нескольких строк кода.

```
FROM ubuntu:16.04

RUN apt update

RUN apt install -y apache2
```

```
RUN echo "Welcome to my web site" > /var/www/html/index.html

EXPOSE 80
```

3.3.3.2 Создайте сайт WordPress на основе Docker на AWS Elastic Beanstalk

Файл с именем Dockerrun.aws.json запустит два контейнера Docker, используя инфраструктуру AWS. Первый контейнер будет запускать ядро базы данных MariaDB, а другой будет содержать саму программу WordPress.

Два контейнера будут общаться друг с другом. Как только файл будет запущен, все, что вам нужно будет сделать, это перейти по URL-адресу, который покажет вам Elastic Beanstalk, и настроить свой сайт.

Этот процесс был должным образом объяснен в главе 19 «Learn Amazon Web Services in a Month of Lunches».

```
{
                "AWSEBDockerrunVersion": 2,
                "containerDefinitions": [
                        { "name": "mariadb",
                        "image": "mariadb:latest",
                        "essential": true,
                        "memory": 128,
                        "portMappings": [
                                {
                                        "hostPort": 3306,
                                        "containerPort": 3306
                                }
                        ],
                        "environment": [
                                {
                                        "name": "MYSQL_ROOT_PASSWORD",
                                        "value": "password"
                                },
                                {
                                        "name": "MYSQL_DATABASE",
                                        "value": "wordpress"
                                }
                        ]
                        },
                        {
                                "name": "wordpress",
                                "image": "wordpress",
                                "essential": true,
                                "memory": 128, "portMappings": [
                                        {
```

```
                                        "hostPort": 80,
                                        "containerPort": 80
                                }
                        ],
                        "links": [
                                "mariadb"
                        ],
                        "environment": [
                                {
                                        "name": "MYSQL_ROOT_PASSWORD",
                                        "value": "password"
                                }
                        ]
                }
        ]
}
```

3.3.3.3 Удаленное управление простым веб-сервером с помощью Ansible playbook

Ansible-это инструмент который позволяет автоматизировать создание полных стеков программного обеспечения на обширных парках удаленных серверов.

Идея заключается в том, что вы составляете один или несколько текстовых файлов, содержимое которых объявляет точное состояние, которое вы хотите для всего системного и прикладного программного обеспечения на указанной машине. При запуске Ansible прочитает эти файлы, войдет в систему на соответствующем хосте или хостах и выполнит все команды, необходимые для достижения желаемого состояния.

Вместо того, чтобы проходить утомительный и подверженный ошибкам процесс вручную на каждом из хостов, которые вы запускаете, вы просто говорите Ansible сделать все это за вас.

В этом примере (бесстыдно украденном из главы 16 "Linux in action") Ansible войдет на каждый удаленный сервер в вашей группе веб-серверов и добавит Apache и локально index.html файл. Наконец, он подтвердит, что служба Apache работает правильно.

```
---

- hosts: webservers

        tasks:

        - name: install the latest version of apache

                apt:

                        name: apache2

                        state: latest

                        update_cache: yes

                - name: copy an index.html file to the web root
```

```
        copy: src=index.html dest=/var/www/html/index2.html

        notify:

        - restart apache

- name: ensure apache is running

        service: name=apache2 state=started


handlers:

- name: restart apache

        service: name=apache2 state=restarted
```

3.4 Пример из практики

С тех пор как мы в последний раз встречались с ним, Кевин продвинулся вперед в своих исследованиях DevOps на основе Bash. Он считает, что каждый из его разработчиков может отправлять свои обновления кода со своих ноутбуков в Git репозиторий компании. Затем он может поместить новый код в виртуальные машины, которые он использует в качестве промежуточных серверов. Там его Bash скрипты автоматически включат код в «живое» приложение, которое может запускать тестовые соединения с одним из поставщиков вашей компании.

Но что-то пошло не так. Все попытки удаленного подключения истекают, и ИТ-команда в расположении поставщика сообщает, что они получают запросы данных от вашего приложения, но не получают ответ.

Кевин тщательно собирает как можно больше информации от поставщика, а затем просматривает журналы на своей локальной виртуальной машине. Похоже на проблему с сетью. Дальнейшие исследования подтверждают, что входящие запросы к виртуальной машине были заблокированы локальной NAT сетью. Через некоторое время Кевину удалось настроить переадресацию портов через локальный маршрутизатор, чтобы трафик, исходящий от поставщика - и только от поставщика - был разрешен.

Кевин был осторожен, чтобы немедленно задокументировать каждую настройку и шаг, которые он использовал в процессе успешного открытия доступа к сети. Он также протестировал свое решение на «чистой» виртуальной машине, а затем создал резервную копию документации в удаленном месте.

Затем он провел 20 минут, просматривая видео с милыми кошками на YouTube. Но он это заслужил.

Глава 4. Первые шаги

В этой главе …

- Поиск информации о технологии и способы её использования.
- Как пакетные менеджеры программного обеспечения могут быть полезны для установки инструментов и когда их лучше избегать.
- Как заглянуть под капот технологии.
- Как разработать план своего обучения.

Вы наверняка захотите найти информацию о том, как использовать ту или иную технологию для оптимизации своей работы, оценить её масштаб и как она подходит для вашей работы.

4.1 Общая картина

Когда я начинаю с нуля, я почти всегда иду искать информацию в Интернете, используя только название технологии. Чаще всего первые две ссылки, на которые я натыкаюсь – официальный сайт технологии и ее страницей в Википедии.

Обе ссылки, вероятно, будут содержать полезную информацию, но моей первой остановкой обычно является Википедия. Мой выбор таков, потому что из-за предсказуемого формата страницы Википедии я быстро найду информацию об основной функции технологии и категории продукта, в которой она используется.

Чтобы проиллюстрировать это давайте взглянем на первое предложение статьи Википедии о PHP:

> *PHP - скриптовый язык общего назначения, интенсивно применяемый для разработки веб-приложений. В настоящее время поддерживается подавляющим большинством хостинг-провайдеров и является одним из лидеров среди языков, применяющихся для создания динамических веб-сайтов*

С другой стороны, вот что в Википедии написано о Python:

> *Python — высокоуровневый язык программирования общего назначения с динамической строгой типизацией и автоматическим управлением памятью*

Оба введения информируют об основных функциях своих продуктов: «скриптовый язык общего назначения…» или «высокоуровневый язык программирования...» соответственно. Встроенные ссылки указывают читателям на отдельные страницы, посвященные объяснению таких тем, как «скриптовый язык» и «высокоуровневый язык программирования». Чтение этих страниц познакомит вас с другими инструментами, созданными для удовлетворения аналогичных потребностей, а также с более глубоким пониманием ценностей и назначения таких инструментов.

Такой высокоуровневый контекст может быть действительно полезным, чтобы вы лучше познакомились с предметом изучения. Но, потратив еще несколько минут на просмотр остальной части страницы Википедии и официального сайта, вы часто дополняете информацию важными структурными знаниями.

Например, интегрируется ли технология с вашей операционной системой по умолчанию? Покрывает ли существующая стабильная версия все ваши потребности или вам лучше установить бета-версию? Существуют ли какие-либо зависимости, которые могут конфликтовать с вашей текущей настройкой системы? Продолжается ли поддержка технологии и планируется ли её дальнейшее улучшение?

4.2 Установка программы

Установка сред программирования или инструментов администрирования может оказаться сложной задачей. Всегда есть множество вариантов установки технологии. Итак, если вы не возражаете, я уделю пару минут вашей жизни, чтобы поговорить о некоторых общих подходах.

Прежде всего, я бы обратил ваше внимание на работу с хранилищем Linux.

4.2.1 Пакетные менеджеры программного обеспечения

Большинство дистрибутивов Linux поставляются с пакетными менеджерами программного обеспечения, такими как APT для семейства Debian/Ubuntu и RPM/YUM для Fedora/CentOS. Они контролируют установку и администрирование программного обеспечения из онлайн-репозиториев, используя регулярно обновляемый индекс, отражающий состояние удаленного репозитория.

На практике это означает, что пользователи Linux могут запросить пакет с помощью одной короткой команды, например такой как `sudo apt install apache2`, и пакетный менеджер запросит онлайн-репозиторий, проверит необходимые зависимости и их текущее состояние, загрузит все необходимые пакеты, а также установит и настроит файлы, необходимые для установки или настройки.

Но настоящая красота пакетных менеджеров заключается в том, как они активно поддерживают общую стабильность системы. Например, когда исправленная версия программного обеспечения добавляется в вышестоящий репозиторий, установленная версия на вашем компьютере будет автоматически обновлена. Аналогично, когда вы решите удалить пакет, который вы больше не используете, пакетный менеджер незаметно проверит состояние системы и удалит только те зависимости, которые не используются каким-либо другим пакетом.

Концепция репозиториев программного обеспечения и интегрированных пакетных менеджеров была настолько успешной, что ее имитировали для других операционных систем. Пользователи macOS знакомы с HomeBrew, а совсем недавно был запущен OneGet для управления программным обеспечением в Windows 10.

Учитывая преимущества надежности и безопасности использования управляемого программного обеспечения, вы обычно не используете другие способы получения программного обеспечения. Но это не всегда так.

Во-первых, не все программное обеспечение можно найти в официальных репозиториях (особенно в репозиториях, отличных от Linux). Также будут возникать случаи, когда версия, доступная в репозиториях, немного устарела - то, с чем я недавно столкнулся с PHP, а иногда ваш проект просто требует настройки, которая не реализована в официальных версиях.

4.2.2 Альтернативные источники

Я не думаю, что мне нужно указывать пользователям Windows, как загружать и устанавливать программы. Объяснять, как установить Git на вашу машину, а также найти и извлечь репозитории пакетов, вероятно, также понятно любому, кто читает эту книгу. С другой стороны, описание ручной компиляции исходного кода с помощью make, может оказаться слишком специфичным для этого раздела.

Так, о чем же на самом деле этот маленький раздел? Речь идет о том, чтобы убедиться, что вы знаете обо всех возможностях, включая языковые менеджеры пакетов, такие как npm (для Node JavaScript), pip (для Python) и RubyGems (для Ruby).

4.3 Интегрированная среда разработки

Если у вас был какой-либо опыт написания кода, вы знаете, насколько сложной может быть настройка среды программирования и компиляции. Даже такая простая вещь, как настройка текстового редактора или интегрированной среды разработки (***Integrated development environment IDE***) именно так, как вы хотите, может оказаться сложной задачей.

Прежде чем слишком глубоко копаться в проекте, стоит провести разведку, независимо от того, хотите ли вы выбрать Eclipse, Visual Studio Code, Atom или Vim, вам необходимо убедиться, что вы понимаете, как он будет взаимодействовать с остальной частью вашего стека.

Оболочка командной строки (***Command-line interface, CLI***)— это то место, где я выполняю большую часть своей работы, и я никогда не откажусь от ее мощности и эффективности. На самом деле все три издания этой книги (Pbook, Ebook и HTML) были написаны в обычном текстовом редакторе из командной строки с использованием Pandoc и LaTeX.

Но, несмотря на мои явно иррациональные навязчивые идеи, я попрошу вас поверить мне, когда я скажу, что у GUI есть свои слабости. Программное обеспечение, построенное на графических интерфейсах, как правило, защищает критически важные программные файлы от вашего взгляда. Иногда это может быть удобно - особенно для менее опытных пользователей.

Но когда ваша работа заключается в разработке программного обеспечения или администрировании, слишком высокое абстрагирование графической оболочки может стать проблемой. Иногда вам просто нужно работать непосредственно на уровне файловой системы.

Имейте в виду, что полезно посвятить несколько минут изучению нового инструмента, пока вы распаковываете его. Используя документацию по продукту и инструменты поиска ОС, посмотрите, можно ли найти, где хранятся конфигурация, данные и исполняемые файлы инструмента.

Вот то, что вы можете найти в Linux. По соглашению, приложения Linux обычно хранят свои файлы конфигурации в иерархии каталогов /etc/, файлы программных данных в /var/ и исполняемые файлы в /usr/.

4.4 Тестовый запуск «Hello world»

После того, как вы установили все что нужно, осталось сделать только один шаг, прежде чем вы сможете погрузиться в свой первый серьезный проект: проверьте, что все работает, выполнив простую задачу, например, «Hello World».

Многие технологии, такие как сервис EC2 AWS, показанный на рис. 4.1, поставляются со встроенными образцами проектов, и вы сможете найти в Интернете информацию о быстрой настройке.

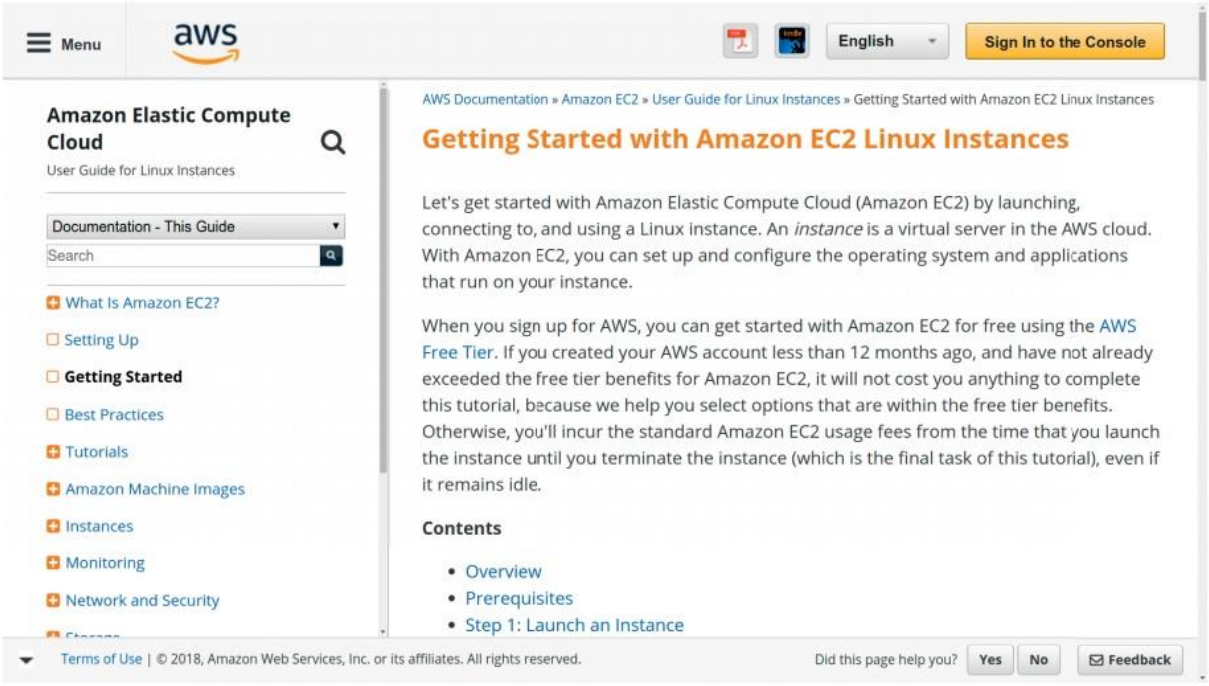

Рисунок 4.1. Страница документации AWS, содержащая рекомендации по началу работы с инстансом EC2.

Иногда единственный способ действительно понять что-то — это пройти через процесс вручную. Если вы хотите воплотить свои планы в жизнь, вам, возможно, придется взять дело в свои руки.

А теперь хорошие новости. Во многих случаях самодельная вариация демоверсии “Быстрый старт” может помочь вам понять большую часть того, что происходит на пути к вашей цели.

Вам не всегда нужно тщательно проходить шаг за шагом через весь процесс. Иногда немного агрессии может сэкономить вам массу времени и энергии. А тот факт, что вы работаете с одноразовой виртуальной машиной, означает, что ошибки практически не несут риска.

Конечно, «быстрый запуск» может быть не очень хорошей идеей для многоуровневых сред с полным стеком, таких как AWS. Там, начиная с самого начала и последовательно работая с запланированной учебной программой, вы можете избежать варианта с пропуском важных деталей, таких как способ выставления счетов или обеспечения безопасности Amazon. Поверьте мне: если вам не нравится идея неожиданной четырехзначной суммы за обслуживание за месяц или скомпрометированной инфраструктуры, то вы не захотите пропускать основы выставления счетов и безопасности.

Поэтому выделите несколько минут и подумайте о масштабах технологии, которую вы пытаетесь изучить. Вы могли бы получить важную информацию о том, как лучше всего подойти к проблеме.

4.5 Примеры из практики

Продолжении истории про Кевина…

По мере того, как он приближается к переводу своей инфраструктуры DevOps в продакшн, Кевин понимает, что ему придется немного сосредоточиться на безопасности. Поскольку его серверы работают под управлением Ubuntu, он нашел информацию о том, что есть три основных альтернативы брандмауэру: собственный несложный брандмауэр Ubuntu (UFW), Iptables и Shorewall. Все это, по счастливому совпадению, подробно обсуждается в главах 9 и 10 моей книги "Linux in action". Честно говоря, я не понимаю, почему Кевин еще не вышел и не купил свой собственный экземпляр.

Но он этого не сделал. Вместо этого он нашел информацию о работе с каждым инструментом в интернете. Он визуально просматривал руководства, копируя ключевые команды, которые, как он думал, подойдут для его нужд. Затем он запустил виртуальные машины с запущенными приложениями своей компании, установил и настроил брандмауэр и с отдельного компьютера проверил доступность, убедившись, что правильные запросы были разрешены, а неправильные запросы были отклонены.

В конце концов Кевин решил, что Shorewall требует слишком времени на обучение, а Iptables казались слишком пугающими. UFW идеально попал в цель. Кевин понял, что если бы его задача была бы немного сложнее, то у него не было бы другого выбора, кроме как погрузиться глубже в более сложную или пугающую платформу.

Глава 5. Создание виртуального рабочего пространства

В этой главе вы узнаете о…

- Использовании VirtualBox для создания легко реплицируемых и совместно используемых виртуальных операционных сред.
- Использование Linux Containers (LXC) для предоставления быстрых и эффективных виртуальных сред на хостах Linux.

Я собираюсь познакомить вас с бесплатным кроссплатформенным продуктом виртуализации VirtualBox от Oracle и, в частности, с некоторыми более продвинутыми приемами, позволяющими выжать больше выгоды из ваших (бесплатных) инвестиций. VirtualBox - это то, что вы можете использовать в любой операционной системе для создания виртуализированных операционных систем, работающих под управлением Windows или Linux любого типа.

> *Вам все равно необходимо получить лицензию на любой образ Windows, который вы решите запустить, хотя вы можете бесплатно установить и использовать копии без активации лицензии в течение месяца или около того.*

Если вы используете настольный компьютер или ноутбук Linux, вы можете рассмотреть возможность установки и использования контейнеров LXC вместо VirtualBox или в дополнение к нему. Если вам нужна простая серверная среда Linux, LXC обеспечит удивительно быструю и легкую работу.

Если вы уже читали мою книгу «Linux в действии» и некоторые из этих материалов кажутся вам немного знакомыми, то это потому, что эта глава представляет собой уменьшенную версию главы 2 «Linux в действии». Содержание было предоставлено с любезного разрешения Manning Publications.

5.1 Начало работы с VirtualBox

VirtualBox предоставляет среду, в которой вы можете запускать столько виртуальных систем, сколько могут обрабатывать ваши физические системные ресурсы. И, конечно же, это особенно полезный инструмент для безопасного тестирования и изучения новых административных навыков, что является нашей основной целью.

5.1.1 Установка VirtualBox

Хотите попробовать все это на ПК с Windows? Перейдите на сайт VirtualBox и загрузите исполняемый архив. Кликните на загруженный файл и выполните несколько шагов по настройке (все значения по умолчанию должны работать). Наконец, вас спросят, согласны ли вы с возможным сбросом сетевых интерфейсов, а затем хотите ли вы установить VirtualBox.

Установить VirtualBox на машину с Ubuntu еще проще. Всего две команды:

```
sudo apt update

sudo apt install virtualbox
```

5.1.2 Определение виртуальной машины

Определение новой виртуальной машины в VirtualBox работает примерно так же сборка компьютера из физических компонентов. Единственное существенное отличие состоит в том, что вместо того, чтобы опускаться на четвереньки с зажатым в зубах фонариком, чтобы вручную добавить оперативную память или увеличить размер памяти в вашем компьютере, VirtualBox позволяет вам определять «аппаратные» характеристики вашей виртуальной машины с помощью мыши.

После нажатия кнопки «New» в интерфейсе VirtualBox вы дадите имя виртуальной машине, и, как вы можете видеть на рисунке 5.1, программное обеспечение должно иметь возможность автоматически правильно заполнять поля «Type» (Тип) и «Version» (Версия). Выбранные здесь тип и версия не устанавливают фактическую операционную систему, а просто используются для применения соответствующих настроек эмуляции оборудования.

Рисунок 5.1. Диалоговое окно Create Virtual Machine: VirtualBox попытается угадать вашу ОС и версию ОС, чтобы позже предложить интеллектуальные варианты выбора по умолчанию.

На следующем экране вы выделите RAM для вашей виртуальной машины. Если вы не планируете запуска что-то требующее большого количества ресурсов - например, размещение нескольких контейнеров или запуск загруженного веб-сервера - объем по умолчанию (768 МБ) должен вас устроить. Вы, конечно, можете увеличить объем оперативной памяти, если это необходимо, но не забудьте оставить достаточно памяти для вашего хостовой системы и любых других виртуальных машин, которые на данный момент уже запущены. Поэтому, если у вашего хоста всего 4 ГБ оперативной памяти, вы, вероятно, не захотите отдавать половину этой памяти своей виртуальной машине.

Помните об этих ограничениях, если вы в итоге решите запускать несколько виртуальных машин одновременно - что будет полезно для тестирования более сложных инфраструктурных проектов. Даже если каждая виртуальная машина использует только объем памяти по умолчанию, две или три из них могут начать «съедать» оперативную память, необходимую для обычных операций хоста.

5.1.2.1 Определение вашего виртуального жесткого диска

Что за компьютер без жесткого диска? В процессе установки VirtualBox вас спросят, хотите ли вы создать новый виртуальный диск для своей виртуальной машины или использовать уже существующий. Бывают случаи, когда вы хотите использовать один диск для двух виртуальных машин, но я предполагаю, что в этом упражнении вам захочется начать с нуля. Поэтому выберите «Create a virtual hard disk now».

На следующем экране вы можете выбрать формат файла жесткого диска, который вы собираетесь создать. Если вы не планируете в конечном итоге экспортировать диск для использования в какой-либо другой среде виртуализации, формат VirtualBox Disk Image (VDI) по умолчанию будет подходящим для вас.

Я также ни разу не пожалел о том, что использовал параметр по умолчанию «Dynamically allocated», чтобы определить, как виртуальный диск будет занимать пространство на хосте. Под «Dynamically» они подразумевают, что пространство на диске хоста будет выделяться виртуальной машине по мере необходимости. Если использование диска виртуальной машины будет оставаться низким, то будет выделено меньше места на хостовой системе.

С другой стороны, на диске фиксированного размера сразу же будет предоставлен полный максимальный объем пространства, независимо от того, сколько он фактически используется.

Когда VirtualBox знает, что вам нужен Linux, и поскольку Linux эффективно использует дисковое пространство, VirtualBox, вероятно, предложит вам только 8 ГБ от общего размера диска, как показано на следующей картинке (рисунок 5.2). Если у вас нет необычно больших планов на виртуальную машину (например, вы собираетесь работать с некоторыми требовательными операциями с базой данных), это, вероятно, будет нормально. С другой стороны, если бы вы выбрали Windows в качестве операционной системы, по умолчанию было бы выбрано 25 ГБ, и не зря: Windows требует много ресурсов. Это отличная иллюстрация того, почему Linux так хорошо подходит для виртуальных сред.

Вы также можете изменить имя и местоположение, которое VirtualBox будет использовать для вашего диска на этом экране.

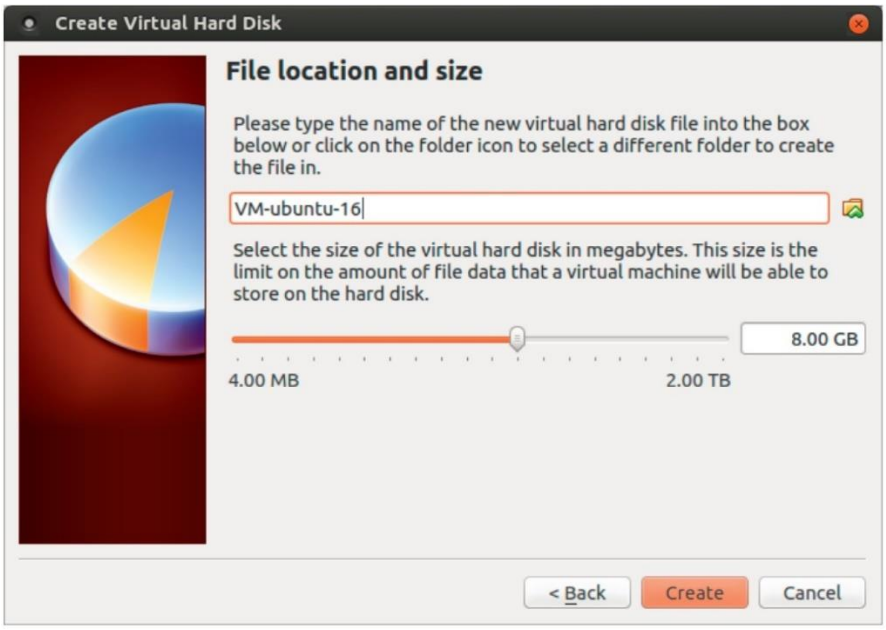

Рисунок 5.2. Если необходимо, ваш виртуальный диск может иметь размер 2 ТБ или максимальное свободное пространство на хост-устройстве.

Когда вы закончите, нажмите «Create», и новая виртуальная машина появится в списке виртуальных машин в левой части диспетчера VirtualBox. Наслаждайтесь вкусом успеха, но это еще не все: это была всего лишь машина. Теперь вам понадобится операционная система, чтобы воплотить планы в жизнь.

5.1.3 Загрузка операционной системы

Теперь, когда вы определили профиль виртуального оборудования своей новой виртуальной машины, необходимо сделать следующее:

- Загрузите файл (в формате ISO), содержащий образ операционной системы, которую вы хотите использовать.
- Загрузите новую виртуальную машину, используя виртуальный DVD-привод, содержащий загруженный вами ISO-образ.
- Пройдите стандартный процесс установки ОС.
- Загрузите виртуальную машину и запустите только что установленную ОС.

Вам нужно будет загрузить файл .ISO, содержащий файлы операционной системы и программу установки. Поиск нужного файла обычно сводится к поиску в Интернете: названия дистрибутива и слова «download». В случае с Ubuntu вы также можете просто перейти на страницу https://ubuntu.com и щелкнуть вкладку «Downloads», как показано на рисунке 5.3. Обратите внимание на различные доступные варианты Ubuntu. Если вы собираетесь использовать эту виртуальную машину для задач администрирования, то небольшая и быстрая версия Server, вероятно, будет лучшим выбором, чем Desktop.

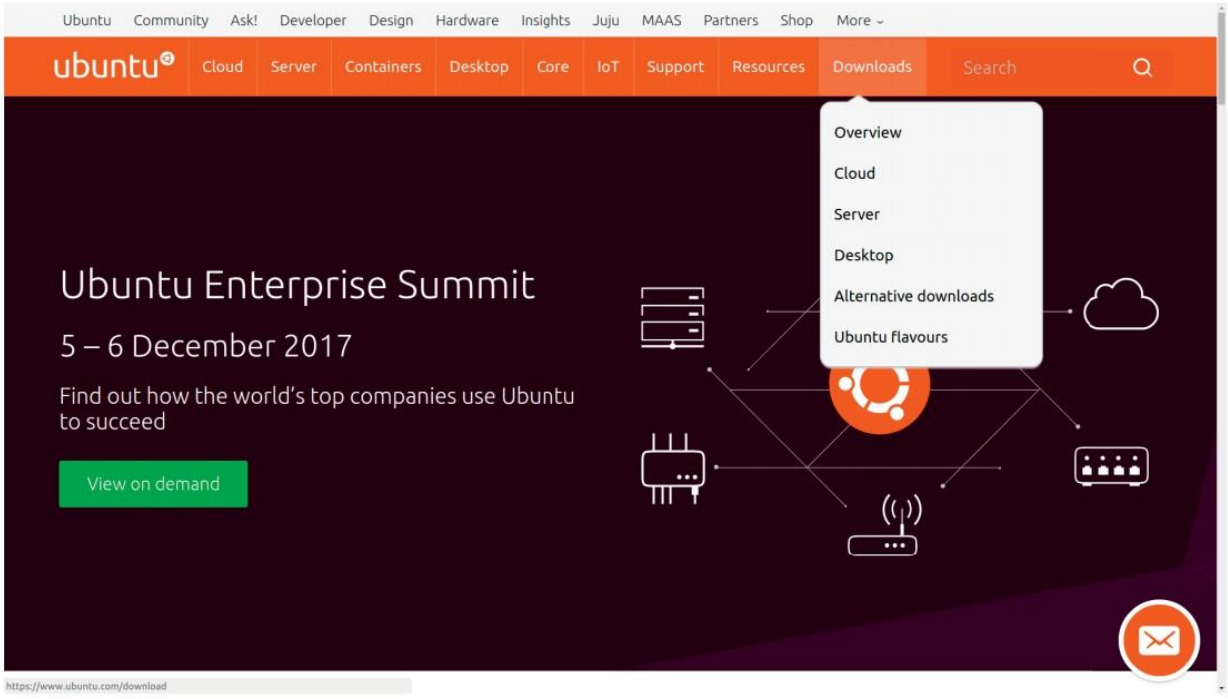

Рисунок 5.3. Раскрывающееся меню «Загрузки» на домашней странице Ubuntu.com. Обратите внимание на диапазон версий, которые предлагает Ubuntu.

5.1.3.1 Проверьте загруженный вами ISO архив

Иногда в процессе загрузки большие файлы могут быть повреждены. Если хотя бы один байт в вашем .ISO был изменен, есть шанс, что вы не сможете выполнить установку. Поскольку вы не хотите тратить время и силы только на то, чтобы искать проблему с загрузкой, всегда полезно немедленно вычислить контрольную сумму (или хэш) для загруженного .ISO, чтобы убедиться, что все в порядке.

Для этого вам нужно получить соответствующую контрольную сумму SHA или MD5 (которая представляет собой длинную строку, которая выглядит примерно так: 4375b73e3a1aa305a36320ffd7484682922262b3) из того места, где вы получили свой .ISO. В случае с Ubuntu это означало бы перейти на веб-страницу releases.ubuntu.com, щелкнуть каталог, соответствующий загруженной вами версии, а затем щелкнуть одну из ссылок на контрольную сумму (например, SHA1SUMS).

Вы должны сравнить соответствующую строку с этой страницы с результатами команды, запущенной из того же каталога, что и ваш загруженный .ISO, который может выглядеть следующим образом:

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

Если они совпадают, всё впорядке. Если нет - а вы дважды проверили, что выбираете нужную версию, - возможно, вам придется загрузить .ISO во второй раз.

5.1.4 Установка операционной системы

Как только ваш файл .ISO будет готов, вернитесь в VirtualBox. Выделив только что созданную виртуальную машину на левой панели, нажмите зеленую кнопку «Start» в верхней части приложения. Вам будет предложено выбрать файл .ISO из файловой системы для использования в качестве виртуального DVD-привода. Естественно, вы выберете тот, который только что скачали. Новая виртуальная машина прочитает этот DVD и запустит установку ОС.

> В большинстве случаев процесс установки проходит нормально. Однако для описания решений каждой из множества мелких вещей, которые могут пойти не так, потребуется пара полных глав. Так что, если у вас возникнут проблемы, вы можете обратиться к документации и руководствам, доступным для вашей операционной системы, или поделиться своим вопросом с онлайн-сообществом.

Когда все будет правильно установлено, может быть еще несколько вещей, о которых нужно позаботиться, прежде чем вы сможете успешно запустить свою виртуальную машину. Выделив имя вашей виртуальной машины, щелкните желтый значок настроек. Здесь вы можете поиграть со средой вашей виртуальной машины и настройками оборудования.

Например, щелкнув «Network», можно определить возможность подключения к сети. Если вы хотите, чтобы ваша виртуальная машина имела полный доступ в Интернет через сетевой интерфейс хост-машины, тогда, как показано на рисунке 5.4, вы можете выбрать «Bridge adapter» в раскрывающемся списке «Attached to», а затем имя адаптера вашего хоста.

> Использование Bridge adapter не всегда может быть вашим первым выбором и иногда может представлять угрозу безопасности. Фактически, выбор «NAT Network» - более распространенный способ предоставить виртуальной машине доступ в Интернет. Однако Bridge adapter — это самый простой способ получить полное сетевое соединение, поэтому, по крайней мере, для тестирования, это полезный подход.

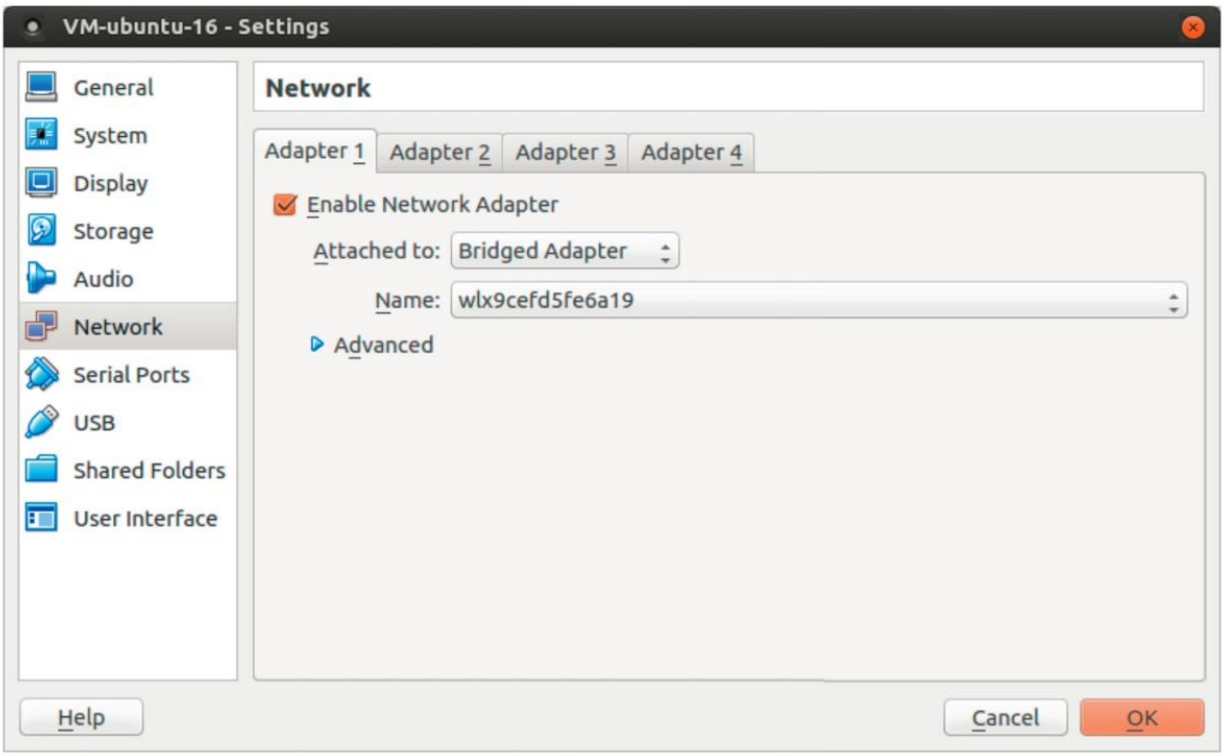

Рисунок 5.4: Вкладка сети на экране настроек. Вы можете определить, какой тип сетевого интерфейса - или интерфейсов - использовать для вашей виртуальной машины.

Следующие разделы немного приятны, но кому не нравятся бесплатные вещи? Я собираюсь рассказать вам о двух связанных приемах: как организовать виртуальные машины VirtualBox, чтобы как можно быстрее запускать новые, и как использовать командную строку для совместного использования виртуальных машин в сети.

5.1.5 Клонирование виртуальных машин для быстрого старта

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 весь процесс установки, слово «быстрой» не подходит.

То есть до тех пор, пока вы не займетесь клонированием. Почему бы не сохранить исходную виртуальную машину в ее чистом состоянии после установки и просто создать идентичный клон всякий раз, когда вы захотите выполнить реальную работу?

Это просто. Взгляните еще раз на приложение VirtualBox. Выберите (остановленную) виртуальную машину, которую вы хотите использовать в качестве главной копии, щелкните ссылку меню «Machine» и затем «Clone». Подтвердите имя, которое хотите дать своему клону, а затем, после нажатия кнопки «Next», хотите ли вы создать «Full clone» (это означает, что для новой виртуальной машины будут созданы совершенно новые копии файлов) или «Linked clone» (что означает новый VM поделится всеми базовыми файлами со своим мастером, сохраняя при этом вашу новую работу отдельно).

Выбор параметра «Linked» будет работать намного быстрее и займет гораздо меньше места на жестком диске. Единственный недостаток заключается в том, что позже вы не сможете перенести этот конкретный клон на другой компьютер. Это ваш выбор.

Теперь нажмите Clone, и на панели VM появится новая виртуальная машина. Запустите её, как обычно, а затем войдите в систему, используя те же учетные данные, которые вы установили на первоначальной машине.

5.1.6 Управление виртуальными машинами из командной строки

VirtualBox поставляется с собственной оболочкой командной строки, которая вызывается с помощью vboxmanage. Для чего использовать командную строку? Потому что, помимо других преимуществ, это позволит вам работать на удаленных серверах, что может значительно увеличить объем возможных проектов. Чтобы увидеть, как работает vboxmanage, используйте list vms, чтобы вывести список всех виртуальных машин, доступных в настоящее время в вашей системе. Вот как это выглядит на моей машине:

```
$ 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-lxd" {62bb89f8–7b45–4df6-a8ea-3d4265dfcc2f}
```

vboxmanage clonevm выполнит такое же действие клонирования, которое я описал выше, с использованием графического интерфейса. Здесь я делаю клон виртуальной машины Kali-Linux-template, называя копию newkali:

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

Это будет хорошо работать, пока мне нужно использовать только новую виртуальную машину здесь, на моем локальном компьютере. Но предположим, я хочу, чтобы у других членов моей команды была точная копия этой виртуальной машины - например, для того чтобы они могли протестировать то, над чем я работаю. Для этого мне нужно преобразовать виртуальную машину в какой-нибудь стандартизованный формат файла. Вот как я могу экспортировать локальную виртуальную машину в файл, используя Open Virtualization Format (.OVA):

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

Затем вам нужно скопировать файл .OVA на компьютер вашего коллеги. Имейте в виду, что файл имеет большой размер. Если у вас нет пропускной способности сети для передачи нескольких ГБ, рассмотрите возможность переноса с помощью USB-устройства.

После завершения передачи с удаленного компьютера остается только импортировать виртуальную машину в VirtualBox этой машины. Команда проста:

```
$ vboxmanage import docker.ova
```

Убедитесь, что операция импорта прошла успешно, используя list vms, и попробуйте запустить виртуальную машину с рабочего стола.

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

Вы также можете поделиться виртуальной машиной из графического интерфейса. Выделив машину, к которой вы хотите предоставить общий доступ, щелкните меню «File», а затем «Export Appliance».

Далее: загадочный и чудесный мир LXC.

5.2 Работа с LXC

VirtualBox отлично подходит для выполнения операций, требующих доступа к ядру, когда вам нужны сеансы рабочего стола с графическим интерфейсом пользователя или для тестирования периферийных операционных систем на нишевом рынке. Но если вы работаете на компьютере с Linux и вам просто нужен быстрый доступ к чистой среде Linux, и вы не ищете какую-либо специальную версию выпуска, тогда вам будет сложно превзойти Linux Containers.

Насколько быстро работают контейнеры LXC? Скоро вы сами это узнаете. Но поскольку они умело разделяют многие системные ресурсы как с хостом, так и с другими контейнерами, они работают как полнопроходные автономные серверы, используя только минимальное пространство для хранения и память.

5.2.1 Начало работы с LXC

Установить LXC на рабочую станцию Ubuntu? Нужно выполнить следующий код:

```
sudo apt update

sudo apt install lxc
```

Мы готовы приступить к делу. Базовый набор навыков LXC на самом деле довольно прост. Я собираюсь показать вам три или четыре команды, которые вам понадобятся, чтобы все это работало, а затем инсайдерский совет, который поразит вас, как только вы поймете, как организован LXC.

Почему бы не приступить к делу и не создать свой первый контейнер? Значение, присвоенное -n, устанавливает имя, которое я хочу использовать для контейнера, а -t сообщает LXC о необходимости создания контейнера из шаблона Ubuntu.

```
$ sudo lxc-create -n mycont -t ubuntu
```

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

```
$ sudo lxc-create -n centos_lxc -t centos

[...]
```

Временный пароль root хранится в:

```
'/var/lib/lxc/centos_lxc/tmp_root_pass'
```

Вы войдете в систему, используя имя пользователя «root» и пароль, содержащиеся в этом файле. Если, с другой стороны, ваш контейнер использовал шаблон Ubuntu, тогда вы будете использовать «ubuntu» как в качестве имени пользователя, так и в качестве пароля. Естественно, если вы планируете использовать этот контейнер для чего-то серьезного, вам нужно сразу же сменить этот пароль.

Используйте lxc-ls --fancy, чтобы проверить статус вашего контейнера:

```
$ sudo lxc-ls --fancy
```

```
NAME STATE AUTOSTART GROUPS IPV4 IPV6

mycont STOPPED 0 - - -
```

Что ж, он есть, но, видимо, требует запуска. Как и раньше, -n указывает по имени контейнер, который вы хотите запустить. -d означает «detach» - это означает, что вы не хотите автоматически переходить в интерактивный сеанс при запуске контейнера.

```
$ sudo lxc-start -d -n mycont
```

В списке ваших контейнеров теперь должно отображаться что-то вроде этого:

```
$ sudo lxc-ls --fancy

NAME STATE AUTOSTART GROUPS IPV4 IPV6

mycont RUNNING 0 - 10.0.3.142 -
```

На этот раз контейнер запущен, и ему был присвоен IP-адрес. Вы можете использовать этот адрес для входа в систему, используя SSH.

```
$ ssh ubuntu@10.0.3.142
```

В качестве альтернативы вы можете запустить сеанс корневой оболочки в работающем контейнере с помощью lxc-attach.

```
$ sudo lxc-attach -n mycont

root@mycont:/#
```

Когда вы закончите играть с новым контейнером, вы можете либо запустить «exit», чтобы выйти из системы, оставив контейнер запущенным:

```
root@mycont:/# exit

exit
```

… Или выключите контейнер с помощью shutdown -h сейчас.

Но прежде, чем вы это сделаете, давайте выясним, насколько быстрыми являются контейнеры LXC. Флаг -h, который я добавил к завершению работы непосредственно перед этим, означает «halt». Если бы я вместо этого использовал -r вместо того, чтобы отключиться навсегда, контейнер перезагрузился бы. Итак, давайте запустим перезагрузку, а затем попробуем сразу же войти в систему, чтобы увидеть, сколько времени потребуется контейнеру, чтобы снова подняться.

```
root@mycont:/# shutdown -r now

sudo lxc-attach -n mycont
```

Как все прошло? Готов поспорить, что к тому времени, когда вам удалось повторно ввести команду lxc-attach, mycont перезагрузился и был готов к действию. Но знаете ли вы, что нажатие клавиши со стрелкой вверх в Bash заполнит командную строку предыдущей командой. Это сделало бы запрос входа в систему еще быстрее. В моем случае заметной задержки не было. Контейнер выключился и полностью перезагрузился менее чем за 2 секунды!

> *Контейнеры Linux также очень бережно относятся к системным ресурсам. В отличие от моего опыта работы с виртуальными машинами VirtualBox, когда одновременное*

выполнение трех уже начинает серьезно влиять на производительность моей хост-рабочей станции объемом 8 ГБ, я могу запускать все виды контейнеров LXC без какого-либо замедления.

А что насчет того инсайдерского совета, который я вам обещал? Что ж, вернувшись в терминал на хост-машине (в отличие от контейнера), вам нужно будет открыть оболочку администратора с помощью sudo su. С этого момента - пока вы не наберете exit - вы будете работать с sudo.

```
$ sudo su

[sudo] password for username:

#
```

Теперь смените каталог на /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 свои собственные каталоги.

```
# cd /var/lib/lxc

# ls

mycont
```

Перейдите в каталог вашего контейнера и перечислите его содержимое. Там будет файл с именем «config» и каталог с именем «rootfs». «Fs» означает «файловая система».

```
# cd mycont

# ls

config rootfs
```

Это каталог rootfs, который вы должны сейчас увидеть:

```
# 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 они знакомы? Конечно! Все они являются частью стандарта иерархии файловой системы Linux. По сути, это корневой каталог (/) контейнера… но в файловой системе хоста. Пока у вас есть права администратора на хосте, вы сможете просматривать эти каталоги и редактировать любые файлы, которые захотите, даже если контейнер не запущен.

Есть много вещей, которые вы сможете делать с этим доступом, но вот тот, который вполне может однажды спасти вашу (профессиональную) жизнь. Предположим, вы сделали что-то глупое с контейнером и заблокировали себя, теперь ничто не мешает вам перемещаться по файловой системе, исправлять испорченный файл конфигурации и вернуться к работе.

Это правда, что экосистема Docker приобрела много уровней функций и сложности с тех пор, как технология вышла из-под тени LXC несколько лет назад. Однако под капотом он по-прежнему

построен на базовой структурной парадигме, которая будет мгновенно узнаваема для всех, кто знаком с LXC.

Это означает, что, если вы захотите опробовать самую быстрорастущую технологию виртуализации десятилетия, у вас уже есть какой-то опыт.

5.3 Запуск кода на удаленном сервере

Теперь, когда у вас есть все эти штуки с виртуальной средой, что вы можете с ними делать? Что ж, очевидно, что такие настройки идеально подходят для игры с системными инструментами и архитектурами.

Предположим, вы не занимаетесь системными вещами, но хотите безопасное место для создания приложений. Что я имею в виду под словом «безопасно»? Я говорю о среде, в которой вы можете свободно устанавливать библиотеки и пакеты зависимостей, не беспокоясь о дестабилизации вашего персонального рабочего компьютера.

Но есть пределы тому, как далеко вы можете зайти. Под поверхностью IDE, таких как Eclipse или Visual Studio, происходит много всего, и люди очень привязываются к их удобству. Но ожидать, что можно будет легко запустить все эти уровни сложности через удаленное соединение - возможно, немного амбициозно.

Но все же, разве неплохо бы было поработать в среде IDE на своем ноутбуке и сохранить, скомпилировать и запустить код удаленно… скажем, на размещенной виртуальной машине или облачном экземпляре? Таким образом, вы можете создавать свои приложения на серверах, на которых они будут запускаться, без риска для стабильности вашей собственной рабочей станции.

Это может быть сделано. Как правило, хитрость заключается в том, чтобы заставить вашу среду IDE работать поверх SSH. Для получения подробной информации поищите в Интернете что-то вроде eclipse edit java code on remote server.

5.4 Пример из жизни

Наш старый друг Кевин все еще усердно работает, пытаясь понять, как развернуть рабочий процесс DevOps для интеграции с поставщиками своей компании. Фактически, он приближается к созданию полного стека. Однако теперь ему нужно найти способ продемонстрировать вживую то, что он сделал, своим начальникам и представителям компаний-поставщиков.

Самым простым решением было клонирование двух или трех виртуальных машин, которые он использовал на своей рабочей станции, и копирование файлов архива на доступный ПК, находящийся в офисе генерального директора. После этого все, что нужно сделать Кевину, — это пройти в офис, пригласить генерального директора и его команду и запустить демонстрацию.

Всё прошло хорошо. Затем Кевин отправляет еще один комплект копий ИТ-отделу поставщика с инструкциями о том, как запустить демонстрацию самостоятельно.

Вопреки всему - и именно так, как вы ожидаете от вымышленного тематического исследования - все идет безупречно.

Глава 6. Использование учебных ресурсов

В этой главе вы узнаете о…

- Видах доступных онлайн-ресурсов для обучения технологиям и способах выбора между ними.
- Как быстрее и эффективнее найти лучшие интернет-ресурсы для решения ваших проблем.

Как вы обычно получает новые знания: С помощью книг? Видео курсов? Практических интерактивных занятий? И что вы обычно ищете, когда отправляетесь на поиски знаний: быстрые решения неотложных проблем? Глубокое понимание всей технологии? Быстрые руководства по началу работы?

Что бы вы ни искали, у вас будет больше шансов получить это, если вы будете знать, что происходит вокруг. Так что будьте внимательны ко многим категориям существующих средств обучения и присоединяйтесь ко мне для ознакомления с текущим состоянием онлайн образования.

6.1 Книги

Книги есть книги. Они существовали всегда, поэтому я сомневаюсь, что могу рассказать вам о них слишком много того, чего вы еще не знаете. И это включает в себя тот факт, что теперь вы можете читать их на своих телефонах и Kindle.

Но вот мысль, которая может оказаться полезной: издатели высокотехнологичных компаний могут инвестировать месяцы тщательного планирования и работы групп редакторов и рецензентов в свои продукты. Эти продукты содержать более концентрированную информацию и имеют большую ценность, чем та которую вы можете найти в других, менее продуманных, медиаформатах.

Поэтому, если вы ищете хорошо продуманную учебную программу, которая поможет вам пройти сложный путь через более крупную технологию, и вы относитесь к тому типу людей, которые хорошо учатся, используя информацию из книг, тогда подумайте о покупке книги по интересной вам тематике.

Некоторые издатели технологий лучше других? Что ж, некоторые издатели вкладывают больше времени и денег в создание книг, которые всесторонне преподносят тему. Так что переход к издателям высшего уровня увеличивает ваши шансы на успех. Но хорошие книги - попадают в мир через самые разные каналы.

Какие издатели находятся на верхнем уровне? Из-за моих постоянных отношений с Мэннингом я обязательно их включил. Но No Starch, O’Reilly и Wiley являются высококлассными издателями.

Существует множество других издателей технологической литературы, которых нельзя игнорировать, в том числе McGraw-Hill, Que, Addison-Wesley и Apress. Крупные корпорации, такие как Microsoft и Adobe, часто также публикуют свои собственные высококачественные книги, чтобы помочь разработчикам и пользователям легко адаптировать свои продукты.

Вам также следует обратить внимание на определенные «суббренды» в каталоге издателя. Например, Мэннинг издает книги в последовательном и надежном стиле и объеме, используя заголовки «x в действии» и «изучите x за месяц обедов». Подобные успешные серии книг включают «X для чайников» Уайли и «X: отсутствующее руководство» O’Reilly Media.

Если вы успешно освоили одну книгу из серии, велики шансы, что вам понравятся и другие.

6.2. Кураторские курсы

Как и книги, обучающие видеоролики будут более эффективными, если они будут тщательно спланированы, разработаны и отрецензированы. В отличие от книг, подписка на онлайн-сервис для всего сайта может мгновенно помочь вам найти решения любой конкретной проблемы, с которой вы столкнулись, независимо от темы. Но в любом случае обучение — это сложный процесс, и независимо от того, как и где проводится обучение, чем лучше учитель, тем больше вы узнаете.

Поэтому, если вам нужны хорошие онлайн-курсы, сначала поищите сайты, на которых работают только опытные преподаватели, изучите их коллекции курсов, чтобы обеспечить охват важных тем и исключить повторение, а также внимательно изучайте и просматривайте контент. Есть множество сайтов, которые следят за качеством своего контента.

6.2.1 Какие есть хорошие сайты для онлайн обучения?

Pluralsight.com. Pluralsight охватывает множество областей имея около 6000 курсов, распределенных по восьми областям (как показано ниже на рис. 6.1) и охватывающих набор навыков от начального до продвинутого. После расширения своей корпоративной направленности в июне 2016 года он теперь обслуживает более 50% компаний из списка Fortune 500, где обычно предоставляют полный доступ к учетным записям для многих или даже всех сотрудников. Я, конечно, пристрастен, так как у меня более десятка собственных курсов по Pluralsight. Pluralsight предлагает бесплатную 10-дневную пробную версию и платные аккаунты за 29 долларов в месяц или 299 долларов в год. Вы также можете пройти бесплатную оценку навыков, чтобы выявить пробелы, которые необходимо заполнить.

Lynda.com. Любая образовательная онлайн-компания, принадлежащая LinkedIn (которая, в свою очередь, теперь принадлежит Microsoft), вероятно, будет профессиональной и надежной. Как и у Pluralsight, у Lynda есть широкий спектр курсов, хотя их каталог больше ориентирован на дизайн и творческие навыки, и, судя по репутации, их курсы, как правило, немного более базовые, а для курсов разработки программного обеспечения, возможно, менее реальные. Lynda предлагает 30-дневную бесплатную пробную версию. После окончания пробной версии подписка будет стоить вам 19,99 или 29,99 долларов в зависимости от того, нужен ли вам доступ к файлам проекта и просмотр в автономном режиме.

CodeAcademy.com. CodeAcademy предоставляет сотни классов программирования среднего и продвинутого уровней бесплатно, но также имеет платные уровни. Уровень Codeacademy Pro, например, дает вам больше ресурсов для практики и подробную программу курса за 19,99 долларов в месяц, тогда как 8-10-недельные курсы Codecademy Pro Intensive включают больше практического опыта и стоят 199 долларов за курс. Их контент не основан на видео, а состоит из текстовых руководств, которые проведут вас через практические рабочие места. Оказывается, довольно эффективный подход.

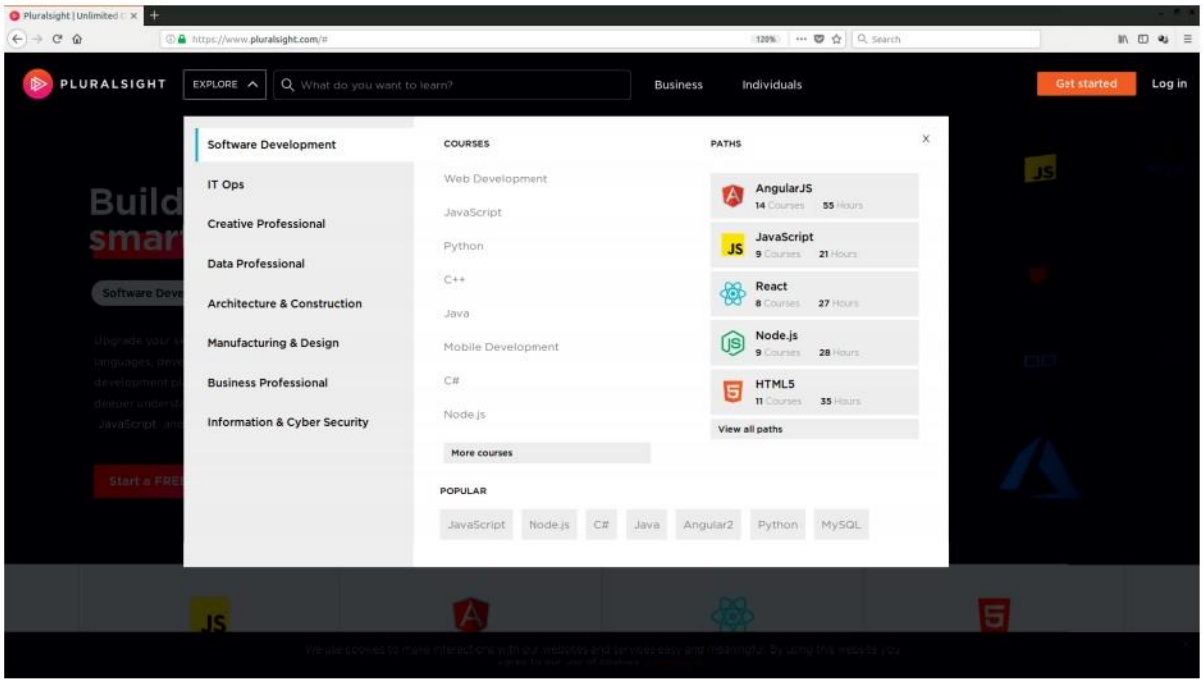

Рисунок 6.1. 6000 курсов Pluralsight разделены на восемь областей, крупнейшей из которых является разработка программного обеспечения.

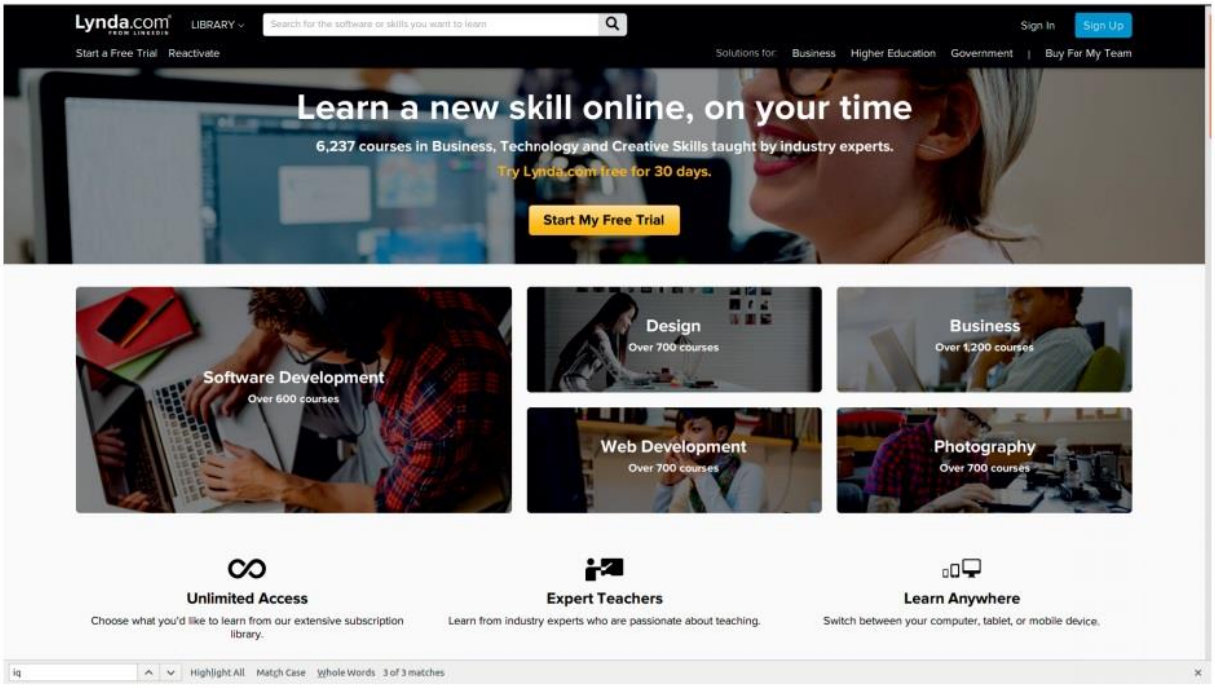

Рисунок 6.2. Разбивка курсов по предметам в Lynda

Treehouse (teamtreehouse.com). Судя по репутации, Treehouse в первую очередь ориентирован на людей, начинающих заниматься веб-дизайном и программированием. Помимо видео, сайт также предлагает «движок Code Challenge Engine» для прямого интерактивного взаимодействия. Базовое членство стоит 25 долларов в месяц, а уровень Pro - 49 долларов в месяц.

freeCodeCamp.org. Как видно из названия, это бесплатный ресурс. Этот сайт не только как способ привлечь трафик на какую-то приносящую доход веб-страницу, но и совершенно бесплатный, потому что его создатели просто считают, что такие возможности должны

существовать. Идея состоит в том, что «туристы» работают над реалистичными проектами, сосредоточенными на задачах кодирования. Но этот отличается в нескольких важных отношениях. Во-первых, участников лагеря поощряют присоединиться к другим местным участникам лагеря, чтобы они могли создавать группы, поддерживающие друг друга. После прохождения сертификации Front End, Data Visualization, Back End или Full Stack, участники лагеря также могут получить реальный опыт программирования для некоммерческих организаций. Наконец, freeCodeCamp проводит выпускников через этапы поиска работы и собеседования в их молодой карьере.

Помимо компаний, предлагающих курсы по всему спектру технических тем, вы также найдете онлайн-ресурсы с более конкретной направленностью, такие как CloudAcademy.com и LinuxAcademy.com.

6.3 Курсы без куратора

Не знаете, как заменить лампочку стоп-сигнала на стороне пассажира на вашем Dodge Caravan 2010 (3,8 л)? Вот видео на YouTube, в котором вы узнаете, как это сделать. Необходимо заменить датчик давления в печи с принудительной подачей газа Carrier, которой уже десять лет? Есть еще одно видео на YouTube, в котором вы узнаете, как это сделать. На самом деле, на YouTube вы найдете подборку видеороликов, которые покажут вам, как делать практически все, что вы можете вообразить, и многое другое, что вы не могли (и, возможно, не должны).

Есть одна очень специфическая проблема, которая мешает вашему прогрессу? Ищете обзор вашего следующего языка с высоты птичьего полета? Не исключайте, что кто-то там уже был и записал решение на видео. Также следите за авторами видео, которые вам понравились, и подписывайтесь на их каналы YouTube. Это упрощает поиск более полезного контента.

Пожалуй, самый известный и успешный канал YouTube из всех — это Salman Khan’s Khan Academy. Хотя он в первую очередь нацелен на школьников до 12 лет, в нем есть много полезного для людей, которые делают первые шаги в программировании - или в физике, или в электротехнике, если на то пошло.

На сайте Udemy.com есть тысячи видеокурсов по самым разным темам, включая технологии. Поскольку любой автор может загружать курсы практически по любой теме, коллекция не может считаться кураторской. Качество видео и преподавания будет значительно отличаться от курса к курсу. Но там достаточно разнообразия, и если вы потратите некоторое время на поиски, то поймете, что вам нужно. Udemy взимает плату за каждый курс, который вы решите пройти.

6.4 Высшее образование

Стоимость традиционных программ высшего образования за последние десятилетия резко возросла. Прямо сейчас четырехлетнее обучение в США может стоить вам примерно в пять раз больше среднего годового дохода семьи в 2016 году (который составлял около 59 000 долларов). Если ваша степень в конечном итоге принесет вам доход в размере 20 000 долларов в год, превышающий тот, который вы бы заработали без нее, вам все равно потребуется больше десяти лет, чтобы выйти на уровень безубыточности. И, возможно, еще много лет, чтобы выплатить реальный долг.

Подобные инвестиции не всегда имеют смысл. Но предположим, что вы могли бы получить те же знания совершенно бесплатно?

Добро пожаловать в мир массовых открытых онлайн-курсов (MOOC). MOOC - это платформа, с помощью которой существующее учебное заведение предоставляет контент курса всем в Интернете, кто хотел бы его использовать.

Присоединившись к MOOC, вы можете просматривать видеозаписи лекций лучших профессоров элитных университетов и участвовать в имитационных интерактивных лабораториях бесплатно и не выходя из собственного дома. Во многих случаях вы также можете получить баллы или сертификат, подтверждающий ваше успешное завершение курса. Сертификация часто требует определенных затрат.

Обратная сторона - хотя технически не все сочтут это недостатком - заключается в том, что университетские MOOC часто будут меньше ориентированы на работу и отрасль и будут тратить больше времени на общую теорию. Иногда они также ожидают, что вы уже освоили некоторые необходимые навыки STEM. Вот несколько крупных порталов MOOC:

Coursera.org. Пройти 4–10-недельные курсы Coursera вместе с тестами и упражнениями бесплатно. Но они также предлагают платные надстройки, включая оценки и сертификацию. Специализации — это несколько курсов Coursera, организованных в более крупную программу, такую как Data Science или Deep Learning. Чтобы получить сертификат специализации, студенты должны завершить выпускной проект. Категории Coursera включают информатику, науку о данных и информационные технологии.

edX.org. edX - это некоммерческая организация, изначально созданная Массачусетским технологическим институтом и Гарвардским университетом, чья платформа предоставляет учебные программы MOOC, созданные более чем сотней университетов и колледжей. Студенты могут пройти аудит курса бесплатно или за разумную плату получить подтвержденные сертификаты об окончании.

MIT OpenCourseWare. OpenCourseWare на самом деле не является обучающей платформой и не очень поможет вам, если вы ищете организованное руководство по определенной теме. Скорее, это онлайн-хранилище, содержащее заметки, викторины и некоторые видео с тысяч курсов Массачусетского технологического института. Контент может дать вам понимание конкретных вопросов, хотя, если вы достаточно амбициозны и решительны, вы можете изучить целые темы из обширных ресурсов, которые вы найдете.

Udacity.com. Я включил Udacity в этот раздел высшего образования, потому что в этом его корни. Но на самом деле, хотя основатели были выходцами из факультета Стэнфордского университета, изначально проект был своего рода бунтом против высоких затрат и раздутых университетских программ. Вместо того, чтобы тратить четыре с лишним года на изучение материалов, которые в значительной степени не соответствуют требованиям реального рынка труда, почему бы не сосредоточиться на навыках, которые ищет отрасль, и сделать это за гораздо меньшее время и за крошечную долю затрат?

Сегодня Udacity предлагает пару десятков или около того учебных программ, которые за несколько месяцев работы могут помочь вам достичь и превзойти начальный уровень компетенции в некоторых областях с высоким спросом. Поскольку учебные программы создаются при непосредственном участии крупных отраслевых работодателей, таких как Amazon, Nvidia и Google, есть неплохие шансы, что трудолюбивый выпускник быстро найдет отличную работу.

6.5 Открытый интернет

Вас ждет целый мир помощи. Не пропустите.

6.5.1 Научитесь составлять строки интеллектуального поиска

Интернет-поиск — это гораздо больше, чем просто ввод нескольких связанных слов в поле поиска и нажатия Enter. В безумии есть своя методика. Вот несколько полезных советов, которые подойдут для любой основной поисковой системы. Мой любимый - DuckDuckGo.

6.5.1.1 Используйте свою проблему, чтобы найти решение

Учитывая, что бесчисленные тысячи людей работали с той же технологией, которую вы сейчас изучаете, очень высока вероятность того, что по крайней мере некоторые из них столкнулись с той же проблемой, что и вы. И, по крайней мере, некоторые из них разместят свои вопросы на пользовательском онлайн-форуме, таком как StackOverflow. Самый быстрый способ просмотреть полученные ответы — это выполнить поиск о том языке, с которым вы столкнулись.

Возникло ли сообщение об ошибке в результате вашей проблемы? Вставьте именно этот текст в свою поисковую систему. Были ли сообщения журнала? Найдите и разместите их тоже.

6.5.1.2 Быть точным

В Интернете миллиарды страниц, поэтому расплывчатые результаты поиска обязательно включают множество ложных срабатываний. Вот почему вы хотите быть максимально точными. Один из эффективных приемов - заключить сообщение об ошибке в кавычки, сообщив поисковой системе, что вы ищете точную фразу, а не один результат, содержащий все или большую часть слов где-нибудь на странице. Однако вы не хотите быть настолько конкретным, чтобы в конечном итоге сузить результаты до нуля.

Следовательно, для записи из журнала ошибок Apache, подобной этой:

> *[Fri Dec 16 02:15:44 2017] [error] [client 54.211.9.96] Client sent malformed Host header*

… Вы должны не указывать дату и IP-адрес клиента, потому что никто другой не может получить эти точные данные. Вместо этого включите в кавычки только часть «*Client sent …*»:

> *“Client sent malformed Host header”*

Если это все еще слишком широкое понятие, попробуйте добавить строки Apache и [error] вне кавычек:

> *“Client sent malformed Host header” apache [error]*

6.5.1.3 Быть своевременным

Поисковые системы позволяют сузить поиск по времени. Если ваша проблема связана с относительно недавней версией выпуска, ограничьте поиск только последней неделей или месяцем.

6.5.1.4 Искать во нужных местах

Иногда внешняя поисковая система лучше выполняет поиск на большом веб-сайте, чем его собственный внутренний инструмент (я смотрю на вас: правительство Канады). Если вы чувствуете, что решение вашей проблемы, вероятно, находится где-то на определенном сайте - например, Stack Overflow's admin cousin, Server Fault - но вы не можете найти его самостоятельно, вы можете ограничить результаты только этим одним сайтом:

> "gss_accept_sec_context(2) failed:" site:serverfault.com

6.5.1.5 Знай, чего ты не хочешь

Наконец, если вы видите, что многие или все ложные срабатывания, которые вы получаете, содержат одно слово, которое вряд ли встретится на страницах, которые вы ищете, исключите его с помощью тире. В этом примере необходимо найти помощь в обучении написанию скриптов Bash, но в выводе продолжали появляться ссылки с советами для начинающих голливудских сценаристов. Вот как от этого избавиться:

> writing scripts -movie

6.5.2 Используйте образцы общедоступного кода

Вы застряли так, что застрять может только разработчик? Вы читаете свой код снова и снова, но не можете найти ошибку. Вы испробовали как минимум полдюжины различных подходов к дизайну и даже на совершенно другом языке. Ничего не помогло. Приложение не работает.

Haunt GitHub и другие места, где живут публичные репозитории кода. Все они доступны для поиска, и все они заполнены примерами отличного кода. Конечно, примеров действительно плохого и даже вредоносного кода будет предостаточно. Будьте бдительны.

Время, проведенное за просмотром чужого кода, - отличный способ почерпнуть новые идеи и узнать о передовых методах и шаблонах программирования. Если ваши навыки работы с поисковой системой так хороши, как я предполагаю, вы, вероятно, найдете рабочие решения того, что вас беспокоит.

6.6 Бесплатные решения

Необязательно делать все в одиночку. Прежде чем приступить к новому важному учебному проекту, внимательно посмотрите на свое сообщество и правительство, чтобы увидеть, какие услуги могут быть доступны.

Многие правительства предлагают поддержку - как финансовую, так и практическую - людям, желающим повысить свои профессиональные навыки. Все больше и больше руководств штатов или провинций присоединяются к движению за открытые учебники, где хорошо написанные и современные технические учебники размещаются в свободном доступе в Интернете. На данный момент я бы сказал, что качество большинства коллекций выглядит немного ненадежным, но долгосрочная цель - снизить стоимость годового обучения на многие сотни долларов.

Ваша компания может пожелать спонсировать ваше обучение. Фактически, многие компании предоставляют своим сотрудникам учетные записи на сайтах онлайн-обучения, и иногда достаточно просто спросить своего начальника или представителя отдела кадров о том, что может быть доступно.

Не то чтобы это должна была быть ваша компания. Обратите внимание на Microsoft Virtual Academy, которая, как вы можете видеть на рисунке 6.3, предлагает всевозможные бесплатные курсы в трех областях: для разработчиков, ИТ-специалистов и специалистов по данным.

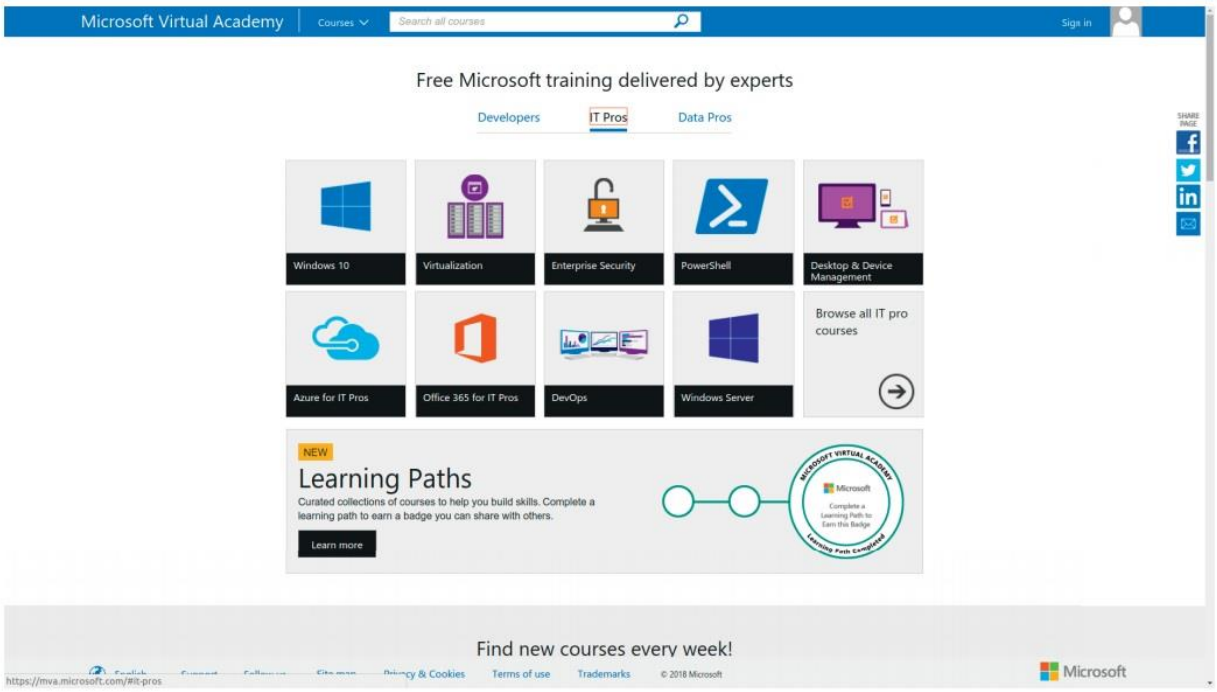

Рисунок 6.3. Категории курсов в домене ИТ-специалистов, доступные в Microsoft Virtual Academy

А как насчет вашего сообщества? Вы можете быть удивлены, насколько опытные и пожилые профессионалы счастливы заниматься наставничеством. Это может потребовать немного мужества, но идите вперед и подойдите к тому, кто вам нравится, чтобы узнать, какую мудрость и практическое руководство вы получите.

6.7 Тематическое исследование

В последнее время мы почти ничего не слышали от Кевина. Что он задумал? Вы не поверите, но он наконец завершил свое исследование и развернул новое приложение для интеграции с системами поставщиков своей компании. Имея свободное время, он начал новый учебный проект: администрирование сети Linux.

Но сейчас Кевин беспокоится. От пользователей время от времени поступали жалобы на невозможность подключения к серверу приложений. Немного: но пара на прошлой неделе и четыре на этой. Это тревожная тенденция.

Было бы неплохо получить представление о сервере и программном стеке.

Кевин почерпнул несколько идей из поиска в Интернете и даже несколько полезных советов от коллеги. Но теперь он ищет что-то более всеобъемлющее.

Воспользовавшись новой учетной записью Pluralsight, которую предоставила ему его компания, Кевин просмотрел библиотеку курсов, чтобы узнать, есть ли что-нибудь, что могло бы помочь. Что ж, посмотри сюда! Представлен новый курс «Linux Performance Monitoring and Tuning». И автор: Дэвид Клинтон. Вы думаете, мы связаны?

Глава 7. Приложение 1: Основы работы в сети

Я полагаю, что ключевая тема книги теперь рассмотрена, и с этого момента мы просто завершаем некоторые незавершенные дела. Возможно, «незавершенные дела» - не совсем правильное словосочетание. Давайте вместо этого попробуем «бонусный материал».

Я знаю. Эта книга должна была быть посвящена изучению новых технологий, а не самой технологии. Но я подумал, что добавлю некоторые базовые сведения о трех самых больших из всех технологий «большой палатки»: сети, Linux и Amazon Web Services. Независимо от того, какие инструменты вы в итоге изучите и с какими будете работать, трудно представить, что вы сможете пройти полную карьеру в сфере ИТ или разработчика, не встретившись в какой-то момент лицом к лицу с членами этого клуба.

Итак, мы начинаем с сети, клея, который скрепляет все вместе. И я имею в виду все. Забудьте об ИТ-мире: если сети выйдут из строя, наши банковские, транспортные, медицинские и промышленные системы исчезнут вместе с ними. Если вы хотите создать свой уголок современного мира, вам лучше понять, как он будет связан со всем остальным.

Содержание этой главы, доступное с любезного разрешения Manning Publications, взято из главы 14 книги «Linux in action» и главы 5 книги «Learn Amazon Web Services in a Month of Lunches».

7.1 Введение в адресацию TCP/IP

Одна из самых важных вещей сети — это простой IP-адрес, по крайней мере один из которых должен быть назначен каждому подключенному устройству. Каждый адрес должен быть уникальным во всей сети, иначе маршрутизация сообщений приведет к хаосу.

На протяжении десятилетий стандартный формат адреса следовал протоколу IPv4: каждый адрес состоит из четырех 8-битных октетов, в общей сложности 32-битных (не волнуйтесь, если вы не понимаете, как считать в двоичном формате). Каждый октет должен быть числом от 0 до 255. Вот типичный (поддельный) пример:

    154.39.230.205

Максимальное теоретическое количество адресов, которые можно извлечь из пула IPv4, составляет чуть более 4 миллиардов (256^4). Когда-то это казалось большим количеством. Но по мере того, как Интернет рос далеко за пределы чьих-либо ожиданий, в пуле IPv4 явно не хватало уникальных адресов для всех бесчисленных устройств, стремящихся к подключению.

> Четыре миллиарда возможных адресов звучат как большое число, если не учитывать, что в настоящее время используется более 1 миллиарда смартфонов Android — это в дополнение ко всем миллионам серверов, маршрутизаторов, ПК и ноутбуков, не говоря уже о телефонах Apple. Есть большая вероятность, что ваша машина, холодильник и домашние камеры видеонаблюдения также имеют свои собственные доступные в сети адреса.

Были предложены два решения надвигающегося краха системы адресации в Интернете (и конца ее жизни в том виде, в каком мы ее знаем): IPv6, который представляет собой совершенно новый протокол адресации, и преобразование сетевых адресов (NAT). IPv6 предоставляет гораздо больший пул адресов, но, поскольку он еще не так широко развернут, я сосредоточусь на NAT.

7.1.1 Адресация NAT

Организационный принцип, лежащий в основе NAT, прост и гениален: вместо того, чтобы назначать уникальный, доступный для чтения в сети адрес каждому устройству в вашем доме или офисе, почему бы не сделать так, чтобы все они совместно использовали один общедоступный адрес, который используется вашим маршрутизатором?

Но как трафик будет поступать на ваши локальные устройства и с них? За счет использования частных адресов. И если вы хотите разделить сетевые ресурсы на несколько подгрупп, как можно всем эффективно управлять? Через сегментацию сети.

Вот как это работает. Когда браузер на одном из ноутбуков, подключенных к вашему домашнему Wi-Fi, посещает сайт, он делает это с использованием общедоступного IP-адреса, назначенного DSL-модему / маршрутизатору, предоставленным вашим интернет-провайдером. Любые другие устройства, подключающиеся через ту же сеть Wi-Fi, используют этот же адрес для всей своей активности в Интернете (см. Рисунок 7.1).

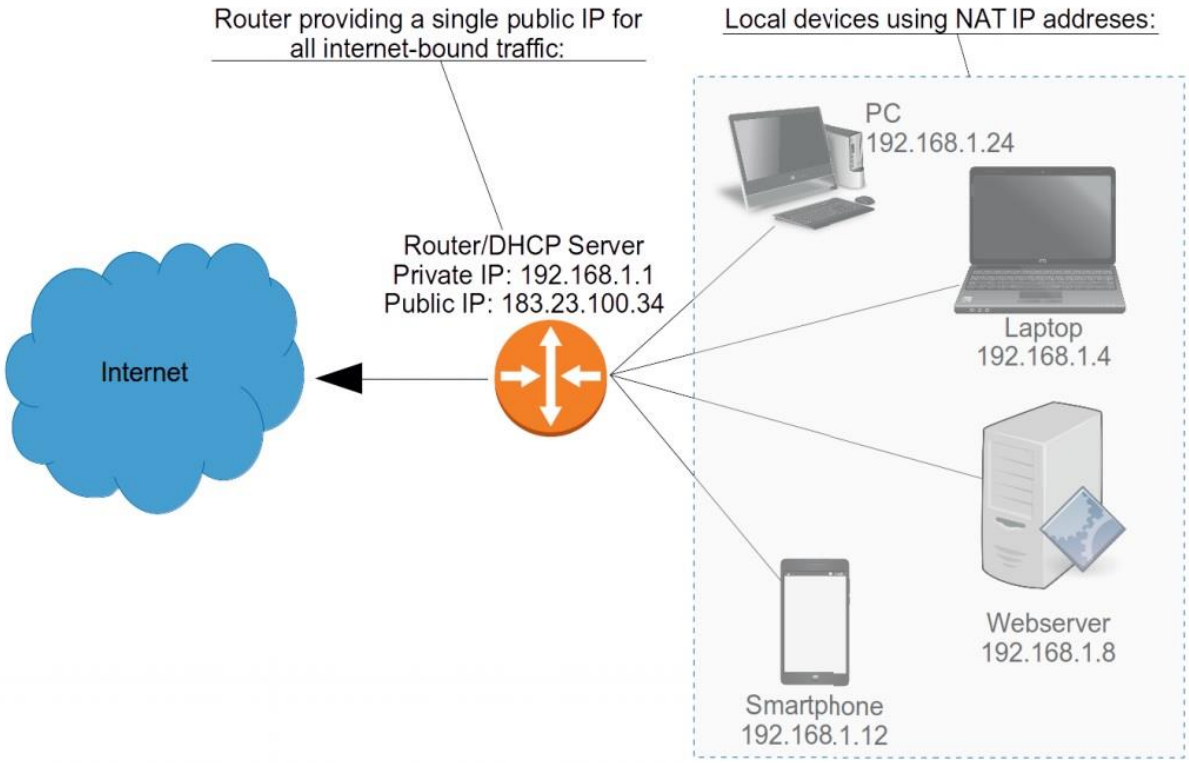

Рисунок 7.1. Типичная конфигурация NAT, показывающая, как несколько локальных устройств - каждое со своим частным адресом - могут быть представлены одним общедоступным IP-адресом.

В большинстве случаев маршрутизатор использует протокол динамической конфигурации хоста (DHCP) для назначения уникальных частных адресов (NAT) каждому локальному устройству, но они уникальны только в локальной среде. Таким образом, все локальные устройства могут иметь полноценную и надежную связь со своими локальными узлами. Это работает так же хорошо для крупных предприятий, многие из которых используют десятки тысяч IP-адресов NAT, и все они находятся за одним общедоступным IP-адресом.

Протокол NAT выделяет три диапазона адресов IPv4, которые могут использоваться только для частной адресации:

• 10.0.0.0–10.255.255.255

• 172.16.0.0–172.31.255.255

• 192.168.0.0–192.168.255.255

Менеджеры локальной сети могут использовать все эти адреса (их более 17 миллионов) как им заблагорассудится. Но адреса обычно организованы в более мелкие блоки сети (или подсети), сеть хоста которых идентифицируется октетами слева от адреса, оставляя октеты справа доступными для назначения отдельным устройствам.

Например, вы можете создать подсеть на 192.168.1, что будет означать, что все адреса в этой подсети будут начинаться с 192.168.1 (сетевая часть адреса) и заканчиваться уникальным однооктетным адресом устройства между 2 и 254. Таким образом, один ПК или портативный компьютер в этой подсети может получить адрес 192.168.1.4, а другой - 192.168.1.48.

> Следуя сетевым соглашениям, DHCP-серверы обычно не назначают числа 0, 1 и 255 последнему октету IP-адреса сетевого устройства.

Следуя этому примеру, вы можете впоследствии добавить параллельную, но отдельную сетевую подсеть с использованием 192.168.2. В этом случае не только 192.168.1.4 и 192.168.2.4 два отдельных адреса, доступных для назначения двум различным устройствам, но - поскольку они находятся в разных сетях - эти два могут даже не иметь доступа друг к другу (см. Рисунок 7.2).

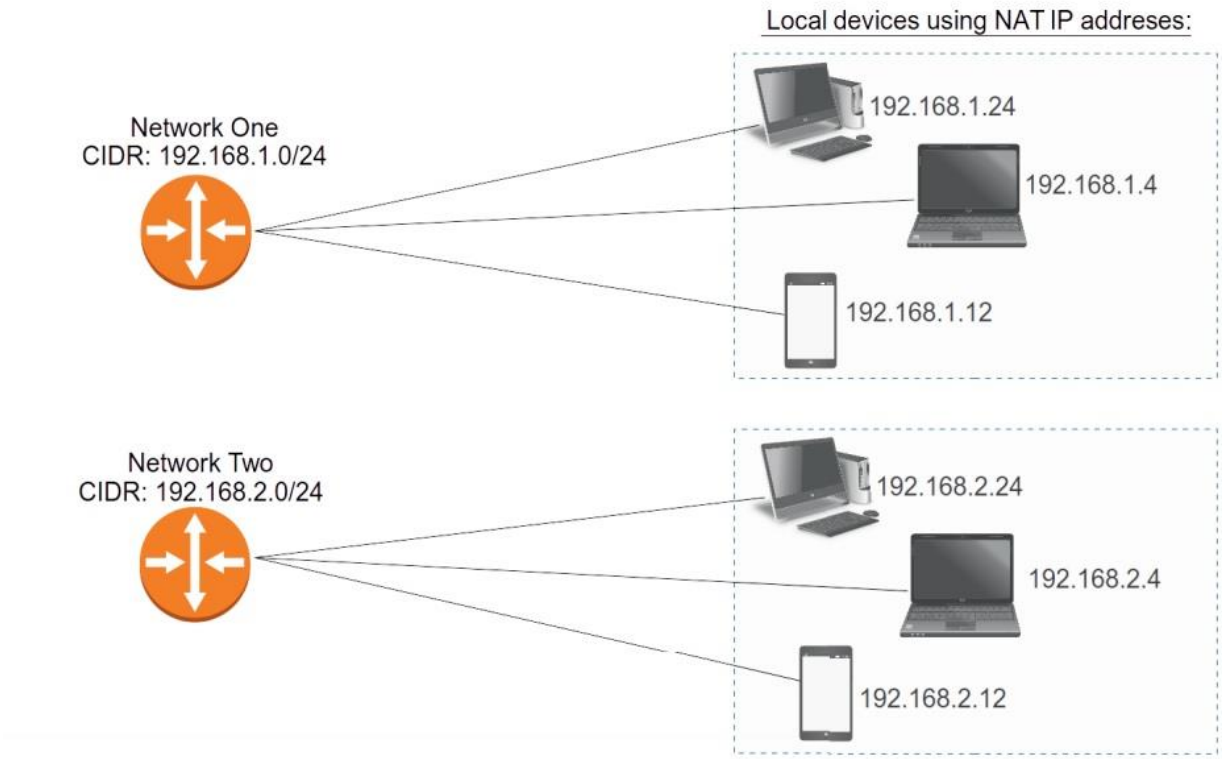

Рисунок 7.2: Устройства, подключенные к двум отдельным подсетям NAT в диапазоне сети 192.168.x.

Поскольку крайне важно убедиться, что системы знают, в какой подсети находится сетевой адрес, нам нужна стандартная нотация, которая может точно сообщать, какие октеты являются частью сети, а какие доступны для использования для устройств. Обычно используются два стандарта: нотация бесклассовой междоменной маршрутизации (CIDR) и сетевая маска. Используя CIDR, первая сеть в предыдущем примере будет представлена

как 192.168.1.0/24: / 24 сообщает вам, что первые три октета (8 * 3 = 24) составляют часть сети, оставляя только четвертый октет для устройства. адреса. Вторая подсеть в CIDR будет описана как 192.168.2.0/24.

Эти же две сети также можно описать с помощью сетевой маски 255.255.255.0. Это означает, что все 8 бит каждого из первых трех октетов используются сетью, но ни один из четвертых октетов не используется.

Необязательно разбивать блоки адресов именно таким образом. Если вы знали, что вам вряд ли когда-либо понадобится много сетевых подсетей в вашем домене, но вы ожидали, что потребуется подключить более 255 устройств, вы могли бы указать только первые два октета (192.168) в качестве сетевых адресов, оставив все, что находится между ними. 192.168.0.0 и 192.168.255.255 для устройств. В нотации CIDR это будет представлено как 192.168.0.0/16 и будет иметь маску сети 255.255.0.0.

Части вашей сети также не должны использовать полные (8-битные) октеты. Часть диапазона, доступного в конкретном октете, может быть выделена для адресов, используемых для целых сетей (например, 192.168.14.x), а оставшаяся часть остается для устройств (или хостов, как их чаще называют). Таким образом, вы можете выделить все адреса первых двух октетов подсети (192 и 168), а также некоторые из адресов третьего октета (0) в качестве сетевых адресов. Это может быть представлено как 192.168.0.0/20 или с сетевой маской 255.255.240.0.

Откуда я взял эти обозначения? Большинство опытных администраторов используют свои навыки бинарного счета, чтобы решить эту проблему самостоятельно. Но это выходит за рамки этой главы и не является необходимым для нормальной работы, с которой вы, вероятно, столкнетесь. Тем не менее, существует множество онлайн-калькуляторов подсети, которые сделают расчет за вас.

7.2 Введение в Domain Name System (DNS)

По мере того, как ваш веб-сайт растет и его открывают все больше людей, я уверен, что вам не понравится идентификация своего сайта по IP-адресу. Красивое, легко запоминающееся имя, например, best-site-ever.com, будет работать намного лучше. Давайте узнаем, как это работает.

Под всеми этими яркими, веселыми веб-страницами с внешними ссылками, которые отображаются в виде мягко вытесненных трехмерных блоков и обозначаются броскими, легко запоминающимися именами, прячутся числа. На самом деле нет места под названием google.com или wikipedia.org; скорее, это 172.217.3.142 и 208.80.154.224. Программное обеспечение, которое выполняет всю работу по подключению нас к известным и любимым веб-сайтам, распознает только числовые IP-адреса.

Инструмент, который переводит туда и обратно между людьми, помешанными на тексте, и нашими машинами, более ориентированными на цифровую обработку, называется системой доменных имен. Домен — это слово, которое часто используется для описания отдельной группы сетевых ресурсов, в частности ресурсов, идентифицируемых по уникальному имени, например, bootstrap-it.com. Как показано на рисунке 7.3, всякий раз, когда вы вводите текстовый адрес в своем браузере, постоянно - и незаметно – запрашивается информация от DNS-сервера.

Первой остановкой обычно является локальный указатель имен и связанных с ними IP-адресов, хранящийся в файле, который автоматически создается ОС на вашем компьютере. Если в этом локальном индексе нет ответа на этот конкретный вопрос о переводе, он перенаправляет запрос на общедоступный DNS-сервер, который поддерживает гораздо более полный индекс и может связать вас с нужным вам сайтом. К хорошо известным общедоступным DNS-серверам относятся

серверы, предоставляемые Google, использующие восхитительно простые адреса 8.8.8.8 и 8.8.4.4, и OpenDNS.

Пока что-то не сломается, вы обычно не будете тратить много времени на размышления о DNS-серверах - если, конечно, вы не хотите, чтобы ваши клиенты могли получить доступ к вашему веб-сайту по его простому текстовому имени. Для этого вам необходимо зарезервировать желаемое имя у регистратора доменных имен. Задача регистратора - обновлять индексы, используемые большими DNS-серверами, чтобы запросы на перевод от любого пользователя в Интернете могли быть быстро удовлетворены.

Как только вы преодолеете это критическое препятствие, вы можете вернуться к игнорированию DNS.

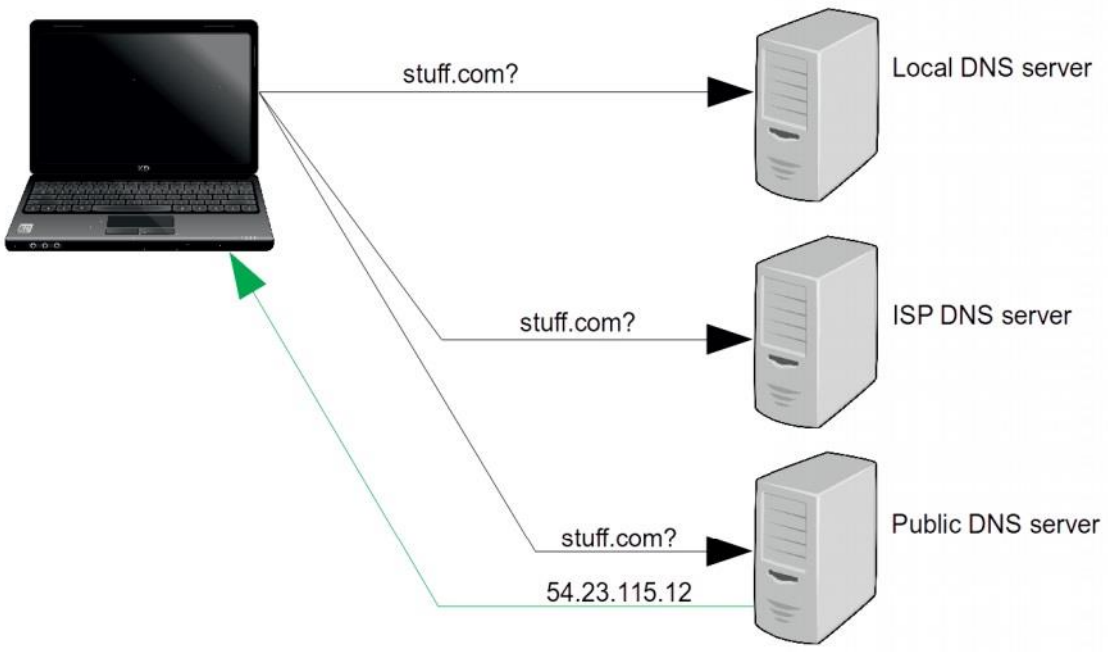

Рисунок 7.3. Запрос адреса DNS для stuff.com и ответ, содержащий (вымышленный) IP-адрес

Глава 8. Приложение 2: базовое администрирование Linux

Что такого хорошего в Linux, что еще в первой главе я причислял его к своим любимым технологиям? Что ж, позвольте мне сказать так: если вы планируете запустить рабочую нагрузку на веб-сервере, устройстве IoT, облачном экземпляре или суперкомпьютере, то есть огромные шансы, что вы будете работать с Linux.

Linux – это операционная система, которая является бесплатной, стабильной, безопасной, бесконечно настраиваемой, поддерживаемая солидным, зрелым сообществом и, как вы скоро увидите, доступна в виде разных дистрибутивов, созданных для всех видов использования.

Итак, поскольку большинство технологий в наши дни работает на платформах Linux, вам понадобится хотя бы базовое знакомство с тем, как устроены файловая система, управление пакетами и меры безопасности.

Конечно, это всего лишь одна короткая глава, и она не заменяет надлежащих навыков администрирования. Что еще хуже, даже согласно моей философии преподавания, эта глава написана неправильно. Как я писал в своей книге «Linux in action»,

> «Linux in action» переворачивает обучение технологиям. То есть, в то время как другие книги, курсы и онлайн-ресурсы организуют свой контент по категориям («Хорошо, мальчики и девочки, все берите свои правила слайдов и угольные карандаши. Сегодня мы собираемся узнать о файловых системах Linux»), я собираюсь использовать реальные проекты для обучения.
>
> Так, например, я мог бы написать целую главу (или две) о файловых системах Linux. Но вместо этого вы узнаете, как создавать корпоративные файловые серверы, диски восстановления системы и скрипты для резервного копирования архивов критически важных данных - в процессе чего вы получите знания файловой системы в качестве бесплатного бонуса.
>
> Не волнуйтесь, все основные навыки и функциональные возможности, необходимые в первые годы карьеры администратора Linux, будут охвачены тогда, когда они действительно необходимы для практического и критически важного проекта. Когда вы закончите, вы узнаете в значительной степени то, что могли бы получить из традиционного источника, но вы также будете знать, как выполнить более десятка крупных административных проектов. И будьте чувствовать себя комфортно на десятках других.

8.1 Дистрибутивы Linux

Как я только что заметил, существует множество версий Linux, называемых дистрибутивами. Идея состоит в том, что вы можете выбрать дистрибутив, наиболее соответствующий потребностям вашего проекта.

Как только вы начнете искать, вы найдете дистрибутивы, оптимизированные для облака, виртуальных машин или серверов. Существуют дистрибутивы для восстановления системы, операций по обеспечению безопасности или новейших экспериментальных аппаратных платформ. Существуют даже потребительские дистрибутивы для настольных компьютеров и ноутбуков, красота и функциональность которых поразят вас.

Рисунок 8.1 должен помочь вам сделать выбор, показывая некоторые из самых популярных дистрибутивов и их распространенные варианты использования.

Один бросился в глаза? Зайдите на его веб-сайт, загрузите соответствующий файл .ISO - и вперед.

Подумайте о том, чтобы записать ISO-образ на флэш-накопитель и опробовать ОС в режиме USB-сеанса. Вы должны понимать, что дистрибутив работает на вашем оборудовании, но ваш фиксированный диск с данными останется нетронутым.

Linux Distribution Types and Key Examples

Purpose	Distribution
Security/anti-hacking	Kali Linux
Consumer desktop	Mint
Lightweight (great for old hardware and diagnostics)	Puppy Linux
Internet of Things administration	Snappy Ubuntu Core
Enterprise server room	CentOS (community version of Red Hat Enterprise Linux)
Cloud computing	Amazon Linux (AWS AMI) Ubuntu Server
All-purpose (except lightweight)	Ubuntu

Рисунок 8.1: Распространенные варианты использования дистрибутива Linux и популярные примеры.

8.2 Работа с привилегиями root

При первой установке большинства дистрибутивов Linux вам будет предложено создать учетную запись пользователя с теоретическими административными привилегиями. Я говорю «теоретический», потому что при нормальном использовании у вас будут привилегии только на ресурсы своей учетной записи, но не на ресурсы других учетных записей или принадлежащие самой системе.

Но для выполнения задачи системного администрирования, требующей редактирования файла конфигурации системы, вам нужно убедить Linux в том, что у вас есть привилегии. Для этого вы временно возьмете на себя привилегия root, предварительно указав команду sudo. Таким образом, чтобы использовать текстовый редактор nano для открытия и редактирования файла /etc/group, вы должны запустить это в командной строке:

```
$ sudo nano /etc/group

[sudo] password for david:
```

Почему бы не работать полный рабочий день как root? Не хорошая идея. Если ваша учетная запись пользователя была каким-то образом взломана при использовании полных привилегий root, злоумышленник теперь будет контролировать всю систему. Всегда лучше, если возможно, уменьшить поверхность атаки.

8.3 Менеджер пакетов

Одно из величайших удовольствий Linux — это окружающая его глубокая и здоровая экосистема свободного ПО. От офисных пакетов корпоративного класса до профессионального программного обеспечения для редактирования аудио и видео и полнофункциональных средств разработки - среди десятков тысяч доступных пакетов FOSS.

Но что еще лучше, так это то, что все это программное обеспечение управляется интегрированными менеджерами пакетов. Что такое менеджер пакетов? Это локальный интерфейс, который соединяет ваш компьютер с удаленными репозиториями программного обеспечения.

Но это гораздо больше. Основные менеджеры пакетов (в частности, APT для систем Debian / Ubuntu / Mint и YUM / DNF для RHEL / CentOS / Fedora) также…

- Активно курируют онлайн-репозитории, чтобы удалить глючные, устаревшие или опасные пакеты.
- Устанавливают и удаляют программное обеспечение по запросу, незаметно обеспечивая работоспособность всех зависимостей.
- Управляют уже установленным программным обеспечением, автоматически обновляя его при необходимости.
- Чудесным образом восстанавливают состояние вашей системы после неудачных или прерванных сеансов управления. Я говорю «чудесным образом», потому что не могу найти другого слова, которое бы точно описало это.

Вот 30-секундное руководство по APT в Ubuntu. Во-первых, обновите локальный индекс, чтобы он знал, что доступно в upstream. Поскольку это повлияет на системные файлы, вам нужно будет использовать sudo.

```
$ sudo apt update
```

Если вы не уверены в точном названии пакета, который ищете, выполните поиск. Этот пример вернет очень длинный список кандидатов с некоторым подключением к Git, но, как оказалось, фактический пакет Git просто называется git. Примечание [installed] сообщает нам текущий статус пакета.

```
$ apt search git

git/xenial-updates,xenial-security,
        now 1:2.7.4-0ubuntu1.3 amd64 [installed]
        fast, scalable, distributed revision control system
```

Finally, if necessary, installing Git is as simple as…

```
$ sudo apt install git
```

8.4 SSH: безопасное удаленное подключение

Судя по вашему виду, я бы сказал, что вы делаете большую часть своей работы на ноутбуке. Но я бы также сказал, что, скорее всего, вы не планируете использовать свой ноутбук в качестве веб-сервера. Итак, как вы настроите свой сервер или переместите свои драгоценные приложения? Сетевое подключение, верно?

Звучит здорово. Но знаете ли вы, что любые данные, передаваемые по общедоступной сети, могут быть перехвачены, прочитаны и изменены? Все это, пароли и все такое. Если, конечно, данные не зашифрованы должным образом.

Лучший способ безопасного создания и шифрования удаленных подключений - использовать Secure Shell (SSH). Протокол настолько хорош и настолько популярен, что теперь он доступен не только там, где вы ожидаете увидеть его в Linux и macOS, но даже в Windows 10.

Вот как это работает. Обычно мы называем ваш портативный компьютер SSH-клиентом, а сервер, поскольку на нем будут размещены ваши сеансы, - SSH-сервером. Убедитесь, что на вашем сервере установлен пакет openssh-server, а на вашем ноутбуке либо openssh-server, либо openssh-client.

```
$ sudo apt update

$ sudo apt install openssh-server
```

Теперь, как показано на рисунке 8.2, открыть новый сеанс так же просто, как ввести ssh, имя пользователя, учетную запись которого вы будете использовать на сервере (мой пример - «ubuntu»), и IP-адрес или доменное имя сервера, а затем ввести пароль.

```
ssh ubuntu@10.0.3.141

ubuntu@10.0.3.141's password:

Welcome to Ubuntu 16.04.1 LTS (GNU/Linux 4.4.0-109-generic x86_64)

        * Documentation: https://help.ubuntu.com

        * Management: https://landscape.canonical.com

        * Support: https://ubuntu.com/advantage

Last login: Thu Dec 7 19:01:45 2017 from 10.0.3.1 ubuntu@server:~$
```

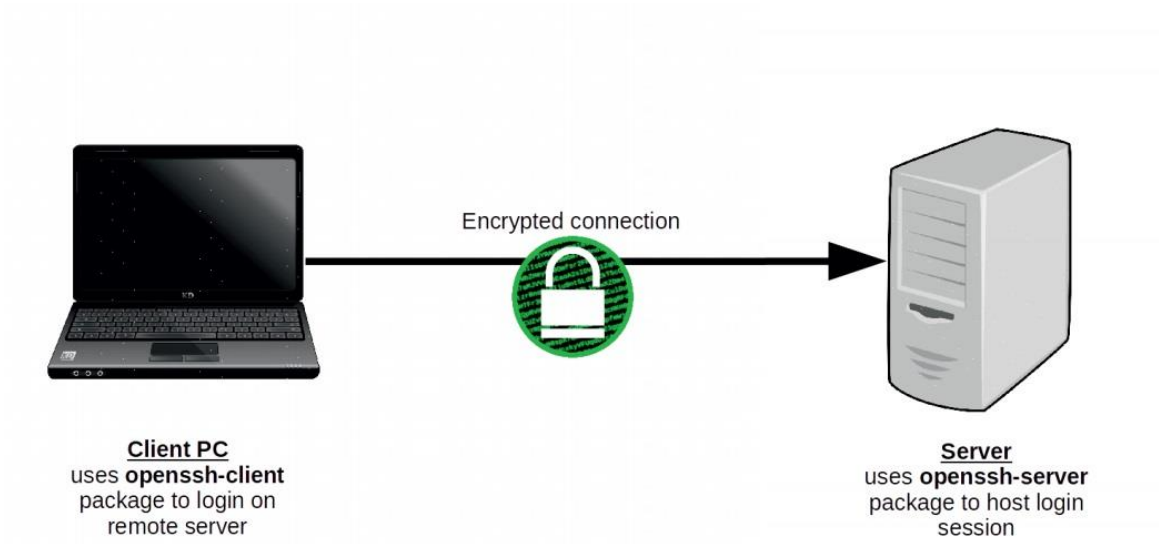

Рисунок 8.2. Вход на удаленный сервер через зашифрованное SSH-соединение.

8.4.1 Безопасная передача файлов

А как насчет того кода приложения, который находится на вашем ноутбуке? OpenSSH предлагает вам еще один инструмент: Secure Copy (SCP). Допустим, у вас есть архив Java JAR, который вы хотите передать. Вот что вы бы делали на своем ноутбуке:

```
$ scp /home/myusername/code/myapp.jar \

        ubuntu@10.0.3.141:/home/ubuntu/
```

Вам будет предложено ввести пароль для «ubuntu» на сервере, а затем файл JAR будет помещен в домашний каталог этого пользователя.

Как насчет копирования файлов в другом направлении? Предположим, вы работаете на своем ноутбуке и хотите переместить некоторые файлы журналов с сервера на ваш ноутбук. Что ж, предположим, что журналы в настоящее время находятся в домашнем каталоге пользователя ubuntu на сервере, вот как это будет работать:

```
$ scp ubuntu@10.0.3.141:/home/ubuntu/logfiles.zip .
```

Точка в конце указывает SCP сохранить файл в вашем текущем каталоге на вашем ноутбуке. Попробуйте сами.

8.5 Файловая система Linux

SSH - это ваш путь к вашему серверу Linux, программное обеспечение обрабатывается через APT или YUM, и вы навязываете свою имперскую волю, вызывая sudo. Все, что осталось, — это научиться заниматься администрированием Linux.

Запомните эту единственную мудрость, и вы никогда не ошибетесь:

> *Все в Linux — это простой текстовый файл.*

Хитрость заключается в том, чтобы просто найти нужные файлы и выяснить, что с ними делать. Итак, позвольте мне познакомить вас со стандартом иерархии файловой системы Unix / Linux.

8.5.1 Стандарт иерархии файловой системы Unix

За сравнительно небольшим исключением, все дистрибутивы Linux организуют файлы одинаково. В значительной степени вы найдете похожую схему в операционной системе macOS на базе Unix, о которой некоторые из вас, возможно, слышали. Вот как все выглядит, когда вы перечисляете содержимое типичного корневого каталога Linux:

```
$ cd /

$ ls

bin initrd.img mnt sbin usr

boot initrd.img.old opt snap var

dev lib proc srv vmlinuz

etc lib64 root sys vmlinuz.old

home media run tmp
```

Вот краткий обзор некоторых из этих каталогов:

- Каталог /etc/ содержит файлы конфигурации, которые определяют способ работы отдельных программ и служб.
- /var/ — это место, где Linux хранит файлы переменных, принадлежащие системе или отдельным приложениям. Это файлы, содержимое которых часто меняется в процессе нормальной работы системы.
- Отдельным пользователям предоставляются каталоги для их личных файлов в каталоге /home/.
- /lib/ содержит общие программные библиотеки.
- Бинарные файлы для большинства инструментов командной строки находятся в /bin/.

8.5.2 Популярные команды в Linux

Я уверен, что у вас не возникнет проблем с этим. Когда вы входите в Linux - или открываете новую оболочку терминала в сеансе графического интерфейса - вы обычно попадаете в домашний каталог вашего пользователя. Предполагая, что вашего пользователя зовут ubuntu, вот как будет выглядеть запущенный pwd («текущий рабочий каталог»):

```
$ pwd

/home/ubuntu
```

Вы можете создать новый каталог в вашем текущем местоположении с помощью mkdir, а затем перечислить содержимое с помощью ls:

```
$ mkdir newdir
```

```
$ ls

newdir
```

Хотите узнать, что находится в вашем новом каталоге? Перейдите к нему с помощью cd («сменить каталог»):

```
$ cd newdir
```

touch создаст новый пустой файл, а cp скопирует его в новое место - или просто сделает копию оригинала с новым именем, если вы не укажете другое местоположение.

```
$ touch newfile

$ cp newfile newerfile

$ ls

newerfile newfile
```

Устали от того, что все эти файлы занимают место на жестком диске и ничего не дают взамен? Покажите им, кто здесь главный, и удалите одного из них.

```
$ rm newerfile
```

В большинстве случаев текстовый редактор nano устанавливается по умолчанию. Вы можете использовать его для добавления или изменения содержимого файла. Когда вы закончите писать свою Великую книгу, нажмите CTRL + x, чтобы выйти, и y, чтобы сохранить файл.

```
$ nano newfile
```

Наконец, чтобы перейти в каталог из другого места в файловой системе, вам нужно будет указать его абсолютный адрес, возвращаясь к корневому каталогу (/). Кстати, в /var/log/ хранится большинство системных журналов. Не стесняйтесь осматриваться.

```
$ cd /var/log
```

8.5.3 Разрешения на объект

По замыслу Linux - многопользовательская система. Это означает, что к одному компьютеру с Linux можно получить доступ и использовать одновременно бесконечное количество пользователей и процессов. Это одна из многих функций, которые делают Linux настолько невероятно полезным: он обеспечивает уровень гибкости и совместной работы, которому нет равных в ИТ.

Но это также означает, что существует большая потребность в защите ресурсов от несанкционированного доступа. Вы же не хотите, чтобы одна команда разработчиков случайно испортила файлы конфигурации, принадлежащие другой команде. И, конечно же, никто не хочет, чтобы ребята из маркетинга ни с чем возились.

Таким образом, все объекты в файловой системе Linux будут иметь набор атрибутов, точно определяющих, кто и что должен делать. Эти атрибуты, известные как разрешения, разбиты на три категории: чтение (r), запись (w) и выполнение (x). Пользователи делятся на три класса: владелец объекта (u), группа объекта (g) и все остальные (o).

ls -l отобразит атрибуты объекта. В этом примере первая -rw сообщает нам, что владелец имеет разрешения на чтение и запись (т.е. Редактирование), группа также имеет разрешения на чтение и запись, а все остальные могут читать, но не могут записывать файл.

```
$ ls -l

-rw-rw-r-- 1 ubuntu ubuntu 0 Jan 14 00:33 newfile
```

Вы можете редактировать разрешения с помощью chmod. Этот пример добавит (+) права на запись для других (o).

```
$ chmod o+w

$ ls -l

-rw-rw-rw- 1 ubuntu ubuntu 0 Jan 14 00:33 newfile
```

Вы также можете использовать chown, чтобы изменить владельца объекта. Прямо сейчас, как вы можете видеть, файл принадлежит пользователю ubuntu и принадлежит группе ubuntu. Эта команда передаст право собственности учетной записи пользователя с именем steve, оставив объект в группе ubuntu.

```
$ sudo chown steve:ubuntu newfile

$ ls -l

-rw-rw-rw- 1 steve ubuntu 0 Jan 14 00:33 newfile
```

8.5.4 Помощь

Когда вы столкнетесь с проблемой, ваша первая остановка, вероятно, будет Google… и не зря. Но не игнорируйте огромное количество полезной информации, доступной вам в командной строке Linux через систему man. Набрав man с последующим названием команды, на которой вы застряли, откроется хорошо организованное руководство по использованию этой команды.

```
$ man chmod
```

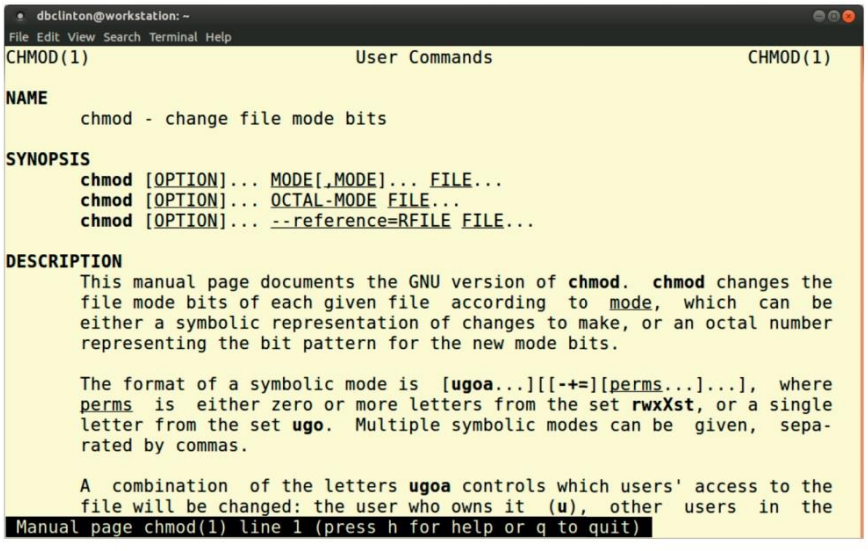

Рисунок 8.3. Первый экран из файла man для chmod.

Глава 9. Приложение 3: Amazon Web Services

Что это за AWS? Вот как я описываю это в своей книге «Learn Amazon Web Services in a Month of Lunches»:

> Предположим, вы не хотите покупать, строить и размещать все дорогое оборудование, необходимое для надлежащей поддержки вашего бизнеса. Возможно, вы не уверены, насколько успешным будет проект, поэтому вкладывать большие средства в серверное, охлаждающее и маршрутизирующее оборудование не имеет смысла. Но если бы вы могли арендовать достаточно чужого оборудования, чтобы соответствовать быстро меняющимся взлетам и падениям спроса ваших услуг, и платить только за то, что вы фактически использовали, тогда это могло бы сработать.
>
> Есть ли кто-нибудь, кто мог бы предоставить вам в аренду такие вещи? Есть. Их называют поставщиками облачных вычислений. А Amazon Web Services (AWS) на сегодняшний день является самым многофункциональным и успешным игроком на рынке (с доходом около 12 миллиардов долларов в 2016 году).
>
> По данным исследовательской компании Gartner (http://mng.bz/31dr), AWS «имеет самую большую долю вычислительных мощностей, используемых платными клиентами - во много раз больше, чем у всех других поставщиков на рынке ... множество возможностей IaaS и PaaS. Он предоставляет самые широкие возможности для управления большим количеством пользователей и ресурсов. Он продолжает быстро расширять список своих услуги и предлагать решения более высокого уровня».
>
> AWS зарекомендовала себя как способная предоставить безопасную и надежную платформу хостинга как для критически важных (электронная коммерция, корпоративные интрасети), так и для приложений с интенсивными вычислениями (аналитика больших данных, среды разработки). Учитывая его глубину, если ваш проект можно виртуализировать в той или иной форме, то, по мнению Gartner, AWS является «безопасным выбором».

ОК. Итак, как это работает? Что ж, это зависит от того, чего вы хотите достичь и сколько времени и энергии вы готовы инвестировать заранее.

Видите ли, было бы довольно просто точно воспроизвести вашу локальную инфраструктуру с использованием ресурсов AWS с относительно мягкой кривой обучения. По сути, все, что вы можете делать на своих серверах без операционной системы, вы можете делать на AWS. Иногда даже можно создать архив вашего приложения и баз данных и перенести их в облако с помощью одной команды.

Но так вы не всегда получаете максимальную отдачу от вложенных средств. Чтобы по-настоящему выжать из предоставляемых вами вычислительных экземпляров и баз данных AWS, вам обычно нужно изменить форму приложения, чтобы воспользоваться преимуществами встроенной масштабируемости и эластичности облака. Если копнуть глубже, вы можете обнаружить, что вам даже не нужны вычислительные ресурсы или экземпляры базы данных, но вместо этого вы можете запускать свое приложение «в воздухе» - без сервера.

В этой главе я даже не будет пытаться научить вас всему этому - прочтите мою книгу Мэннинга «Learn Amazon Web Services in a Month of Lunches», чтобы узнать обо всем этом. Скорее я попытаюсь познакомить вас с архитектурными принципами, которые определяют успешные рабочие нагрузки AWS.

9.1 Перенос рабочих нагрузок

Учетная запись AWS предоставляет вам все виртуальные вычислительные экземпляры, которые вам могут понадобиться, через их сервис Elastic Compute Cloud (EC2). Как показано на рисунке 9.1, вы можете выбрать предустановленную операционную систему, которую хотите подготовить, или создать свои собственные образы машин. Вы также сможете установить тип экземпляра (профиль оборудования, который вы хотите запустить), сетевую среду, ресурсы хранения и настройки брандмауэра (называемые группой безопасности). Общее время от запуска до запуска: ~ 30 секунд при использовании значений по умолчанию.

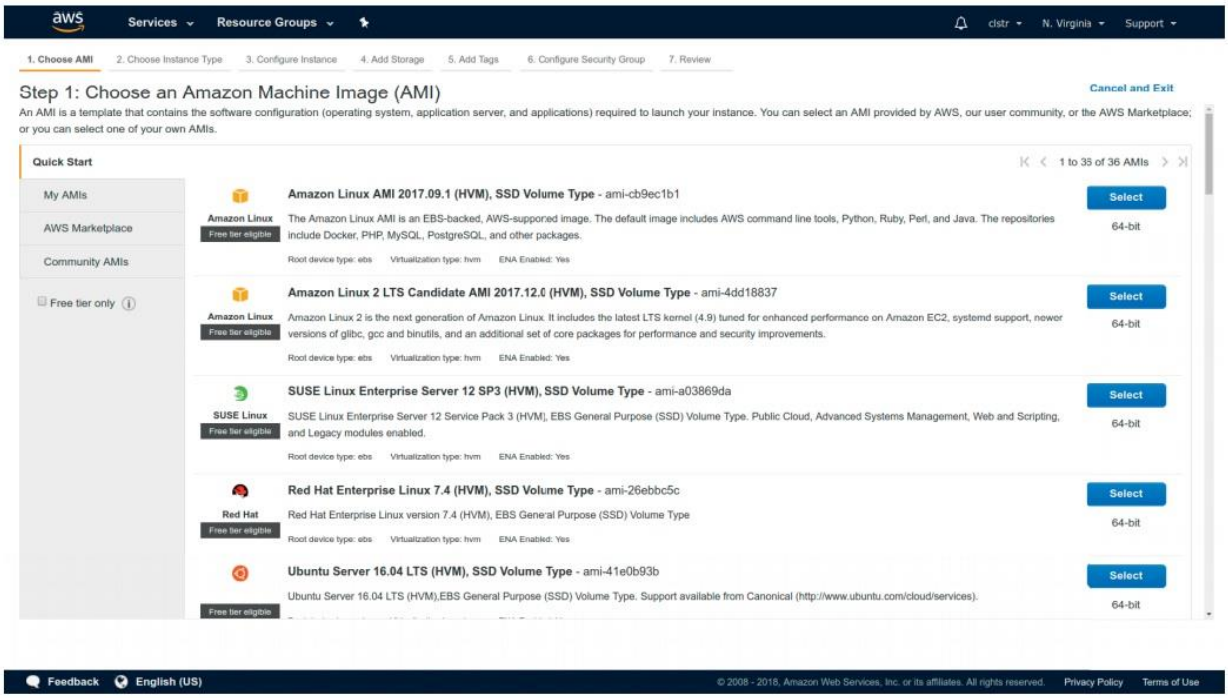

Рисунок 9.1. Выберите предустановленный образ операционной системы для нового экземпляра AWS EC2, который впоследствии можно будет запустить.

После того, как ваш сервер запущен, вы можете получить к нему удаленный доступ - обычно через SSH - и приступить к запуску вашего приложения. Следите за событиями во время тестирования/переходного периода, чтобы увидеть, что может потребоваться изменить.

- Ваш экземпляр слишком слабый, чтобы выдержать нагрузку? Обновите его в несколько кликов.
- Использует ли ваше приложение много больших медиафайлов? Храните их с меньшими затратами на Amazon Simple Storage Service (S3).
- Ваш сервер бездействует по много часов каждый день? Выключите его и запланируйте запуск на тот момент, когда он вам понадобится ... и сэкономите при этом кучу денег.

Простые серверы построить совсем не сложно. Но оптимизация более сложной инфраструктуры — это своего рода наука.

9.2 Рефакторинг и оптимизация облачных рабочих нагрузок

Чтобы понять, как эффективно настраивать свои облачные ресурсы, необходимо твердое понимание двух концепций:

> **Эластичность** — это способность системы отслеживать потребности пользователей и соответственно автоматически увеличивать, и уменьшать развернутые ресурсы. Это качество - важная составляющая секрета облачных вычислений, поскольку именно оно позволяет проектировать и развертывать приложения. Вы можете быть уверены, что облачный провайдер будет незаметно выделять или выводить ресурсы из эксплуатации в соответствии с потребностями, гарантируя, что вы никогда не застрянете со счетом за неиспользованные услуги.
>
> **Масштабируемость** описывает способ, которым система спроектирована для удовлетворения меняющегося спроса. Это означает, что основной дизайн поддерживает быстрые, непредсказуемые изменения. Например, масштабируемое программное обеспечение можно легко перенести на новый сервер и запустить без какой-либо ручной настройки. Или, другими словами, состав масштабируемой инфраструктуры можно быстро изменить таким образом, чтобы все старые части сразу знали, как работать вместе с новыми.

В AWS это может включать развертывание ваших серверов за Elastic Load Balancer (ELB). Балансировщик направляет входящие запросы к экземплярам, работающим как часть группы Auto Scaling. Группа, в свою очередь, автоматизирует процесс изменения количества запущенных экземпляров вверх и вниз для удовлетворения спроса. На рисунке 9.2 показана страница политики, часть процесса настройки автоматического масштабирования. Здесь вы определяете события, которые должны выполнять запуск новых экземпляров. В этом случае, когда существующие серверы достигают 80% мощности ЦП, будет запущена новый экземпляр.

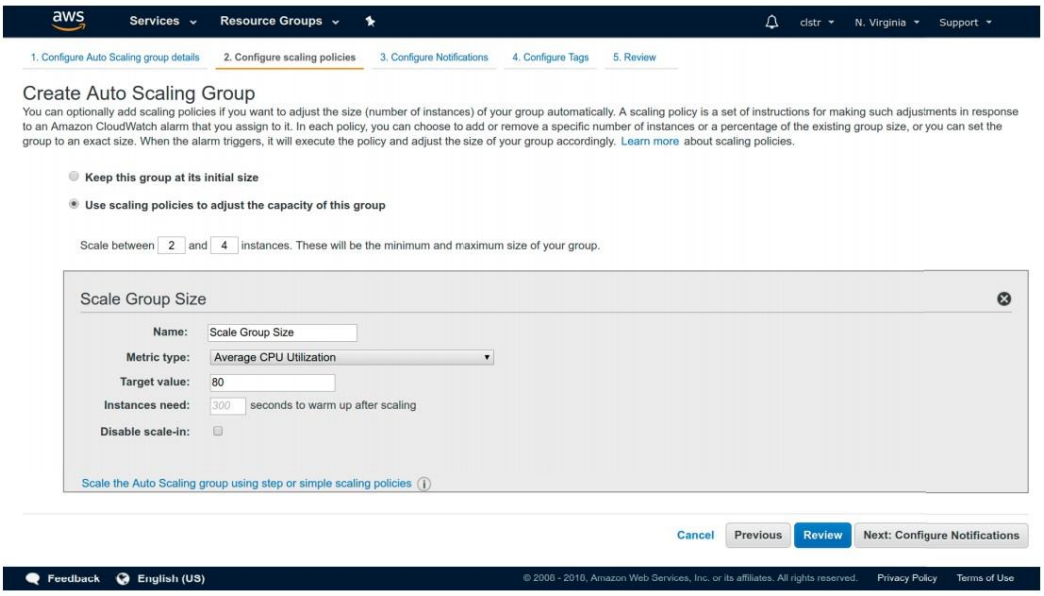

Рисунок 9.2. Группа масштабирования автоматически запускает или удаляет предварительно настроенные экземпляры в соответствии с изменяющимся спросом.

9.3 Serverless

Вместо того, чтобы арендовать весь экземпляр виртуального сервера, как это было бы через EC2, в «serverless» мире вы арендуете всего пару секунд на одном из собственных серверов AWS. Поскольку механика всего этого менее заметна, мы называем этот процесс «serverless».

Основная реализация serverless вычислений на AWS называется Lambda. В настоящее время она позволяет создавать функции на Node.js, C#, Java и Python. Lambda глубоко интегрирована с широким спектром сервисов AWS - как в качестве триггеров для вызова и функции Lambda, так и в качестве ресурсов, которые будут использоваться как часть функции.

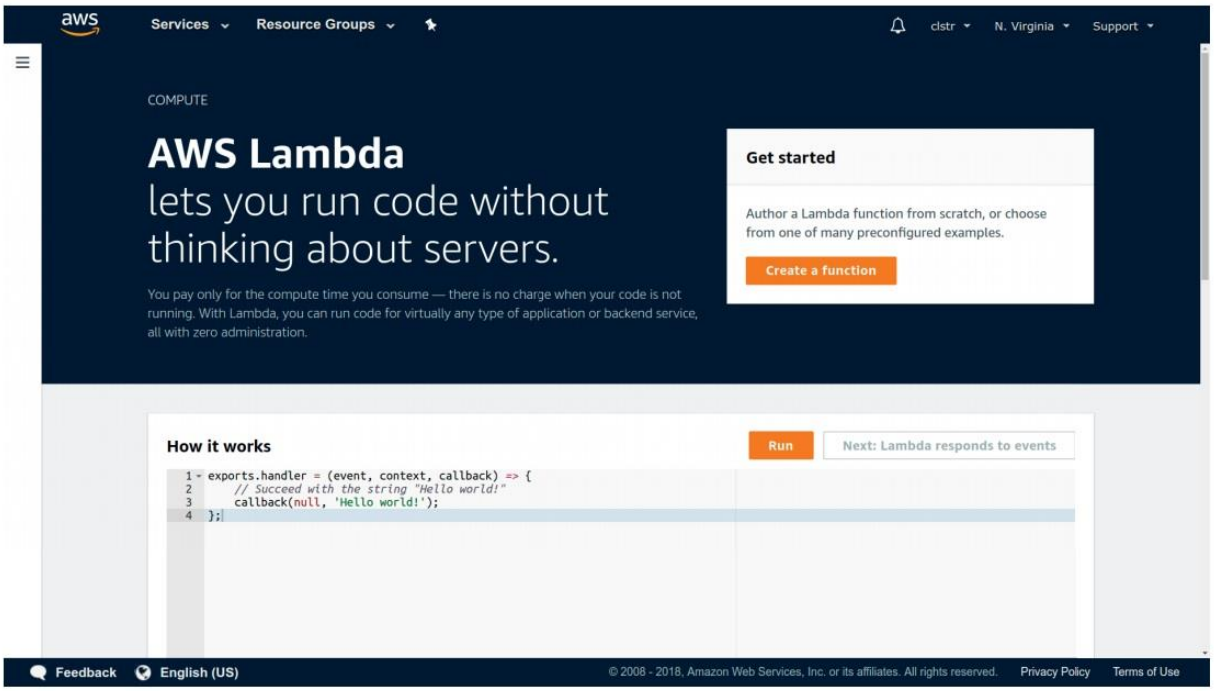

Рисунок 9.3: Введение в AWS Lambda. Эта кнопка Run фактически запустит живую функцию.

Идея состоит в том, что события, происходящие в других сервисах AWS, можно настроить так, чтобы запускать быстрое выполнение функции, используя только минимум облачных ресурсов, необходимых для запуска действия. Классическим примером для иллюстрации того, как это работает, является старый трюк создания эскиза из файла изображения; и действительно, кому не нужно время от времени создавать миниатюры из изображений?

Вот как это происходит: всякий раз, когда клиент вашего онлайн-хранилища изображений сохраняет фотографию в вашу корзину S3, активируется функция Lambda, создает небольшую версию изображения в виде эскиза, сохраняет ее во второй корзине и связывает два изображения поэтому эскиз будет отображаться как ссылка на оригинал на веб-сайтах. Вам не нужно запускать (и оплачивать) весь инстанс EC2 24 часа в сутки на случай, если появятся чьи-то свадебные фотографии. Вместо этого вы запускаете именно то, что вам нужно, именно тогда, когда вам это нужно.

Глава 10. Завершая все это

Итак, serverless, AWS, технологии «больших палаток» … и целый ряд инструментов и приемов для успешного достижения всех ваших целей в учебе. Моя часть уже сделана.

Но я приглашаю вас продолжить с того места, где я остановился. Вы видите что-то, что я упустил, или ошибку, большую или маленькую? Пожалуйста, дайте мне знать. Интернет полезен настолько, насколько полезным делаем его мы.

Знаете кого-нибудь, кому может понравиться эта книга? Пожалуйста, передайте её - в цифровом или печатном виде. Хотите поделиться своими мыслями о книге? Подумайте о том, чтобы написать обзор на Amazon.

И будьте на связи.

David Clinton bootstrap-it.com office@bootstrap-it.com