Введение

Давным-давно в далеком-предалеком вычислительном центре древнее племя могущественных существ, известных как «сисадмины», вручную развертывало инфраструктуру. Каждый сервер, база данных (БД), балансировщик нагрузки и фрагмент сетевой конфигурации создавались и управлялись вручную. Это было мрачное и ужасное время: страх простоя, случайной ошибки в конфигурации, медленных и хрупких развертываний и того, что может произойти, если сисадмины перейдут на темную сторону (то есть возьмут отпуск). Но спешу вас обрадовать — благодаря движению DevOps у нас теперь есть замечательный инструмент для администрирования: *Terraform*.

Terraform (https://www.terraform.io/) — это инструмент с открытым исходным кодом от компании HashiCorp. Он позволяет описывать инфраструктуру в виде кода на простом декларативном языке и развертывать/администрировать ее в различных публичных облачных сервисах (скажем, Amazon Web Services, Microsoft Azure, Google Cloud Platform, DigitalOcean), а также закрытых облаках и платформах виртуализации (OpenStack, VMWare и др.) всего несколькими командами. Например, вместо того, чтобы вручную щелкать кнопкой мыши на веб-странице или вводить десятки команд в консоль, вы можете воспользоваться следующим кодом и сконфигурировать сервер в AWS:

```
provider "aws" {
  region = "us-east-2"
}

resource "aws_instance" "example" {
  ami           = "ami-0c55b159cbfafe1f0"
  instance_type = "t2.micro"
}
```

Чтобы его развернуть, введите следующее:

```
$ terraform init
$ terraform apply
```

Благодаря своей простоте и мощи Terraform стал ключевым игроком в мире DevOps. Он позволяет заменить громоздкие, хрупкие и неавтоматизированные средства для управления инфраструктурой надежным автоматизированным инструментом, поверх которого вы можете объединить все остальные элементы

DevOps (автоматическое тестирование, непрерывную интеграцию и непрерывное развертывание) и сопутствующий инструментарий (например, Docker, Chef, Puppet).

Прочитайте эту книгу, и вы сможете сразу приступить к работе с Terraform.

Изучив всего несколько глав, вы пройдете путь от простейшего примера «Hello, World» (на самом деле вы только что видели его!) до развертывания полного технологического стека (виртуальных серверов, кластера серверов, балансировщиков нагрузки, базы данных), рассчитанного на огромный трафик и крупную команду разработчиков. Это практическое руководство не только научит принципам DevOps и инфраструктуры как кода (infrastructure as code, IaC), но и проведет вас через десятки примеров кода, которые можно опробовать дома. Поэтому держите свой компьютер под рукой.

Дочитав книгу, вы будете готовы к работе с Terraform в реальных условиях.

Целевая аудитория книги

Книга предназначена для всех, кто отвечает за уже написанный код. Это относится к сисадминам, специалистам по эксплуатации, релиз-инженерам, инженерам по мониторингу, инженерам DevOps, разработчикам инфраструктуры, разработчикам полного цикла, руководителям инженерной группы и техническим директорам. Какой бы ни была ваша должность, если вы занимаетесь инфраструктурой, развертываете код, конфигурируете серверы, масштабируете кластеры, выполняете резервное копирование данных, мониторите приложения и отвечаете на вызовы в три часа ночи, эта книга для вас.

В совокупности эти обязанности обычно называют операционной деятельностью (или системным администрированием). Раньше часто встречались разработчики, которые умели писать код, но не разбирались в системном администрировании; точно так же нередко попадались сисадмины без умения писать код. Когда-то такое разделение было приемлемым, но в современном мире, который уже нельзя представить без облачных вычислений и движения DevOps, практически любому разработчику необходимы навыки администрирования, а любой сисадмин должен уметь программировать.

Для чтения этой книги не обязательно быть специалистом в той или иной области — поверхностного знакомства с языками программирования, командной строкой и серверным программным обеспечением (сайтами) должно хватить. Всему остальному можно научиться в процессе. Таким образом, по окончании чтения вы будете уверенно разбираться в одном из важнейших аспектов современной разработки и системного администрирования: в управлении инфраструктурой как кодом.

Вы не только научитесь управлять инфраструктурой в виде кода, используя Terraform, но и узнаете, как это вписывается в общую концепцию DevOps. Вот несколько вопросов, на которые вы сможете ответить по прочтении этой книги.

- Зачем вообще использовать IaC?
- Какая разница между управлением конфигурацией, оркестрацией, выделением ресурсов и шаблонизацией серверов?
- Когда следует использовать Terraform, Chef, Ansible, Puppet, Pulumi, CloudFormation, Docker, Packer или Kubernetes?
- Как работает система Terraform и как с ее помощью управлять инфраструктурой?
- Как создавать многократно используемые модули Terraform?
- Как безопасно управлять конфиденциальной информацией при работе с Terraform?
- Как использовать Terraform с несколькими регионами, учетными записями и облаками?
- Как писать код для Terraform, который будет достаточно надежным для практического применения?
- Как тестировать свой код для Terraform?
- Как внедрить Terraform в свой процесс автоматического развертывания?
- Как лучше всего использовать Terraform в командной работе?

Вам понадобятся лишь компьютер (Terraform поддерживает большинство операционных систем), интернет-соединение и желание учиться.

Почему я написал эту книгу

Terraform — мощный инструмент, совместимый со всеми популярными облачными провайдерами. Он основан на простом языке, позволяет повторно использовать код, выполнять тестирование и управлять версиями. Это открытый проект с дружелюбным и активным сообществом. Но, надо признать, он еще не до конца сформирован.

Terraform — относительно новая технология. Несмотря на ее популярность, по-прежнему сложно найти книги и статьи или встретить специалистов, которые бы помогли вам овладеть этим инструментом. Официальная документация Terraform хорошо подходит для знакомства с базовым синтаксисом и возможностями, но в ней мало информации об идиоматических шаблонах, рекомендуемых методиках, тестировании, повторном использовании кода и рабочих процессах

в команде. Это как пытаться овладеть французским языком с помощью одного лишь словаря, игнорируя грамматику и идиомы.

Я написал эту книгу, чтобы помочь разработчикам изучить Terraform. Я пользуюсь этим инструментом четыре года из пяти с момента его создания — в основном в моей компании Gruntwork (http://www.gruntwork.io). Там он сыграл ключевую роль в создании библиотеки более чем из 300 000 строк проверенного временем инфраструктурного кода, готового к повторному использованию и уже применяемого сотнями компаний в промышленных условиях. Написание и поддержка такого большого объема инфраструктурного кода на таком длинном отрезке времени в таком огромном количестве разных компаний и сценариев применения позволило нам извлечь много непростых уроков. Я хочу поделиться с вами ими, чтобы вы могли сократить этот долгий процесс и овладеть Terraform в считаные дни.

Конечно, одним чтением этого не добьешься. Чтобы начать свободно разговаривать на французском, придется потратить какое-то время на общение с носителями языка, просмотр французских телепередач и прослушивание французской музыки. Чтобы овладеть Terraform, нужно написать для этой системы настоящий код, использовать его в реальном ПО и развернуть это ПО на настоящих серверах. Поэтому приготовьтесь к чтению, написанию и выполнению большого количества кода.

Структура издания

Вот список тем, которые освещаются в книге.

- *Глава 1 «Почему Terraform»*. Как DevOps меняет наш подход к выполнению ПО; краткий обзор инструментов IaC, включая управление конфигурацией, шаблонизацию серверов, оркестрацию и выделение ресурсов; преимущества IaC; сравнение Terraform, Chef, Puppet, Ansible, Pulumi, OpenStack Heat и CloudFormation; как сочетать такие инструменты, как Terraform, Packer, Docker, Ansible и Kubernetes.
- *Глава 2 «Приступаем к работе с Terraform»*. Установка Terraform; краткий обзор синтаксиса Terraform; обзор утилиты командной строки Terraform; как развернуть один сервер; как развернуть веб-сервер; как развернуть кластер веб-серверов; как развернуть балансировщик нагрузки; как очистить созданные вами ресурсы.
- *Глава 3 «Как управлять состоянием Terraform»*. Что такое состояние Terraform; как хранить состояние, чтобы к нему имели доступ разные члены команды; как блокировать файлы состояния, чтобы предотвратить состояние гонки; как изолировать файлы состояния, чтобы смягчить последствия ошибок;

как использовать рабочие области Terraform; рекомендуемая структура каталогов для проектов Terraform; как работать с состоянием, доступным только для чтения.

- *Глава 4 «Многократное использование инфраструктуры с помощью модулей Terraform»*. Что такое модули; как создать простой модуль; как сделать модуль конфигурируемым с помощью входящих и исходящих значений; локальные переменные; версионирование модулей; потенциальные проблемы с модулями; использование модулей для описания настраиваемых элементов инфраструктуры с возможностью повторного применения.
- *Глава 5 «Работа с Terraform: циклы, условные выражения, развертывание и подводные камни»*. Циклы с параметром `count`, выражения `for_each` и `for`, строковая директива `for`; условный оператор с параметром `count`, выражениями `for_each` и `for`, строковой директивой `if`; встроенные функции; развертывание с нулевым временем простоя; часто встречающиеся подводные камни, связанные с ограничениями `count` и `for_each`, развертываниями без простоя; как хорошие планы могут провалиться и как безопасно осуществлять рефакторинг кода Terraform.
- *Глава 6 «Управление конфиденциальными данными»*. Введение в управление секретами; обзор различных типов секретов, разных способов их хранения и доступа к ним; сравнение распространенных инструментов управления секретами, таких как HashiCorp Vault, AWS Secrets Manager и Azure Key Vault; как управлять секретами при работе с провайдерами, включая аутентификацию через переменные окружения, роли IAM и OIDC; управление секретами при работе с ресурсами и источниками данных, в том числе с использованием переменных окружения, зашифрованных файлов и централизованных хранилищ секретов; безопасное обращение с файлами состояния и файлами планов.
- *Глава 7 «Работа с несколькими провайдерами»*. Подробный обзор особенностей работы с провайдерами, поддерживающими Terraform, в том числе порядок выбора и установки определенной версии и ее использование в коде. Применение нескольких копий одного и того же провайдера, в том числе развертывание в нескольких регионах AWS, развертывание в нескольких учетных записях AWS и создание повторно используемых модулей, способных использовать несколько провайдеров. Как задействовать несколько разных провайдеров вместе, например, для запуска кластера Kubernetes (EKS) в AWS и развертывания контейнеров Docker в кластере.
- *Глава 8 «Код Terraform промышленного уровня»*. Почему проекты DevOps всегда развертываются дольше, чем ожидается; что характеризует инфраструктуру, готовую к промышленному использованию; как создавать модули Terraform для промышленного применения; небольшие модули; сборные модули; тестируемые модули; готовые к выпуску модули; реестр Terraform;

проверка переменных; управление версиями Terraform, провайдеров Terraform, модулей Terraform и Terragrunt; «аварийные люки» в Terraform.

- *Глава 9 «Как тестировать код Terraform»*. Ручное тестирование кода Terraform; создание и удаление тестового окружения; автоматизированное тестирование кода Terraform; Terrarest; модульные тесты; интеграционные тесты; сквозные тесты; внедрение зависимостей; параллельное выполнение тестов; этапы тестирования; попытки; пирамида тестирования; статический анализ; план тестирования; тестирование сервера.
- *Глава 10 «Как использовать Terraform в команде»*. Как внедрить Terraform в командную работу; как убедить начальство; рабочий процесс развертывания прикладного кода; рабочий процесс развертывания инфраструктурного кода; управление версиями; золотое правило Terraform; рецензирование кода; рекомендации по оформлению кода; принятый в Terraform стиль; CI/CD для Terraform; процесс развертывания.

Эту книгу можно читать последовательно или сразу переходить к тем главам, которые вас больше всего интересуют. Имейте в виду, что все примеры последующих глав основаны на коде из предыдущих. Если вы листаете туда-сюда, используйте в качестве ориентира архив исходного кода (как описано в разделе «Примеры с открытым исходным кодом» далее). В приложении вы найдете список книг и статей о Terraform, системном администрировании, IaC и DevOps.

Что нового в третьем издании

Первое издание этой книги вышло в 2017 году, второе вышло в 2019 году, и хотя мне сложно в это поверить, но сейчас я работаю уже над третьим изданием. Время летит. Удивительно, как много изменилось за эти годы!

Если вы читали второе издание книги и хотите узнать, что нового появилось в этом издании, или вам просто интересно увидеть, как Terraform развивался между 2019 и 2022 годами, то ниже перечислены некоторые основные различия между вторым и третьим изданиями.

- *Сотни страниц обновленного содержания*. Третье издание книги примерно на 100 страниц объемнее второго. Также, по моим оценкам, примерно от трети до половины страниц второго издания были обновлены. Почему так много новой информации? Потому что с момента выхода второго издания вышло шесть новых версий Terraform со значительными изменениями: 0.13, 0.14, 0.15, 1.0, 1.1 и 1.2. Более того, многие провайдеры Terraform провели собственные серьезные обновления, в том числе AWS, который на момент выхода второго издания был представлен версией 2, а к моменту выхода третьего — версией 4. Кроме того, за последние несколько лет значительно

выросло сообщество Terraform, что привело к появлению множества новых приемов, инструментов и модулей. Я постарался отразить как можно больше этих изменений, добавив две новые главы и внеся существенные обновления в остальные, как описано ниже.

- *Новые функциональные возможности провайдеров.* В Terraform значительно улучшена поддержка провайдеров. В третье издание я добавил совершенно новую главу (главу 7), где рассказывается, как работать с несколькими провайдерами: например, как организовать развертывание в нескольких регионах, нескольких учетных записях и нескольких облаках. Кроме того, по многочисленным просьбам в эту главу включен совершенно новый набор примеров, показывающих, как использовать Terraform, Kubernetes, Docker, AWS и EKS для запуска контейнерных приложений. Наконец, я обновил все остальные главы, чтобы осветить новые функциональные возможности провайдеров, появившиеся в последних нескольких версиях, включая блок `require_providers`, добавленный в Terraform 0.13 (предлагает лучший способ установки и версионирования официальных и неофициальных провайдеров Terraform, а также управления ими); файлы блокировки, внедренные в Terraform 0.14 (помогает гарантировать использование одних и тех же версий провайдеров всеми членами вашей команды) и параметр `configuration_aliases`, появившийся в Terraform 0.15 (улучшает управление провайдерами внутри модулей).
- *Улучшенное управление секретами.* При использовании кода Terraform часто приходится иметь дело со многими видами секретов: паролями баз данных, ключами API, учетными данными облачного провайдера, сертификатами TLS и т. д. Поэтому я добавил в третье издание новую главу (главу 6), посвященную этой теме, где сравниваются распространенные инструменты управления секретами и представлены примеры кода, иллюстрирующие разные методы безопасного использования секретов в Terraform, включая переменные окружения, зашифрованные файлы, централизованные хранилища секретов, роли IAM, OIDC и многое другое.
- *Новые функциональные возможности модулей.* В Terraform 0.13 появилась возможность использовать `count`, `for_each` и `depend_on` в блоках `module`, что делает модули более мощными, гибкими и пригодными для повторного применения. Примеры использования этих новых возможностей вы найдете в главах 5 и 7.
- *Новые возможности проверки.* В главе 8 я добавил примеры использования функции `validation`, появившейся в Terraform 0.13 и выполняющей простые проверки переменных (например, превышение минимальных или максимальных значений), а также функций `precondition` и `postcondition`, появившихся в Terraform 1.2 и выполняющих простые проверки ресурсов и источников данных перед запуском `apply` (например, для принудительного использования AMI при выборе пользователем архитектуры x86_64) либо после запуска `apply`

(например, проверка успешности шифрования тома EBS). В главе 6 я покажу, как использовать параметр `sensitive` (добавлен в Terraform 0.14 и 0.15), не позволяющий выводить секреты при запуске `plan` или `apply`.

- Новые возможности рефакторинга. В Terraform 1.1 появился блок `moved`, обеспечивающий лучший способ выполнения таких операций рефакторинга, как переименование ресурса. Раньше для этого пользователь должен был вручную запускать операции `terraform state mv`, а теперь, как показано в новом примере в главе 5, этот процесс можно полностью автоматизировать, что делает обновление модулей более безопасным и удобным.
- *Дополнительные возможности тестирования*. Инструменты автоматического тестирования кода Terraform продолжают совершенствоваться. В главе 9 я добавил пример кода и привел сравнение инструментов статического анализа, включая `tfsec`, `tflint`, `terrascan` и `validate`, инструментов тестирования `plan`, включая Terratest, OPA и Sentinel, а также инструментов тестирования серверов, включая `inspec`, `serverspec` и `goss`. Я также добавил сравнение существующих подходов к тестированию, чтобы вы могли выбрать наиболее подходящий для вас.
- *Улучшенная стабильность*. Выпуск версии Terraform 1.0 стал важной вехой для Terraform, означающей не только достижение определенного уровня зрелости, но и гарантии совместимости — все выпуски 1.x будут обратно совместимы, поэтому обновление между выпусками v1.x больше не должно требовать изменений кода, рабочих процессов или файлов состояния. Файлы состояния Terraform теперь перекрестно совместимы с Terraform 0.14, 0.15 и всеми выпусками 1.x, а источники данных удаленного состояния Terraform перекрестно совместимы с Terraform 0.12.30, 0.13.6, 0.14.0, 0.15.0 и всеми версиями 1.x. Я также обновил главу 8, добавив примеры, показывающие как лучше управлять версиями Terraform (включая использование `tfenv`), Terragrunt (включая использование `tgswitch`) и провайдеров Terraform (включая использование файла блокировки).
- *Возросшая зрелость*. Terraform был загружен более 100 миллионов раз, насчитывает более 1500 участников и используется почти в 79 % компаний из списка Fortune 500[1], поэтому можно с уверенностью сказать, что экосистема значительно выросла за последние несколько лет и стала более зрелой. Сейчас проект Terraform насчитывает больше разработчиков, провайдеров, повторно используемых модулей, инструментов, плагинов и классов. Появилось больше книг и учебных пособий. Кроме того, HashiCorp, компания, создавшая Terraform, провела IPO (первичное публичное размещение акций) в 2021 году, поэтому теперь Terraform больше не является небольшим

[1] Согласно HashiCorp S1 (https://www.sec.gov/Archives/edgar/data/0001720671/000119312521319849/d205906ds1.htm).

стартапом — это крупная и устойчивая компания, продающая свои акции, для которой Terraform является крупнейшим предложением.

- Множество других изменений. Кроме перечисленного выше, произошло множество других изменений, включая запуск Terraform Cloud (веб-интерфейс для использования Terraform); возросшую зрелость инструментов, популярных в сообществе, таких как Terragrunt, Terratest и tfenv; добавление множества новых возможностей провайдеров (в том числе новых способов развертывания с нулевым временем простоя, таких как обновление экземпляра, о чем рассказывается в главе 5); появление новых функций (в частности, в главе 5 я добавил примеры использования функции `one` и в главе 7 — функции `try`); прекращение поддержки многих старых возможностей (например, источника данных `template_file`, многих параметров `aws_s3_bucket`, `list` и `map`, внешних ссылок в `destroy`) и многое другое.

Что нового во втором издании

Возвращаясь еще раньше в прошлое, замечу, что второе издание книги стало примерно на 150 страниц больше первого издания. Вот краткий перечень этих изменений, иллюстрирующий, как эволюционировал проект Terraform в период с 2017 по 2019 год.

- *Четыре больших выпуска Terraform.* Когда вышла первая книга, стабильной версией Terraform была 0.8. Спустя четыре основных выпуска Terraform имеет версию 0.12. За это время появились некоторые поразительные новшества, о которых пойдет речь далее. Чтобы обновиться, пользователям придется попотеть![1]
- *Улучшения в автоматическом тестировании.* Существенно эволюционировали методики и инструментарий написания автоматических тестов для кода Terraform. Во втором издании я добавил новую главу (глава 9 в этом издании), затрагивающую такие темы, как модульное, интеграционное и сквозное тестирование, внедрение зависимостей, распараллеливание тестов, статический анализ и др.
- *Улучшения в модулях.* Инструментарий и методики создания модулей Terraform тоже заметно эволюционировали. Во втором издании я добавил новую главу (глава 8 в этом издании), где вы найдете руководство по написанию испытанных модулей промышленного уровня с возможностью повторного использования — таких, которым можно доверить благополучие своей компании.

[1] Подробности ищите в руководствах по обновлению Terraform по адресу https://www.terraform.io/upgrade-guides/index.html.

- *Улучшения в рабочем процессе*. Глава 8 (в этом издании — глава 10) была полностью переписана согласно тем изменениям, которые произошли в процедуре интеграции Terraform в рабочий процесс команд. Там, помимо прочего, можно найти подробное руководство, описывающее основные этапы создания прикладного и инфраструктурного кода: разработку, тестирование и развертывание в промышленной среде.
- *HCL2*. В Terraform 0.12 внутренний язык HCL обновился до HCL2. Это включает в себя поддержку полноценных выражений, развитые ограничители типов, условные выражения с отложенным вычислением, поддержку выражений `null`, `for_each` и `for`, встроенные блоки и др. Все примеры кода во втором издании были адаптированы для HCL2, а новые возможности языка подробно рассматриваются в главах 5 и 6 (в этом издании — глава 8).
- *Переработанные механизмы хранения состояния*. В Terraform 0.9 появилась концепция внутренних хранилищ. Это полноценный механизм хранения и разделения состояния Terraform со встроенной поддержкой блокирования. В Terraform 0.9 также были представлены окружения состояния, которые позволяют управлять развертываниями в разных средах; но уже в версии 0.10 им на смену пришли рабочие области. Все эти темы рассматриваются в главе 3.
- *Вынос провайдеров из ядра Terraform*. В Terraform 0.10 из ядра был вынесен код для всех провайдеров (то есть код для AWS, GCP, Azure и т. д.). Благодаря этому разработка провайдеров теперь ведется в отдельных репозиториях, в своем собственном темпе и с выпуском независимых версий. Однако теперь придется загружать код провайдера с помощью команды `terraform init` каждый раз, когда вы начинаете работать с новым модулем. Об этом пойдет речь в главах 2 и 7 (в этом издании — глава 9).
- *Большое количество новых провайдеров*. В 2016 году проект Terraform официально поддерживал лишь несколько основных облачных провайдеров (AWS, GCP и Azure). Сейчас же их количество превысило 100, а провайдеров, разрабатываемых сообществом, и того больше[1]. Благодаря этому вы можете использовать код для работы не только с множеством разных облаков (например, теперь существуют провайдеры для Alicloud, Oracle Cloud Infrastructure, VMware vSphere и др.), но и с другими аспектами окружающего мира, включая системы управления версиями (GitHub, GitLab или BitBucket), хранилища данных (MySQL, PostreSQL или InfluxDB), системы мониторинга и оповещения (включая DataDog, New Relic или Grafana), платформы наподобие Kubernetes, Helm, Heroku, Rundeck или Rightscale и многое другое. Более того, сейчас у каждого провайдера намного лучшее покрытие: скажем, провайдер для AWS охватывает большинство сервисов

[1] Список провайдеров для Terraform можно найти на странице https://registry.terraform.io/browse/providers.

этой платформы, а поддержка новых сервисов часто появляется даже раньше, чем у CloudFormation!

- *Реестр Terraform*. В 2017 году компания HashiCorp представила реестр Terraform (https://registry.terraform.io/) — пользовательский интерфейс, который облегчает просмотр и загрузку открытых универсальных модулей Terraform, разрабатываемых сообществом. В 2018 году была добавлена поддержка частных реестров, позволяющая организациям создавать свои закрытые реестры. В Terraform 0.11 появился полноценный синтаксис для загрузки модулей из реестра. Подробнее о реестре рассказывается в главе 8.
- *Улучшенная обработка ошибок*. В Terraform 0.9 обновилась обработка ошибок состояния: если при записи состояния в удаленное хранилище обнаруживается ошибка, это состояние сохраняется локально, в файле `errored.tfstate`. В Terraform 0.12 механизм был полностью переработан. Теперь ошибки перехватываются раньше, а сообщения о них стали более понятными и теперь содержат путь к файлу, номер строчки и фрагмент кода.
- *Много других мелких изменений*. Было сделано много менее значительных изменений, включая появление локальных переменных (см. раздел «Локальные переменные модуля» главы 4), новые «аварийные люки» для взаимодействия с внешним миром с помощью скриптов (например, раздел «За границами Terraform» главы 8), выполнение `plan` в рамках команды `apply`, исправление циклических проблем с `create_before_destroy`, значительное улучшение параметра `count`, которое позволяет ссылаться в нем на источники данных и ресурсы, десятки новых встроенных функций, обновленное наследование `provider` и многое другое.

Чего нет в этой книге

Книга не задумывалась как исчерпывающее руководство по Terraform. Она не охватывает все облачные провайдеры, все ресурсы, которые поддерживаются каждым из них, или каждую команду, доступную в этой системе. За этими подробностями я отсылаю вас к документации по адресу https://www.terraform.io/docs/index.html.

Документация содержит множество полезной информации, но, если вы только знакомитесь с Terraform, с концепцией «инфраструктура как код» или с системным администрированием, вы попросту не знаете, какие вопросы задавать. Поэтому данная книга сосредоточена на том, чего *нет* в документации: как выйти за рамки вводных примеров и начать использовать Terraform в реальных условиях. Моя цель — быстро подготовить вас к работе с этой системой. Для этого мы обсудим, зачем вообще может понадобиться Terraform, как внедрить этот