Оглавление

Предисловие к третьему изданию

Со времени публикации второго издания этой книги в 2017 году многое изменилось в мире Ansible и Python, включая выход нескольких новых версий. Немало изменений произошло и за пределами проекта: например, Red Hat, компания-основательница проекта Ansible, была куплена корпорацией IBM. Однако это никак не повлияло не проект Ansible: он все так же продолжает развиваться и привлекать новых пользователей. Развитие облачных и контейнерных технологий тоже значительно повлияло на общий ландшафт.

Мы внесли множество изменений в это издание. Наиболее существенное – добавление шести новых глав, охватывающих контейнеры, Molecule, коллекции Ansible, создание образов, поддержку облачной инфраструктуры и CI/CD. Мы также обновили и дополнили другие главы, уделив больше внимания передовым приемам разработки программного обеспечения и фреймворкам тестирования, помогающим проверить код и придать дополнительную уверенность в нем. Мы обновили все примеры кода для совместимости с последней версией Ansible, а также все сведения, что связаны с Python. Мы постарались отразить все важные изменения, произошедшие в период между 2017 и 2022 годами. Мы могли бы продолжать и дальше, но не будем этого делать, потому что вы сами, погрузившись в книгу, сможете увидеть, насколько далеко продвинулся Ansible.

Обозначения и соглашения, принятые в этой книге

В книге действуют следующие типографские соглашения:

Курсив

Используется для обозначения новых терминов, имен файлов и их расширений.

`Моноширинный шрифт`

Используется для оформления листингов программ, а также в обычном тексте для обозначения элементов программы, таких как имена переменных или функций, баз данных, типов данных, переменных окружения, инструкций и ключевых слов.

`Моноширинный полужирный шрифт`

Используется для выделения команд и другого текста, который должен быть набран самим пользователем.

`Моноширинный курсив`

Используется для выделения текста, который нужно заменить данными пользователя или значениями, определяемыми контекстом.

Так обозначаются примечания общего характера.

Так обозначаются предупреждения и предостережения.

Скачивание исходного кода примеров

Скачать файлы с дополнительной информацией для книг издательства «ДМК Пресс» можно на сайте **www.dmkpress.com** или **www.дмк.рф** на странице с описанием соответствующей книги.

Мы высоко ценим, хотя и не требуем, ссылки на наши издания. В ссылке обычно указываются имя автора, название книги, издательство и ISBN, например: «*Мейер Б., Хохштейн Л., Мозер Р.* Запускаем Ansible. М.: O'Reilly; ДМК Пресс, 2023. Copyright © 2023 O'Reilly Media, Inc., 978-1-098-10915-8 (англ.), 978-5-97060-513-4 (рус.)».

Если вы полагаете, что планируемое использование кода выходит за рамки изложенной выше лицензии, пожалуйста, обратитесь к нам по адресу **dmkpress@gmail.com**.

Список опечаток

Хотя мы приняли все возможные меры, для того чтобы удостовериться в качестве наших текстов, ошибки все равно случаются. Если вы найдете ошибку в одной из наших книг – возможно, ошибку в тексте или в коде, – мы будем очень благодарны, если вы сообщите нам о ней. Сделав это, вы избавите других читателей от расстройств и поможете нам улучшить последующие версии этой книги.

Если вы найдете какие-либо ошибки в коде, пожалуйста, сообщите о них главному редактору по адресу **dmkpress@gmail.com**, и мы исправим это в следующих тиражах.

Глава 1

Введение

Сейчас интересное время для работы в ИТ-индустрии. Мы не поставляем нашим клиентам программное обеспечение, установив его на одну-единственную машину и совершая дежурные звонки раз в день. Вместо этого мы медленно превращаемся в облачных инженеров.

Сейчас мы развертываем программные приложения, связывая воедино службы, которые работают в распределенной компьютерной сети и взаимодействуют по разным сетевым протоколам. Типичное приложение может включать веб-серверы, серверы приложений, систему кеширования данных в оперативной памяти, очереди задач, очереди сообщений, базы данных SQL, NoSQL-хранилища и балансировщики нагрузки.

Мы также должны убедиться в наличии достаточного количества ресурсов, и в случае ошибок в системе (а они будут случаться) мы элегантно выйдем из ситуации. Также имеются второстепенные службы, которые нужно разворачивать и поддерживать, такие как служба журналирования, мониторинга и анализа. Имеются и внешние службы, с которыми нужно устанавливать взаимодействие, например с интерфейсами «инфраструктура как сервис» (Infrastructure-as-a-Service, IaaS) для управления экземплярами виртуальных машин[1].

Мы можем связать эти службы вручную: запустить нужные серверы, зайти на каждый из них, установить пакеты приложений, отредактировать конфигурационные файлы и т. д. Но это серьезный труд. Такой процесс требует много времени, способствует появлению множества ошибок и просто утомляет, особенно в третий или четвертый раз. А работа вручную над более сложными задачами, как, например, установка облака OpenStack для вашего приложения, – так и просто сумасшествие. Есть способ лучше.

Если вы читаете эту книгу, значит, уже загорелись идеей управления конфигурациями и теперь рассматриваете Ansible как средство управ-

[1] Рекомендую превосходные книгу Томаса А. Лимонцелли (Thomas A. Limoncelli), Страта Р. Чалупа (Strata R. Chalup) и Кристины Дж. Хоган (Christina J. Hogan) «The Practice of Cloud System Administration», тома 1 и 2 (Addison-Wesley), и книгу Мартина Клеппмана (Martin Kleppman) «Designing Data-Intensive Applications» (O'Reilly).

ления. Кем бы вы ни были, разработчиком, развертывающим свой код в промышленном окружении, или системным администратором, ищущим лучшие средства автоматизации, я думаю, вы найдете в лице Ansible превосходное решение ваших проблем.

Примечание о версиях

Все примеры кода в этой книге были протестированы в версии Ansible 2.9.0, которая на момент написания книги являлась самой свежей. Ansible Tower включает версию 2.9.27 в последнем выпуске. Версия Ansible 2.8 завершила свой жизненный путь выпуском 2.8.20 в апреле 2021 года.

Многие годы сообщество Ansible активно разрабатывало новые роли и модули, в результате чего на свет появились тысячи модулей и более 20 000 ролей. Сложности, неизбежно возникающие при управлении такими масштабными проектами, привели создателей к необходимости реорганизовать Ansible и разделить его на три части:

- *компоненты ядра*, созданные командой Ansible;
- *сертифицированные* разработки, созданные бизнес-партнерами Red Hat;
- разработки *сообщества*, созданные тысячами энтузиастов по всему миру.

Ansible 2.9 имеет массу встроенных возможностей, но последующие версии будут более модульными. Эта новая организация проекта упрощает управление ими.

Примеры, представленные в этой книге, должны работать в разных версиях Ansible, но вообще смена версии предполагает тестирование, о чем мы поговорим в главе 14.

Откуда взялось название «Ansible»?

Название заимствовано из области научной фантастики. *Ansible* – это устройство связи, способное передавать информацию быстрее скорости света. Писатель Урсула Ле Гуин впервые представила эту идею в своем романе «Планета Роканнона», а остальные писатели-фантасты подхватили ее. Если быть более точным, Майкл ДеХаан, сооснователь проекта, позаимствовал название Ansible из книги Орсона Скотта Карда «Игра Эндера». В этой книге ansible использовался для одновременного контроля большого числа кораблей, удаленных на огромные расстояния. Подумайте об этом как о метафоре контроля удаленных серверов.

Ansible: область применения

Ansible часто описывают как *инструмент управления конфигурациями*, и обычно он упоминается в том же контексте, что и Chef, Puppet и Salt. Когда мы говорим об *управлении конфигурациями*, то часто подразумеваем описание состояния серверов в некотором виде, а затем применение специальных средств для приведения серверов в это состояние: установку необходимых пакетов приложений, копирование конфигурационных файлов с определенными разрешениями в файловой системе, запуск необходимых служб и т. д. Подобно другим средствам управления, Ansible предоставляет *предметно-ориентированный язык (Domain Specific Language, DSL)*, который используется для описания состояний серверов.

Эти инструменты также можно использовать для развертывания программного обеспечения. Под развертыванием мы часто подразумеваем процесс получения двоичного кода из исходного (если необходимо), копирования необходимых файлов на сервер(ы), добавление конфигурационных свойств и переменных окружения и запуск служб в определенном порядке. Capistrano и Fabric – два примера инструментов с открытым кодом для развертывания приложений. Ansible тоже является превосходным инструментом как для развертывания, так и для управления конфигурациями программного обеспечения. Использование единой системы управления конфигурациями и развертыванием значительно упрощает жизнь системным администраторам.

Некоторые специалисты отмечают необходимость согласования развертывания, когда в процесс вовлечено несколько удаленных серверов и операции должны осуществляться в определенном порядке. Например, базу данных нужно установить до установки веб-серверов или выводить веб-серверы из-под управления балансировщика нагрузки только по одному, чтобы система не прекращала работу во время обновления. Система Ansible хороша и в этом, поскольку изначально создавалась для проведения манипуляций сразу на нескольких серверах. Ansible имеет удивительно простую модель управления порядком действий.

Наконец, вы услышите, как люди говорят о подготовке и наполнении (provisioning) новых серверов. В контексте облачных услуг, таких как Amazon EC2, под *подготовкой и наполнением* подразумевается развертывание новых экземпляров виртуальной машины или облачных служб «программное обеспечение как услуга» (Software as a Service, SaaS). Ansible охватывает и эту область, предоставляя несколько модулей поддержки облаков, включая EC2, Azure[1], Digital Ocean, Google Compute Engine, Linode и Rackspace[2], а также любые облака, поддерживающие OpenStack API.

[1] Да, Azure поддерживает серверы на Linux.

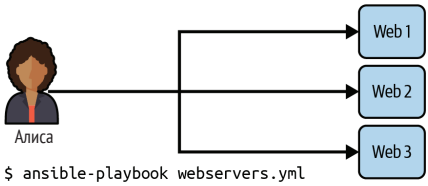
[2] Например, смотрите презентацию Джесса Китинга (Jesse Keating) «Using Ansible at Scale to Manage a Public Cloud» (https://oreil.ly/djLsk), ранее работавшего в Rackspace.

Несколько сбивает с толку использование термина *провайдер* (provisioner) в документации к утилите Vagrant, которую мы обсудим далее в этой главе, в отношении системы управления конфигурациями. Так, Vagrant называет Ansible своего рода провайдером там, где, как мне кажется, провайдером является сам Vagrant, поскольку именно он отвечает за запуск виртуальных машин.

Как работает Ansible

На рис. 1.1 показан простой пример использования Ansible. Пользователь, которого мы будем звать Алиса, использует Ansible для настройки трех веб-серверов NGINX, действующих под управлением Ubuntu. Она написала для Ansible сценарий *webservers.yml*. В терминологии Ansible сценарии называются *playbook*. Сценарий описывает, какие *хосты* (в Ansible они называются удаленными серверами) подлежат настройке и упорядоченный список *задач*, которые должны быть выполнены на этих хостах. В этом примере хосты носят имена web1, web2 и web3, и для настройки каждого из них требуется выполнить следующие задачи:

- установить Nginx;
- сгенерировать конфигурационные файлы для Nginx;
- скопировать сертификат безопасности;
- запустить Nginx.



Playbook: webservers.yml

```
---
- name: Configure webservers
  hosts: webservers
  become: True
  tasks:
    - name: Install nginx
      package: name=nginx
    - name: Install config file
      template:
        src: nginx.config.j2
        dest: /etc/nginx/nginx.conf
      notify: Restart nginx
  handlers:
    - name: Restart nginx
      service:name=nginx state=restarted
```

Рис. 1.1. Ansible выполняет сценарий настройки трех веб-серверов

В следующей главе мы обсудим, что в действительности входит в этот сценарий. Алиса запускает сценарий командой `ansible-playbook`. В примере сценарий называется *webservers.yml* и запускается вводом команды в терминале:

```
$ ansible-playbook webservers.yml
```

Ansible устанавливает параллельные SSH-соединения с хостами web1, web2 и web3 и выполняет первую задачу из списка на всех хостах одновременно. В этом примере первая задача – установка пакета NGINX, которая в сценарии выглядит так:

```
- name: Install nginx
  package:
    name: nginx
```

Выполняя ее, Ansible проделает следующие действия.

1. Сгенерирует сценарий на языке Python, который установит пакет NGINX.
2. Скопирует его на хосты web1, web2 и web3.
3. Запустит на хостах web1, web2 и web3.
4. Дождется, пока сценарий завершится на всех хостах.

Затем Ansible перейдет к следующей задаче в списке и повторит эти же четыре шага.

Важно отметить, что:

1. каждая задача выполняется на всех хостах одновременно;
2. Ansible ожидает завершения задачи на всех хостах, прежде чем приступить к выполнению следующей;
3. задачи выполняются в установленном вами порядке.

Какие преимущества дает Ansible?

Существует несколько систем управления конфигурациями с открытым исходным кодом, так почему мы выбираем Ansible? Ниже перечисляется 21 причина, подталкивающая нас к этому выбору. Но основными являются простота, широта возможностей и защищенность.

Простота

Разработчики стремились максимально упростить процесс установки и освоение Ansible.

Простота синтаксиса

Сценарии Ansible определяются в файлах формата YAML с использованием синтаксиса шаблонов Jinja2. Напомним, что в терминоло-

гии Ansible сценарии управления конфигурацией называются *playbook* (сценарий, пьеса). Фактически синтаксис сценариев Ansible основан на YAML, языке описания данных, который создавался специально, чтобы легко восприниматься человеком. В некотором роде YAML для JSON – то же, что Markdown для HTML.

Простота аудита

Сценарии Ansible легко поддаются исследованию – например, можно легко получить список всех действий и вовлеченных хостов. Для выполнения пробных прогонов мы часто используем команду `ansible-playbook --check`. Встроенная поддержка журналирования позволяет увидеть, кто, что и где делал. Механизм журналирования реализован как подключаемый модуль, а созданные им журналы легко получить с помощью сборщиков журналов.

Практически ничего не нужно устанавливать на удаленных хостах

Для управления серверами с помощью Ansible на серверах Linux должна быть установлена поддержка SSH и Python, а на серверах Windows должен быть включен WinRM. В Windows Ansible использует PowerShell вместо Python, что избавляет от необходимости предварительно устанавливать на хосте какого-либо агента или любое другое программное обеспечение.

На *управляющей машине* (той, что используется для управления удаленными машинами) должен быть установлен Python версии 3.8 или выше. В зависимости от ресурсов, которыми требуется управлять с помощью Ansible, может потребоваться установить дополнительные сторонние библиотеки. Загляните в документацию, чтобы узнать, имеются ли у модуля особые требования.

Возможность масштабирования вниз

Да, Ansible можно использовать для управления сотнями и даже тысячами узлов. Но что нас особенно зацепило, так это его масштабируемость вниз. Ansible можно запустить на очень скромном оборудовании, таком как Raspberry Pi или старом ПК, и даже использовать для управления единственным узлом – нужно лишь написать один сценарий. Ansible подтверждает принцип Алана Кея: «Простое должно оставаться простым, а сложное – возможным».

Простота распространения

Мы не думаем, что вам понадобится повторно использовать сценарии Ansible в разных контекстах. В главе 7 мы обсудим роли, предлагающие возможность организации сценариев, и Ansible Galaxy, онлайн-репози-

торий для хранения этих ролей.

Основной единицей повторного использования в сообществе Ansible в настоящее время является *коллекция*. Вы можете упаковать свои модули, плагины, библиотеки, роли и даже сборники игр в коллекцию и поделиться ею с другими через Ansible Galaxy. Также можно организовать распространение внутри организации с помощью инструмента Automation Hub, входящего в состав Ansible Tower. Роли могут использоваться совместно как отдельные репозитории.

На практике, однако, каждая организация настраивает свои серверы немного не так, как другие, поэтому лучше писать свои сборники сценариев для своей организации, а не пытаться повторно использовать те, что находятся в общем доступе. Мы считаем, что основная ценность изучения чужих схем заключается в возможности увидеть, как все работает, если только вы не работаете с конкретным продуктом, производитель которого является сертифицированным партнером или членом сообщества Ansible.

Простота абстракций

Ansible работает с простыми *абстракциями* системных ресурсов, таких как файлы, каталоги, пользователи, группы, службы, пакеты и веб-сервисы.

Для сравнения давайте посмотрим, как настроить каталог в командной оболочке. Для этого используются три команды:

```
mkdir -p /etc/skel/.ssh
chown root:root /etc/skel/.ssh
chmod go-wrx /etc/skel/.ssh
```

Ansible, в свою очередь, предлагает абстракцию – модуль `file`, с помощью которого определяются параметры желаемого состояния. Следующее единственное действие дает тот же эффект, что и три команды выше:

```
- name: Ensure .ssh directory in user skeleton
  file:
    path: /etc/skel/.ssh
    mode: '0700'
    owner: root
    group: root
    state: directory
```

Этот слой абстракции позволяет использовать одни и те же сценарии для управления конфигурациями серверов с Linux. Например, вместо использования конкретного диспетчера пакетов, такого как `dnf`, `yum` или `apt`, Ansible предлагает абстракцию «пакет» (просто имейте в виду, что

имена пакетов могут отличаться). Но при желании можно также использовать системные абстракции.

Если вы действительно этого хотите, то можете написать свои сценарии Ansible для выполнения различных действий в разных операционных системах на удаленных серверах. Но Бас, один из авторов этой книги, старается избегать этого по возможности, предпочитая писать сценарии для реально используемых систем.

Выполнение задач сверху вниз

В книгах по управлению конфигурациями часто упоминается идея *конвергенции* (сходимости), или *последовательного приведения к конечному состоянию*, которая нередко ассоциируется с именем Марка Бургесса (Mark Burgess) и его системой управления конфигурациями CFEngine. Если система управления конфигурациями конвергентна, то она может многократно выполнять управляющие воздействия, с каждым разом приводя сервер все ближе к желаемому состоянию.

Идея конвергенции неприменима к Ansible из-за отсутствия понятия многоэтапных воздействий на конфигурацию серверов. Модули Ansible устроены так, что единственный запуск сценария Ansible сразу приводит каждый сервер в желаемое состояние.

Широта возможностей

Ansible помогает значительно повысить производительность в нескольких областях управления системами. Абстракции высокого уровня, предоставляемые Ansible (например, роли), дают возможность устанавливать и настраивать программное обеспечение быстрее и потенциально безопаснее.

Встроенные модули

Ansible можно использовать для выполнения произвольных команд оболочки на удаленных серверах, но по-настоящему сильной его стороной является набор встроенных модулей. Модули необходимы для выполнения таких задач, как установка пакетов приложений, перезапуск службы или копирование конфигурационных файлов.

Как мы увидим позже, модули Ansible несут *декларативную* функцию и используются для описания требуемого состояния серверов. Например, вы могли бы вызвать модуль `user`, чтобы убедиться в существовании учетной записи `deploy` в группе `web`:

```
- name: Ensure deploy user exists
  user:
    name: deploy
    group: web
```

Использование технологии принудительной настройки

Некоторые системы управления конфигурациями, использующие агентов, такие как Chef и Puppet, по умолчанию основаны на технологии *добровольной настройки*. Агенты, установленные на серверах, периодически подключаются к центральной службе и читают информацию о конфигурации. Управление изменениями конфигурации серверов в этом случае выглядит так:

1. вы: вносите изменения в сценарий управления конфигурациями,
2. вы: передаете изменения центральной службе,
3. агент на сервере: периодически включается по таймеру,
4. агент на сервере: подключается к центральной службе,
5. агент на сервере: читает новые сценарии управления конфигурациями,
6. агент на сервере: запускает полученные сценарии локально, обновляя состояние сервера.

Ansible, напротив, по умолчанию использует технологию *принудительной настройки*. Внесение изменений выглядит так:

1. вы: вносите изменения в сценарий,
2. вы: запускаете новый сценарий,
3. Ansible: подключается к серверам и запускает модули, обновляя состояние серверов.

Как только вы запустите команду `ansible-playbook`, Ansible подключится к удаленным серверам и выполнит всю работу; это снижает риск выхода из строя случайных серверов, когда запланированные на них задачи не могут успешно изменить их состояние. Принудительная настройка дает важное преимущество – вы контролируете время обновления серверов. Вам не приходится ждать. Каждое действие в сценарии может быть нацелено на один или группу серверов. Вы можете выполнять больше операций автоматически, не выполняя вход на серверы вручную.

Многоуровневая оркестрация

Технология принудительной настройки позволяет также реализовать с помощью Ansible *многоуровневую оркестрацию* – управление отдельными группами компьютеров для выполнения различных операций, таких как обновление ПО. Вы можете организовать управление системами мониторинга, балансировщиками нагрузки, базами данных и веб-серверами с помощью конкретных инструкций и обеспечить их согласованную работу. Это очень сложно сделать с системой, основанной на технологии добровольной настройки.

Отсутствие ведущего узла

Сторонники добровольной настройки утверждают, что их подход лучше масштабируется на большое число серверов и удобнее, когда новые серверы могут появиться в любой момент. Однако централизованная система управления конфигурацией может испытывать значительную нагрузку, когда тысячи агентов одновременно попытаются извлечь свою конфигурацию, особенно если им требуется выполнить несколько циклов для конвергенции. Для сравнения: Ansible официально поддерживает особый режим, называемый `ansible-pull`, в котором сценарии извлекаются из репозитория, такого как GitHub. Ansible не нуждается в ведущем узле, но при желании вы можете использовать централизованную систему для запуска сценариев.

Поддержка плагинов

Значительная часть функциональности Ansible реализована в виде подключаемых модулей – плагинов, из которых наиболее часто используются плагины Lookup и Filter. Плагины расширяют базовые возможности Ansible логикой и функциями, доступными для всех модулей. Модули вводят в язык Ansible новые «глаголы». Вы тоже можете писать свои плагины (глава 10) и модули (глава 12) на Python .

Ansible можно интегрировать с другими продуктами. Примерами успешной интеграции могут служить Kubernetes и Ansible Tower. Ansible Runner – это «инструмент и библиотека для Python, помогающая организовать взаимодействие с Ansible напрямую или в составе другой системы, например через интерфейс образа контейнера, как автономный инструмент или как модуль на Python, который можно импортировать»[1].

С помощью библиотеки *ansible-runner* можно запустить сценарий Ansible из программы на Python:

```
#!/usr/bin/env python3
import ansible_runner

r = ansible_runner.run(private_data_dir='./playbooks', playbook='playbook.yml')

print("{}: {}".format(r.status, r.rc))
print("Final status:")
print(r.stats)
```

Поддержка решения широкого круга задач

Модули Ansible предназначены для решения широкого круга задач системного администрирования. В списке ниже перечислены катего-

[1] Цитата из документации к Ansible Runner (https://oreil.ly/sZwPY).

рии доступных модулей. Эти ссылки ведут в список модулей (**https://oreil.ly/0Xel7**) в документации:

- Cloud (**https://oreil.ly/0xeNu**);
- Files (**https://oreil.ly/3cq87**);
- Monitoring (**https://oreil.ly/z6dde**);
- Source Control (**https://oreil.ly/WEMHZ**);
- Clustering (**https://oreil.ly/b31cn**);
- Identity (**https://oreil.ly/39yJA**);
- Net Tools (**https://oreil.ly/Pb137**);
- Storage (**https://oreil.ly/IZBGX**);
- Commands (**https://oreil.ly/wyyJZ**);
- Infrastructure (**https://oreil.ly/XhW90**);
- Network (**https://oreil.ly/UFHZo**);
- System (**https://oreil.ly/mn569**);
- Crypto (**https://oreil.ly/puZGg**);
- Inventory (**https://oreil.ly/zBvdF**);
- Notification (**https://oreil.ly/ulrdH**);
- Utilities (**https://oreil.ly/veSG4**);
- Database (**https://oreil.ly/iEv9l**);
- Messaging (**https://oreil.ly/aTOvP**);
- Packaging (**https://oreil.ly/71GLO**);
- Windows (**https://oreil.ly/c8NwK**).

Настоящая масштабируемость

Крупные предприятия успешно используют Ansible для настройки десятков тысяч узлов и отлично поддерживают окружения, в которых серверы появляются и исчезают динамически. Организации с сотнями групп разработчиков программного обеспечения обычно используют AWX или комбинацию Ansible Tower и Automation Hub для аудита и защиты с контролем доступа на основе ролей.

Вас волнует масштабируемость SSH? Ansible использует мультиплексирование SSH для оптимизации производительности и реальные примеры управления тысячами узлов с помощью Ansible (глава 12).

Защищенность

Автоматизация с помощью Ansible помогает повысить защищенность системы до базовых уровней безопасности и стандартов соответствия.

Самодокументирующийся код

Авторам книги нравится думать о сценариях Ansible как о выполняемой документации. Они сродни файлам README, которые описывают действия, необходимые для развертывания программного обеспечения, но, в отличие от них, сценарии всегда содержат актуальные инструкции, поскольку сами являются выполняемым кодом. Эксперты могут создавать сценарии, отражающие передовой опыт, а новички – использовать их как учебники и пребывать в уверенности, что получат хороший результат.

Воспроизводимость

Если всю свою систему вы настроите с помощью Ansible, то она пройдет то, что Стив Трауготт (Steve Traugott) называет «тестированием десятым этажом» (**https://oreil.ly/AMf1S**): «Могу ли я взять случайную машину, для которой никогда не выполнялось резервного копирования, выкинуть ее из окна десятого этажа и при этом не потерять работу системного администратора?»

Эквивалентность создаваемых окружений

Ansible поддерживает определенный способ организации контента, помогающий определить конфигурацию на надлежащем уровне. Вы с легкостью сможете определить настройки для различных окружений: разработки, тестирования, обкатки и промышленной эксплуатации. Окружение обкатки обычно делается максимально похожим на промышленное окружение, чтобы разработчики могли выявить любые проблемы до того, как изменения попадут в промышленное окружение.

Шифрование переменных

При необходимости хранить конфиденциальные данные, такие как пароли или токены, можно использовать эффективный инструмент `ansible-vault`. Мы используем его для шифрования переменных в Git. Более подробно этот вопрос обсуждается в главе 8.

Защищенный транспорт

Ansible просто использует Secure Shell (SSH) для Linux и WinRM для Windows. Обычно мы защищаем и укрепляем эти широко используемые протоколы управления системами с помощью защищенных настроек конфигурации и брандмауэра.

Если вы предпочитаете модель, основанную на приемах добровольной настройки, то для вас Ansible официально поддерживает особый режим, называемый `ansible-pull`. В этой книге не раскрываются особенности этого режима, но вы можете узнать больше об этом из официальной документации (**https://docs.ansible.com/**).

Идемпотентность

Модули также являются идемпотентными[1]. Идемпотентность – замечательное свойство и означает, что сценарий Ansible можно применить к одному и тому же серверу много раз без всякого ущерба для конфигурации последнего. Давайте рассмотрим пример, когда нам нужно создать пользователя `deploy`:

```
- name: Ensure deploy user exists
  user:
    name: deploy
    group: web
```

Если пользователя `deploy` не существует, то Ansible создаст его. Если он существует, то Ansible просто перейдет к следующему шагу. То есть сценарии Ansible можно запускать на сервере много раз. Это важное отличие от сценариев командной оболочки, потому что повторный запуск таких сценариев может привести к незапланированным – и хорошо, если безобидным – последствиям[2].

Отсутствие демонов

В Ansible нет агента, прослушивающего некоторый порт. Поэтому у злоумышленников нет цели для атаки на Ansible. (Однако существуют другие цели для атаки – элементы цепочки доставки программного обеспечения, такого как библиотеки Python и другие импортируемые компоненты.)

Связь между Ansible и Ansible, Inc.

Название *Ansible* относится как к программному обеспечению, так и к компании, управляющей проектом. Майкл ДеХаан, создатель программного обеспечения Ansible, является бывшим техническим директором компании Ansible. Во избежание путаницы хочу уточнить, что для обозначения продукта я использую Ansible, а компании – Ansible, Inc.

Ansible, Inc. проводит обучение и предоставляет консультационные услуги по Ansible, а также собственной веб-системе управления Ansible Tower, о которой рассказывается в главе 19. В октябре 2015 года Red Hat купила Ansible Inc., а в 2019 году IBM купила Red Hat.

[1] Идемпоте́нтность – свойство объекта или операции при повторном применении операции к объекту давать тот же результат, что и при одинарном. – *Прим. перев.*

[2] Если вам интересно, что думает автор Ansible об идемпотентности и конвергенции, прочтите публикацию Майкла ДеХаана «Idempotence, convergence, and other silly fancy words we use too often» («Идемпотентность, конвергенция и другие причудливые слова, которые мы используем слишком часто») на странице группы Ansible Project (https://oreil.ly/pNSNr).

Не слишком ли проста система Ansible?

В период работы над книгой редактор сказал Лорин, что «некоторые специалисты, использующие систему управления конфигурациями XYZ, называют Ansible «циклом for по сценариям, запускаемым через SSH». Планируя переход с другой системы управления конфигурациями на Ansible, действительно могут возникнуть сомнения в его эффективности.

Однако, как скоро будет показано, Ansible имеет гораздо более широкие возможности, чем сценарии командной оболочки. Как уже упоминалось, модули Ansible гарантируют идемпотентность, Ansible имеет превосходную поддержку шаблонов и переменных с разными областями видимости. Любой, кто считает, что суть Ansible заключается в работе со сценариями командной оболочки, никогда не занимался поддержкой нетривиальных программ на языке оболочки. Если есть выбор, я предпочту Ansible сценариям командной оболочки.

Что я должен знать?

Для эффективной работы с Ansible необходимо знать основы администрирования операционной системы Unix/Linux. Ansible позволяет автоматизировать процессы, но не является волшебным инструментом, способным выполнять операции, которые вы не знаете, как выполнить.

Читатели данной книги должны быть знакомы по крайней мере с одним из дистрибутивов Linux (Ubuntu, RHEL/CentOS, SUSE и пр.) и понимать, как:

- подключиться к удаленной машине через SSH;
- работать в командной строке Bash (каналы и перенаправление);
- устанавливать пакеты приложений;
- использовать команду *sudo*;
- проверять и устанавливать разрешения для файлов;
- запускать и останавливать службы;
- устанавливать переменные окружения;
- писать сценарии (на любом языке).

Если все это вам известно, то можете смело приступать к работе с Ansible.

Я не предполагаю, что вы знаете какой-то определенный язык программирования. Например, вам не нужно знать Python, если вы не собираетесь самостоятельно писать модули.

О чем не рассказывается в этой книге

Эта книга не является исчерпывающим руководством по работе с Ansible. Она позволяет подготовиться к использованию Ansible в кратчайшие сроки и дает описание некоторых задач, которые недостаточно полно описываются в официальной документации.

Книга не описывает использования официальных модулей Ansible. Их более 3500, и они достаточно хорошо представлены в официальной документации. Для просмотра справочной документации и списка модулей, упоминавшихся выше, можно использовать инструмент командной строки `ansible-doc`.

Глава 8 охватывает только основные возможности механизма шаблонов Jinja2, главным образом потому, что авторы используют только самые основные функции Jinja2 при работе с Ansible. Для более глубокого знакомства Jinja2 обращайтесь к официальной документации Jinja2 (**https://oreil.ly/LAXa7**).

Книга не дает детального описания некоторых возможностей Ansible, используемых в основном для поддержки ранних версий Linux.

Наконец, некоторые особенности Ansible мы не будем рассматривать, просто чтобы не увеличивать и без того немалый объем книги. Для знакомства с этими особенностями обращайтесь к официальной документации (**https://docs.ansible.com/**).

Поехали!

В этой вводной главе мы в общих чертах рассмотрели основные понятия Ansible, в том числе особенности взаимодействий с удаленными серверами и отличия от других инструментов управления конфигурациями. В следующих главах обсуждается практическое использование Ansible.