Содержание

Предисловие 16

Предисловие ко второму изданию 18

Предисловие к первому изданию 20

Глава 1. Введение 23
Примечание о версиях 24
Ansible: область применения 24
Как работает Ansible 25
Какие преимущества дает Ansible? 26
Простота синтаксиса 27
Отсутствие необходимости установки на удаленных хостах 27
Основан на технологии принудительной настройки 27
Управление небольшим числом серверов 28
Встроенные модули 28
Очень тонкий слой абстракции 29
Не слишком ли проста система Ansible? 30
Что я должен знать? 31
О чем не рассказывается в этой книге 31
Установка Ansible 32
Подготовка сервера для экспериментов 33
Использование Vagrant для подготовки сервера 33
Передача информации о сервере в Ansible 36
Упрощение задачи с помощью файла ansible.cfg 37
Что дальше 40

Глава 2. Сценарии: начало 41
Подготовка 41
Очень простой сценарий 42
Файл конфигурации Nginx 44
Создание начальной страницы 44
Создание группы веб-серверов 45
Запуск сценария 45
Сценарии пишутся на YAML 47
Начало файла 47
Комментарии 47

Строки 47
Булевы выражения 47
Списки 48
Словари 48
Объединение строк 49
Структура сценария 49
Операции 50
Задачи 52
Модули 53
Резюме 54
Есть изменения? Отслеживание состояния хоста 54
Становимся знатоками: поддержка TLS 55
Создание сертификата TLS 56
Переменные 56
Создание шаблона с конфигурацией Nginx 58
Обработчики 59
Запуск сценария 60

Глава 3. Реестр: описание серверов 63
Файл реестра 63
Вводная часть: несколько машин Vagrant 64
Поведенческие параметры хостов в реестре 67
ansible_connection 67
ansible_shell_type 67
ansible_python_interpreter 68
ansible_*_interpreter 68
Переопределение поведенческих параметров по умолчанию 68
Группы, группы и еще раз группы 68
Пример: развертывание приложения Django 70
Псевдонимы и порты 72
Группировка групп 72
Имена хостов с номерами (домашние питомцы и стадо) 72
Переменные хостов и групп: внутренняя сторона реестра 73
Переменные хостов и групп: создание собственных файлов 75
Динамический реестр 76
Интерфейс сценария динамического реестра 77
Написание сценария динамического реестра 78
Предопределенные сценарии реестра 81
Деление реестра на несколько файлов 82
Добавление элементов во время выполнения с помощью add_host и group_by 82
add_host 82
group_by 83

Глава 4. Переменные и факты 85
Определение переменных в сценариях 85
Вывод значений переменных 86
Регистрация переменных 86
Факты 89
Просмотр всех фактов, доступных для сервера 90
Вывод подмножества фактов 91
Любой модуль может возвращать факты 92
Локальные факты 93
Использование модуля set_fact для задания новой переменной 94
Встроенные переменные 94
hostvars 95
inventory_hostname 95
groups 96
Установка переменных из командной строки 96
Приоритет 97

Глава 5. Введение в Mezzanine: тестовое приложение 99
Почему сложно развертывать приложения в промышленном окружении 99
База данных PostgreSQL 103
Сервер приложений Gunicorn 103
Веб-сервер Nginx 104
Диспетчер процессов Supervisor 105

Глава 6. Развертывание Mezzanine с помощью Ansible 106
Вывод списка задач в сценарии 106
Организация устанавливаемых файлов 107
Переменные и скрытые переменные 108
Использование цикла (with_items) для установки большого количества пакетов 109
Добавление выражения become в задачу 111
Обновление кэша диспетчера пакетов apt 111
Извлечение проекта из репозитория Git 113
Установка Mezzanine и других пакетов в virtualenv 115
Короткое отступление: составные аргументы задач 117
Настройка базы данных 120
Создание файла local_settings.py из шаблона 121
Выполнение команд django-manage 124
Запуск своих сценариев на Python в контексте приложения 125
Настройка конфигурационных файлов служб 127
Активация конфигурации Nginx 130
Установка сертификатов TLS 130
Установка задания cron для Twitter 131

Сценарий целиком 132
Запуск сценария на машине Vagrant 136
Устранение проблем 136
Не получается извлечь файлы из репозитория Git 136
Недоступен хост с адресом 192.168.33.10.xip.io 137
Bad Request (400) 137
Установка Mezzanine на нескольких машинах 137

Глава 7. Роли: масштабирование сценария 138
Базовая структура роли 138
Примеры ролей: database и mezzanine 139
Использование ролей в сценариях 139
Предварительные и заключительные задачи 140
Роль database для развертывания базы данных 141
Роль mezzanine для развертывания Mezzanine 143
Создание файлов и директорий ролей с помощью ansible-galaxy 148
Зависимые роли 148
Ansible Galaxy 149
Веб-интерфейс 149
Интерфейс командной строки 150
Добавление собственной роли 151

Глава 8. Сложные сценарии 152
Команды changed_when и failed_when 152
Фильтры 155
Фильтр default 156
Фильтры для зарегистрированных переменных 156
Фильтры для путей к файлам 156
Создание собственного фильтра 157
Подстановки 158
file 159
pipe 160
env 160
password 160
template 161
csvfile 161
dnstxt 162
redis_kv 163
etcd 164
Написание собственного плагина 164
Сложные циклы 164
with_lines 165
with_fileglob 165

with_dict ... 166
Циклические конструкции как плагины подстановок ... 167
Управление циклами ... 167
Выбор имени переменной цикла ... 167
Управление выводом ... 168
Подключение ... 169
Динамическое подключение ... 170
Подключение ролей ... 171
Блоки ... 172
Обработка ошибок с помощью блоков ... 172
Шифрование конфиденциальных данных при помощи Vault ... 175

Глава 9. Управление хостами, задачами и обработчиками ... 178
Шаблоны для выбора хостов ... 178
Ограничение обслуживаемых хостов ... 179
Запуск задачи на управляющей машине ... 179
Запуск задачи на сторонней машине ... 180
Последовательное выполнение задачи на хостах по одному ... 180
Пакетная обработка хостов ... 182
Однократный запуск ... 183
Стратегии выполнения ... 183
linear ... 184
free ... 185
Улучшенные обработчики ... 186
Обработчики в pre_tasks и post_tasks ... 186
Принудительный запуск обработчиков ... 187
Выполнение обработчиков по событиям ... 189
Сбор фактов вручную ... 195
Получение IP-адреса хоста ... 195

Глава 10. Плагины обратного вызова ... 197
Плагины стандартного вывода ... 197
actionable ... 198
debug ... 198
dense ... 199
json ... 199
minimal ... 200
oneline ... 200
selective ... 200
skippy ... 200
Другие плагины ... 201
foreman ... 201
hipchat ... 202

jabber....202
junit....202
log_plays....203
logentries....203
logstash....203
mail....204
osx_say....204
profile_tasks....204
slack....205
timer....205

Глава 11. Ускорение работы Ansible....206
Мультиплексирование SSH и ControlPersist....206
Включение мультиплексирования SSH вручную....207
Параметры мультиплексирования SSH в Ansible....208
Конвейерный режим....209
Включение конвейерного режима....210
Настройка хостов для поддержки конвейерного режима....210
Кэширование фактов....211
Кэширование фактов в файлах JSON....213
Кэширование фактов в Redis....213
Кэширование фактов в Memcached....214
Параллелизм....214
Асинхронное выполнение задач с помощью Async....215

Глава 12. Собственные модули....217
Пример: проверка доступности удаленного сервера....217
Использование модуля script вместо написания своего модуля....217
Где хранить свои модули....218
Как Ansible вызывает модули....218
Генерация автономного сценария на Python с аргументами (только модули на Python)....219
Копирование модуля на хост....219
Создание файла с аргументами на хосте (для модулей не на языке Python)....219
Вызов модуля....219
Ожидаемый вывод....220
Ожидаемые выходные переменные....220
Реализация модулей на Python....221
Анализ аргументов....222
Доступ к параметрам....223
Импортирование вспомогательного класса AnsibleModule....223
Свойства аргументов....224

AnsibleModule: параметры метода инициализатора 226
Возврат признака успешного завершения или неудачи 229
Вызов внешних команд 229
Режим проверки (пробный прогон) 230
Документирование модуля 231
Отладка модуля 233
Создание модуля на Bash 234
Альтернативное местоположение интерпретатора Bash 235
Примеры модулей 236

Глава 13. Vagrant 237
Полезные параметры настройки Vagrant 237
Перенаправление портов и приватные IP-адреса 237
Перенаправление агента 239
Сценарий наполнения Ansible 239
Когда выполняется сценарий наполнения 239
Реестр, генерируемый системой Vagrant 240
Наполнение нескольких машин одновременно 241
Определение групп 242
Локальные сценарии наполнения 243

Глава 14. Amazon EC2 244
Терминология 246
Экземпляр 246
Образ машины Amazon 246
Теги 246
Учетные данные пользователя 247
Переменные окружения 247
Файлы конфигурации 248
Необходимое условие: библиотека Python Boto 248
Динамическая инвентаризация 249
Кэширование реестра 251
Другие параметры настройки 251
Автоматические группы 251
Определение динамических групп с помощью тегов 252
Присваивание тегов имеющимся ресурсам 252
Создание более точных названий групп 253
EC2 Virtual Private Cloud (VPC) и EC2 Classic 254
Конфигурирование ansible.cfg для использования с ec2 255
Запуск новых экземпляров 255
Пары ключей EC2 257
Создание нового ключа 257
Выгрузка существующего ключа 258

Группы безопасности ... 258
Разрешенные IP-адреса ... 260
Порты групп безопасности ... 260
Получение новейшего AMI ... 261
Добавление нового экземпляра в группу ... 262
Ожидание запуска сервера ... 264
Создание экземпляров идемпотентным способом ... 265
Подведение итогов ... 265
Создание виртуального приватного облака ... 267
Динамическая инвентаризация и VPC ... 272
Создание AMI ... 272
Использование модуля ec2_ami ... 272
Использование Packer ... 273
Другие модули ... 277

Глава 15. Docker ... 278
Объединение Docker и Ansible ... 279
Жизненный цикл приложения Docker ... 280
Пример применения: Ghost ... 281
Подключение к демону Docker ... 281
Запуск контейнера на локальной машине ... 281
Создание образа из Dockerfile ... 282
Управление несколькими контейнерами на локальной машине ... 284
Отправка образа в реестр Docker ... 285
Запрос информации о локальном образе ... 287
Развертывание приложения в контейнере Docker ... 288
Postgres ... 288
Веб-сервер ... 289
Веб-сервер: Ghost ... 290
Веб-сервер: Nginx ... 291
Удаление контейнеров ... 291
Прямое подключение к контейнерам ... 292
Контейнеры Ansible ... 293
Контейнер Conductor ... 293
Создание образов Docker ... 294
Настройка container.yml ... 295
Запуск на локальной машине ... 297
Публикация образов в реестрах ... 298
Развертывание контейнеров в промышленном окружении ... 300

Глава 16. Отладка сценариев Ansible ... 301
Информативные сообщения об ошибках ... 301
Отладка ошибок с SSH-подключением ... 302

Модуль debug 303
Интерактивный отладчик сценариев 304
Модуль assert 305
Проверка сценария перед запуском 307
Проверка синтаксиса 307
Список хостов 307
Список задач 308
Проверка режима 308
Вывод изменений в файлах 308
Выбор задач для запуска 309
Пошаговое выполнение 309
Выполнение с указанной задачи 309
Теги 310

Глава 17. Управление хостами Windows 311
Подключение к Windows 311
PowerShell 312
Модули поддержки Windows 314
Наш первый сценарий 315
Обновление Windows 316
Добавление локальных пользователей 317
Итоги 320

Глава 18. Ansible для сетевых устройств 321
Статус сетевых модулей 322
Список поддерживаемых производителей сетевого оборудования 322
Подготовка сетевого устройства 322
Настройка аутентификации через SSH 323
Как работают модули 325
Наш первый сценарий 326
Реестр и переменные для сетевых модулей 327
Локальное подключение 328
Подключение к хосту 329
Переменные для аутентификации 329
Сохранение конфигурации 330
Использование конфигураций из файлов 331
Шаблоны, шаблоны, шаблоны 334
Сбор фактов 336
Итоги 338

Глава 19. Ansible Tower: Ansible для предприятий 339
Модели подписки 340
Пробная версия Ansible Tower 340

Какие задачи решает Ansible Tower ... 341
Управление доступом ... 341
Проекты ... 342
Управление инвентаризацией ... 342
Запуск заданий из шаблонов ... 344
RESTful API ... 346
Интерфейс командной строки Ansible Tower ... 347
Установка ... 347
Создание пользователя ... 348
Запуск задания ... 350
Послесловие ... 351

Приложение A. SSH ... 352
«Родной» SSH ... 352
SSH-агент ... 352
Запуск ssh-agent ... 353
macOS ... 353
Linux ... 354
Agent Forwarding ... 354
Команда sudo и перенаправление агента ... 356
Ключи хоста ... 357

Приложение B. Использование ролей IAM для учетных данных EC2 ... 361
Консоль управления AWS ... 361
Командная строка ... 362

Глоссарий ... 365

Библиография ... 368

Предметный указатель ... 369

Об авторах ... 380

Колофон ... 381

Я буквально проглотил рукопись первого издания «Установка и работа с Ansible» за несколько часов: Лорин прекрасно справился с задачей описания всех аспектов Ansible, и я был рад услышать, что он решил объединиться с Рене для подготовки второго издания. Эти два автора проделали громадную работу, чтобы показать нам, как пользоваться невероятно удобной утилитой, и я не могу вспомнить ни одного момента, которого бы они не охватили в полной мере.

– Ян-Пит Менс (Jan-Piet Mens), консультант

Впечатляющая глубина освещения Ansible. Эта книга прекрасно подойдет не только начинающим, но и опытным специалистам, желающим понять все тонкости использования продвинутых возможностей. Фантастический источник информации для стремящихся повысить свой уровень владения Ansible.

– Мэтт Джейнс (Matt Jaynes),
ведущий инженер, High Velocity Ops

Самое замечательное в Ansible – возможность начать с простого прототипа и быстро продвигаться к намеченной цели. Однако со временем начинает ощущаться нехватка знаний, которые порой трудно получить.

«Установка и работа с Ansible» – очень ценный источник, восполняющий эту нехватку и разъясняющий особенности Ansible с самых основ до сложностей работы с YAML и Jinja2. А благодаря наличию массы практичных примеров она позволяет получить представление, как другие автоматизируют свои окружения.

В течение последних нескольких лет, проводя теоретические и практические занятия, я всегда рекомендовал эту книгу своим коллегам и клиентам.

– Даг Виерс (Dag Wieers),
консультант и инженер-системотехник
в области систем на основе Linux,
долгое время участвовавший в разработке Ansible

Эта книга помогает быстро приступить к использованию системы управления конфигурациями Ansible и описывает ее во всех подробностях. В ней приводится большое количество подсказок и практических советов и охватывается широкий круг вариантов использования, включая AWS, Windows и Docker.

– Инго Йохим (Ingo Jochim),
руководитель отдела облачных реализаций, itelligence GMS/CIS

Лорин и Рене проделали большую работу, написав эту книгу. Авторы берут читателя за руку и ведут его через наиболее важные этапы создания и управления проектов Ansible. Эта книга намного больше, чем справочник, – она охватывает ряд важнейших концептуальных тем, отсутствующих в официальной документации. Это превосходный источник знаний для начинающих и практических идей для более опытных пользователей Ansible.

– Доминик Бартон (Dominique Barton),
инженер DevOps в confirm IT solutions

Предисловие

Разработка системы Ansible началась в феврале 2012-го с создания простого побочного проекта, и ее стремительное развитие стало приятным сюрпризом. Сейчас над продуктом работает порядка тысячи человек (а идеи принадлежат даже большему числу людей), и он широко используется практически во всех странах мира. И наверняка вам удастся обнаружить, по крайней мере, нескольких человек, использующих его, в сообществе знакомых вам ИТ-специалистов.

Привлекательность Ansible объясняется ее простотой. И правда, Ansible не несет в себе новых, но объединяет все лучшее из уже существующих идей, разработанных другими экспертами, делая их чуть более доступными.

Создавая Ansible, я старался найти для нее место где-то между решениями автоматизации ИТ-задач (естественная реакция на огромные коммерческие пакеты программного обеспечения) и простыми сценариями, минимально необходимыми для выполнения своей работы. Кроме того, мне хотелось заменить систему управления конфигурациями, развертыванием и организацией проектов и нашу библиотеку произвольных, но важных сценариев командной оболочки единой системой. Вот в чем состояла идея.

Могли ли мы убрать важные архитектурные компоненты из стека автоматизации? Устранив демоны управления и переложив работу на OpenSSH, система могла бы начать управление компьютерами незамедлительно без установки агентов на контролируемых машинах. Кроме того, система стала бы более надежной и безопасной.

Я заметил, что в предыдущих попытках создания систем автоматизации простые вещи заметно усложнялись, а написание сценариев автоматизации часто и надолго уводило меня в сторону от того, чему бы я хотел посвятить больше времени. В то же время мне не хотелось получить систему, на изучение которой не нужны месяцы.

Честно говоря, мне больше нравится писать программы, чем заниматься управлением системами автоматизации. Мне хотелось бы тратить на автоматизацию как можно меньше времени, чтобы высвободить его на решение более интересных задач. Ansible – это не та система, с которой приходится работать сутки напролет. Используя ее, вы сможете зайти, что-то поправить, выйти и продолжить заниматься своими делами. Я надеюсь, что эта черта Ansible понравится вам.

Хотя я потратил много времени, стараясь сделать документацию для Ansible исчерпывающей, всегда полезно взглянуть на одни и те же вещи под разными углами. Полезно увидеть практическое применение справочной документации. В книге «Установка и работа с Ansible» Лорин представляет Ansible, используя идиоматический подход, в точности как следовало бы изучать эту

систему. Лорин работал с Ansible практически с самого начала, и я очень благодарен ему за его вклад.

Я также безмерно благодарен каждому, кто принимал участие в проекте до настоящего времени, и каждому, кто подключится к нему будущем. Наслаждайтесь книгой и получайте удовольствие от управления вашим компьютерным флотом! И не забудьте установить cowsay!

– *Майкл Дехаан* (*Michael DeHaan*)
Создатель Ansible (программной части),
бывший технический директор компании Ansible, Inc.
апрель 2015

Предисловие ко второму изданию

За время, прошедшее с момента выхода первого издания (еще в 2014 году), в мире Ansible произошли большие изменения. Проект Ansible достиг следующей старшей версии 2.0. Также большие изменения произошли за рамками проекта: Ansible, Inc. – компания, стоящая за проектом Ansible, – была приобретена компанией Red Hat. Это никак не повлияло на разработку проекта Ansible: он так же активно развивается и привлекает новых пользователей.

Мы внесли множество изменений в это издание. Наиболее заметным стало появление пяти новых глав. Теперь книга охватывает плагины обратного вызова, хосты под управлением Windows, сетевое оборудование и Ansible Tower. Мы добавили в главу «Сложные сценарии» так много нового, что пришлось разбить ее на две части и добавить главу «Настройка хостов, запуск и обработчики». Мы также переписали главу «Docker», включив в нее описание новых модулей Docker.

Мы обновили все примеры кода для совместимости с Ansible 2.3. Например, устаревшую инструкцию `sudo` мы повсюду заменили более новой `become`. Мы также удалили ссылки на устаревшие модули, такие как `docker`, `ec2_vpc` и `ec2_ami_search`, и заменили их более новыми. Глава «Vagrant» теперь охватывает локальные сценарии вызова Ansible, глава «Amazon EC2» – Packer Ansible, механизм удаленного вызова, глава «Ускорение работы Ansible» – асинхронные задания, а глава «Отладка сценариев Ansible» – новые средства отладки, появившиеся в версии 2.1.

Также было внесено множество мелких изменений. Например, мы отказались от использования контрольных сумм MD5 в OpenSSH и перешли на хэши SHA256, внесли соответствующие изменения в примеры. Наконец, мы исправили ошибки, помеченные нашими читателями.

Примечание к стилю изложения

Первое издание книги было написано одним автором, и в нем часто использовалось местоимение «я» первого лица. Это издание написано уже двумя авторами, поэтому употребление местоимения в первом лице кое-где может показаться странным. Тем не менее мы решили не исправлять его, потому что в большинстве случаев оно используется для выражения мнения одного из авторов.

Благодарности

От Лорин

Мои благодарности Яну-Пит Менсу (Jan-Piet Mens), Мэтту Джейнсу (Matt Jaynes) и Джону Джарвису (John Jarvis) за отзывы в процессе написания книги. Спасибо Айзаку Салдана (Isaac Saldana) и Майку Ровану (Mike Rowan) из SendGrid за поддержку этого начинания. Благодарю Майкла ДеХаана (Michael DeHaan) за создание Ansible и поддержку сообщества, которое разрослось вокруг продукта, а также за отзыв о книге, включая объяснения, почему в качестве названия было выбрано *Ansible*. Спасибо моему редактору Брайану Андерсону (Brian Anderson) за его безграничное терпение в работе со мной.

Спасибо маме и папе за их неизменную поддержку; моему брату Эрику (Eric), настоящему писателю в нашей семье; двум моим сыновьям Бенджамину (Benjamin) и Джулиану (Julian). И наконец, спасибо моей жене Стейси (Stacy) за все.

От Рене

Спасибо моей семье, моей жене Симоне (Simone) за любовь и поддержку, моим трем деткам, Джил (Gil), Сарине (Sarina) и Лиан (Léanne), за свет и радость, что они привнесли в мою жизнь; спасибо всем, кто внес свой вклад в развитие Ansible, спасибо вам за ваш труд; и особое спасибо Маттиасу Блейзеру (Matthias Blaser), познакомившему меня с Ansible.

Предисловие к первому изданию

Почему я написал эту книгу

Когда я писал свое первое веб-приложение, используя Django, популярный фреймворк на Python, я запомнил чувство удовлетворения, когда приложение наконец-то заработало на моем компьютере. Я запустил команду `django manage.py runserver`, указал в браузере *http://localhost:8000* и увидел свое веб-приложение во всей его красе.

Потом я подумал про все эти… *моменты*, которые необходимо учесть, чтобы просто запустить это чертово приложение на Linux-сервере. Кроме Django и моего приложения, мне потребовалось установить Apache и модуль `mod_python`, чтобы Apache мог работать с приложениями Django. Затем мне пришлось установить правильные значения в конфигурационном файле Apache, заставлявшие мое приложение работать и правильно обслуживать статичные компоненты.

Это было несложно – немного усилий, и готово. Мне не хотелось завязнуть в работе с файлами конфигурации, я лишь хотел, чтобы мое приложение работало. И оно работало, и все было прекрасно… пока через несколько месяцев мне не понадобилось запустить его снова на другом сервере и проделать всю ту же работу с самого начала.

В конце концов, я осознал, что все, что я делал, я делал неправильно. Правильный способ решать такого рода задачи имеет название, и это название – *управление конфигурациями*. Самое замечательное в управлении конфигурациями – полученные знания всегда сохраняют свою актуальность. Больше нет необходимости рыться в поисках нужной страницы в документации или копаться в старых записях.

Недавно коллега заинтересовался применением Ansible для внедрения нового проекта и спросил, как можно использовать идею Ansible на практике, кроме того что указано в официальной документации. Я не знал, что посоветовать почитать, и решил написать книгу, которая восполнит этот пробел, – и вот вы видите эту книгу перед собой. Увы, для него эта книга вышла слишком поздно, но я надеюсь, она окажется полезной для вас.

Кому адресована эта книга

Эта книга для всех, кто работает с Linux- или Unix-подобными серверами. Если вы когда-либо использовали термины *системное администрирование*, *раз-*

вертывание, *управление конфигурациями* или (вздох) *DevOps*, вы обязательно найдете для себя что-то полезное.

Хотя я изучал Linux-серверы, моя квалификация связана с разработкой программного обеспечения. А это значит, что все примеры в книге более тяготеют к внедрению программного обеспечения, хотя мы с Эндрю Клей Шафером (Andrew Clay Shafer, [webops]) пришли к тому, что внедрение и конфигурация не имеют четкой границы.

Структура книги

Я не большой фанат общепринятых принципов структурирования книг: глава 1 охватывает то-то и то-то, глава 2 охватывает это и то и тому подобное. Я подозреваю, что никто не читает этих строк (я лично – никогда), гораздо проще заглянуть в оглавление.

Книга построена так, что каждая последующая глава опирается на предыдущую. Таким образом, я предполагаю, что вы будете читать книгу от начала и до конца. Книга написана в основном в стиле учебного пособия и дает возможность выполнять примеры на вашем компьютере в процессе чтения. Большинство примеров основано на веб-приложениях.

Обозначения и соглашения, принятые в этой книге

В книге действуют следующие типографские соглашения:

Курсив
: Указывает на новые термины, названия файлов и их расширения.

`Моноширинный шрифт`
: Используется для 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`
: Служит для выделения команд или другого текста, который должен быть набран самим пользователем.

`Моноширинный курсив`
: Указывает на текст, который нужно заменить данными пользователя, или значениями, определяемыми контекстом.


Так обозначаются примечания общего характера.


Так обозначаются советы и рекомендации.


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: «*Хохштейн Л., Мозер Р.* Запускаем Ansible. М.: O'Reilly; ДМК Пресс, 2018. Copyright © 2017 O'Reilly Media, Inc., 978-1-491-97980-8 (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.

Отзывы и пожелания

Мы всегда рады отзывам наших читателей. Расскажите нам, что вы думаете об этой книге – что понравилось или, может быть, не понравилось. Отзывы важны для нас, чтобы выпускать книги, которые будут для вас максимально полезны.

Вы можете написать отзыв прямо на нашем сайте www.dmkpress.com, зайдя на страницу книги, и оставить комментарий в разделе «Отзывы и рецензии». Также можно послать письмо главному редактору по адресу dmkpress@gmail.com, при этом напишите название книги в теме письма.

Если есть тема, в которой вы квалифицированы, и вы заинтересованы в написании новой книги, заполните форму на нашем сайте по адресу http://dmkpress.com/authors/publish_book/ или напишите в издательство 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[1]. Вместо этого мы медленно превращаемся в системных инженеров.

Сейчас мы устанавливаем 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 кэширования данных в оперативной памяти, очереди задач, очереди сообщений, базы данных SQL, системы хранения данных, NoSQL-хранилища и балансировщики нагрузки.

Мы также должны убедиться в наличии достаточного количества ресурсов, и в случае падения системы (а она будет падать) 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[2].

Мы можем связать эти службы вручную: «прикрутить» нужные серверы, зайдя на каждый из них, установив пакеты приложений, отредактировав файлы конфигурации, и т. д. Но это серьезный труд. Такой процесс требует много времени, способствует появлению множества ошибок, да 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 управления. Кем

---

[1] Да, мы согласны, никто и никогда на самом деле так не поставлял программное обеспечение.

[2] Рекомендую превосходные книги «The Practice of Cloud System Administration» и «Designing Data-Intensive Applications» по созданию и поддержке этих типов распределенных систем.

бы вы ни были, разработчиком или системным администратором, ищущим лучшего 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 2.3.0.0, которая на момент написания книги являлась самой свежей. Поскольку поддержка предыдущих версий является важной целью проекта Ansible, эти примеры должны поддерживаться и последующими версиями в неизменном виде.

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 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

Систему Ansible часто описывают как средство управления конфигурациями, и обычно она упоминается в том же контексте, что и *Chef*, *Puppet* и *Salt*. Когда мы говорим об управлении конфигурациями, то часто подразумеваем некое описательное состояние серверов, а затем фиксацию их реального состояния с использованием специальных средств: необходимые пакеты приложений установлены, файлы конфигурации содержат ожидаемые значения и имеют требуемые разрешения в файловой системе, необходимые службы работают 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) и запуск служб. *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 об *инициализации* (provisioning) новых серверов. В контексте облачных услуг, таких как Amazon EC2, под инициализацией подразумевается развертывание нового экземпляра виртуальной машины. Ansible охватывает и эту область, предоставляя несколько модулей поддержки облаков, включая EC2, Azure, Digital Ocean, Google Compute Engine, Linode и Rackspace, а также любые облака, поддерживающие OpenStack API.

Несколько сбивает с толку использование термина *инициатор* в документации к утилите *Vagrant*, которую мы обсудим далее в этой главе, в отношении системы управления конфигурациями. Так, Vagrant называет Ansible своего рода инициатором там, где, как мне кажется, инициатором 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 Стейси, применяет 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 (Ansible называет их *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 файлы конфигурации для Nginx;
- скопировать сертификат безопасности;
- запустить Nginx.

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

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

Ansible устанавливает параллельные SSH-соединения с хостами *web1*, *web2* и *web3*. Выполняет первую задачу из списка на всех хостах одновременно.

В этом примере первая задача – установка apt-пакета Nginx (поскольку Ubuntu использует диспетчер пакетов `apt`). То есть данная задача в сценарии выглядит примерно так:

```
- name: install nginx
  apt: 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.

Далее Ansible приступит к следующей задаче в списке и повторит описанные эти же четыре шага. Важно отметить, что:

- каждая задача выполняется на всех хостах одновременно;
- Ansible ожидает, пока задача будет завершена на всех хостах, прежде чем приступить к выполнению следующей;
- задачи выполняются в установленном вами порядке.

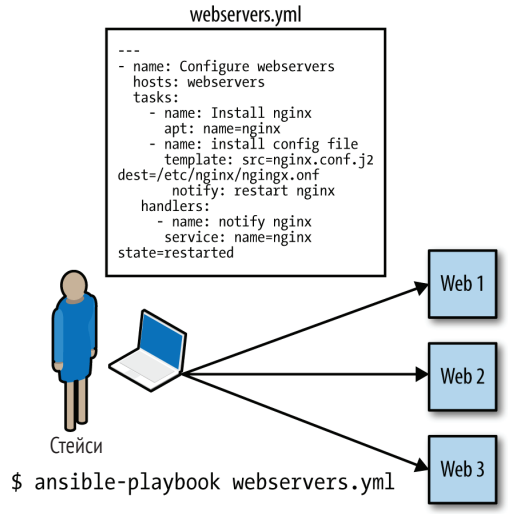

Рис. 1.1 ❖ Ansible выполняет сценарий настройки трех веб-серверов

Какие преимущества дает Ansible?

Существует несколько открытых систем управления конфигурациями. Ниже перечисляются некоторые особенности, привлекшие мое внимание к Ansible.

Простота синтаксиса

Напомню, что задачи управления конфигурациями в Ansible определяются в виде *сценариев* (playbooks). 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.

Мне нравится думать про сценарии Ansible как про выполняемую документацию. 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.

Отсутствие необходимости установки на удаленных хостах

Для управления серверами с помощью Ansible на них должна быть установлена поддержка SSH и Python версии 2.5 или выше либо Python 2.4 с библиотекой *simplejson*. Нет никакой необходимости устанавливать на хостах любое другое программное обеспечение.

На управляющей машине (той, что вы используете для управления удаленными машинами) должен быть установлен Python версии 2.6 или выше.

Некоторые модули могут потребовать установки Python версии 2.5 или выше, другие могут иметь иные требования. Обязательно проверяйте документацию по каждому модулю, чтобы понять, имеет ли он специфические требования.

Основан на 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 центральный 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.

Принудительная настройка дает важное преимущество – вы контролируете время обновления серверов. Вам не приходится ждать. Сторонники добровольной настройки утверждают, что их подход лучше масштабируется на большое число серверов и удобнее, когда новые серверы могут появиться в любой момент. Однако отложим эту дискуссию на потом, а пока отмечу, что Ansible с успехом использовался для управления тысячами узлов и показал отличные результаты в сетях с динамически добавляемыми и удаляемыми серверами.

Если вам действительно нравится модель, основанная на приемах добровольной настройки, для вас Ansible официально поддерживает особый режим, называемый *ansible-pull*. Я не раскрываю особенностей этого режима в рамках данной книги. Но вы можете узнать больше об этом из официальной документации: http://docs.ansible.com/ansible/playbooks_intro.html#ansible-pull.

Управление небольшим числом серверов

Да, Ansible можно использовать для управления сотнями и даже тысячами узлов. Но управлять единственным узлом с помощью Ansible также очень легко – вам нужно лишь написать один сценарий. Ansible подтверждает принцип Алана Кея: «Простое должно оставаться простым, а сложное – возможным».

Встроенные модули

Ansible можно использовать для выполнения произвольных команд оболочки на удаленных серверах, но его действительно мощной стороной является набор модулей. Модули необходимы для выполнения таких задач, как установка пакетов приложений, перезапуск службы или копирование файлов конфигурации.

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`:

```
user: name=deploy group=web
```

Модули также являются *идемпотентными*[1]. Если пользователь `deploy` не существует, Ansible создаст его. Если он существует, Ansible просто перейдет к следующему шагу. То есть сценарии Ansible можно запускать на сервере много раз. Это большое усовершенствование, по сравнению с подходом на основе сценариев командной оболочки, потому что повторный запуск таких сценариев может привести к незапланированным и хорошо, если безобидным последствиям.

---

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

Как обстоит дело с конвергенцией?

В книгах по управлению конфигурациями часто упоминается идея *конвергенции* (или сходимости), которая нередко ассоциируется с именем Марка Бургесса (Mark Burgess) и его системой управления конфигурациями *CFEngine*. Если система управления конфигурациями конвергентна, 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.

Если вам интересно, что думает автор Ansible об идее конвергенции, прочтите публикацию Майкла ДеХаана «Идемпотентность, конвергенция и другие причудливые слова, которые мы используем слишком часто» («Idempotence, convergence, and other silly fancy words we use too often») на странице группы Ansible Project: https://bit.ly/1InGh1A.

Очень тонкий слой абстракции

Некоторые системы управления конфигурациями предоставляют уровень абстракции настолько мощный, что позволяют использовать одни и те же сценарии для управления серверами с разными операционными системами. Например, вместо конкретных диспетчеров пакетов, таких как yum или apt, можно использовать абстракцию «пакет», поддерживаемую системой управления конфигурациями.

Ansible работает не так – для установки пакетов в системы, основанные на диспетчере apt, вы должны использовать диспетчер apt, а в системы, основанные на диспетчере yum, – диспетчер yum.

На практике это упрощает использование Ansible, хотя на первый взгляд может показаться недостатком. Ansible не требует изучения новых наборов абстракций, нивелирующих разницу между операционными системами. Это сокращает объем документации для изучения перед началом написания сценариев.

При желании вы можете писать собственные сценарии Ansible для выполнения определенных действий, в зависимости от операционной системы на удаленном сервере. Но я стараюсь избегать этого, концентрируя свое внимание на написании сценариев для конкретных операционных систем, таких как Ubuntu.

Модуль является основной единицей повторного использования в сообществе Ansible. Поскольку область применения модуля ограничена и зависит от определенной операционной системы, это позволяет писать качественные и надежно работающие модули. Проект Ansible всегда открыт для новых модулей, предлагаемых сообществом. Я это знаю, поскольку сам предложил несколько.

Сценарии Ansible не предназначены для использования в разных контекстах. В главе 7 мы обсудим *роли* как средство организации сценариев для повторного использования. Также мы обсудим Ansible Galaxy – онлайн-репозиторий ролей.

Однако на практике каждая организация сервера настраивается с некоторыми отличиями, поэтому лучше постараться написать сценарии для своей компании, чем пытаться использовать универсальные. Единственный повод для изучения чужих сценариев – это, например, взглянуть, как и что было сделано.

Связь между Ansible и Ansible, Inc.

Название *Ansible* относится как к программному обеспечению, так и к компании, ведущей проект. 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-го Red Hat купила Ansible Inc.

Не слишком ли проста система Ansible?

В период работы над книгой мой редактор сказал мне, что «некоторые специалисты, использующие систему управления конфигурациями XYZ, называют Ansible «циклом for по сценариям». 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 широкую функциональность, 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.

А как насчет масштабируемости SSH? В главе 12 будет показано, что Ansible применяет SSH-мультиплексирование для оптимизации производительности. Некоторые специалисты используют Ansible для управления тысячами узлов[1].

[1] Например, ознакомьтесь с материалом «Использование Ansible для управления масштабируемым публичным облаком» («Using Ansible at Scale to Manage a Public Cloud») от Jesse Keating, бывшего сотрудника Rackspace.

Я не настолько хорошо знаком с остальными системами, чтобы рассматривать их различия в деталях. Если вам необходим детальный сравнительный анализ систем управления конфигурациями, прочитайте книгу «Taste Test: Puppet, Chef, Salt, Ansible» Мэтта Джейнса (Matt Jaynes). Так случилось, что Мэтт предпочел Ansible.

Что я должен знать?

Для эффективной работы с Ansible необходимо знать основы администрирования операционной системы Linux. Ansible позволяет автоматизировать процессы, но не выполняет волшебным образом тех из них, с которыми вы не справляетесь.

Предполагаю, что 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 среды;
- писать сценарии (на любом языке).

Если все это вам известно, 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.

Ansible использует формат файлов YAML и язык шаблонов Jinja2. Следовательно, вам необходимо изучить их, но обе технологии просты в освоении.

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 200, и они достаточно хорошо представлены в официальной документации.

Книга охватывает только основные возможности механизма шаблонов Jinja2, поскольку их вполне достаточно для работы с Ansible. Для более глубокого изучения Jinja2 я рекомендую обратиться к официальной документации по Jinja2 на странице http://jinja.pocoo.org/docs/dev/.

Книга не дает детального описания функций Ansible, используемых в основном для работы в ранних версиях Linux. Сюда относятся клиент SSH *Paramiko* и *ускоренный режим*.

Наконец, я не рассматриваю некоторых функций Ansible I, чтобы сохранить размер книги в разумных пределах. К ним относятся: режим обновления конфигурации по инициативе клиентов, журналирование, соединение с хостами по протоколам, отличным от SSH, и запрос у пользователя паролей и другой информации.

Установка Ansible

На сегодняшний день все основные дистрибутивы Linux включают пакет Ansible. Поэтому, если вы работаете в Linux, вы сможете установить его, используя «родной» диспетчер пакетов. Но имейте в виду, что это может быть не самая последняя версия Ansible. Если вы работаете в Mac OS X, я рекомендую использовать замечательный диспетчер пакетов Homebrew.

Если такого пакета в вашей версии ОС нет, вы можете установить Ansible с помощью *pip*, диспетчера пакетов Python, выполнив следующую команду:

```
$ sudo pip install ansible
```

При желании Ansible можно установить в локальное *виртуальное окружение* Python (virtualenv). Если вы незнакомы с виртуальными окружениями, можете использовать более новый инструмент под названием *pipsi*. Он автоматически создаст новое виртуальное окружение и установит в него Ansible:

```
$ wget https://raw.githubusercontent.com/mitsuhiko/pipsi/master/get-pipsi.py
$ python get-pipsi.py
$ pipsi install ansible
```

Если вы решите воспользоваться pipsi, добавьте путь *~/.local/bin* в переменную окружения PATH. Некоторые плагины и модули Ansible могут потребовать установки дополнительных библиотек Python. Если вы произвели установку с помощью pipsi и хотели бы установить *docker-py* (необходимый для модулей из библиотеки Ansible Docker) и *boto* (необходимый для модулей из библиотеки Ansible EC2), выполните следующие команды:

```
$ cd ~/.local/venvs/ansible
$ source bin/activate
$ pip install docker-py boto
```

Если вам интересно испытать в работе новейшую версию Ansible, загрузите ее из GitHub:

```
$ git clone https://github.com/ansible/ansible.git –recursive
```

При работе с этой версией вам каждый раз будет нужно выполнять следующие команды, чтобы установить переменные окружения, включая переменную PATH, чтобы оболочка смогла находить программы *ansible* и *ansible-playbooks*.

```
$ cd ./ansible
$ source ./hacking/env-setup
```

Дополнительную информацию об установке можно найти на следующих ресурсах:

- официальная документация по установке Ansible (http://docs.ansible.com/ansible/intro_installation.html);
- pip (http://pip.readthedocs.io/en/stable/);
- virtualenv (http://docs.python-guide.org/en/latest/dev/virtualenvs/);
- pipsi (https://github.com/mitsuhiko/pipsi).

Подготовка сервера для экспериментов

Для выполнения примеров, приведенных в книге, вам необходимо иметь SSH-доступ и права пользователя root на сервере Linux. К счастью, сегодня легко получить недорогой доступ к виртуальной машине Linux в общедоступных службах облачных услуг, таких как Amazon EC2, Google Compute Engine, Microsoft Azure[1], Digital Ocean, Linode.., в общем, вы поняли.

Использование Vagrant для подготовки сервера

Если вы предпочитаете не тратиться на облачные услуги, я предложил бы установить Vagrant – отличный инструмент управления виртуальными машинами с открытым кодом. С его помощью можно запустить виртуальную машину с Linux на ноутбуке. Она и послужит вам сервером для экспериментов.

В Vagrant имеется встроенная возможность подготовки виртуальных машин с Ansible. Подробнее об этом будет рассказано в главе 3. А пока будем считать виртуальную машину под управлением Vagrant обычным сервером Linux.

Vagrant требует установки VirtualBox. Скачайте VirtualBox (https://www.virtualbox.org/), а затем Vagrant (https://www.vagrantup.com/).

Рекомендую создать отдельный каталог для сценариев Ansible и прочих файлов. В следующем примере я создал такой каталог с именем *playbooks*.

Выполните следующие команды, чтобы создать файл конфигурации Vagrant (Vagrantfile) для 64-битого образа виртуальной машины[2] с Ubuntu 14.04 (Trusty Tahr) и загрузить ее.

```
$ mkdir playbooks
$ cd playbooks
$ vagrant init ubuntu/trusty64
$ vagrant up
```

При первом запуске команда `vagrant up` загрузит файл образа виртуальной машины. На это может потребоваться некоторое время в зависимости от качества соединения с Интернетом.

В случае успеха вы увидите, как в окне терминала побегут следующие строки:

---

[1] Да, Azure поддерживает серверы Linux.

[2] Виртуальная машина в терминологии Vagrant называется *machine*, а ее образ – *box*.

```
Bringing machine 'default' up with 'virtualbox' provider...
==> default: Importing base box 'ubuntu/trusty64'...
==> default: Matching MAC address for NAT networking...
==> default: Checking if box 'ubuntu/trusty64' is up to date...
==> default: Setting the name of the VM: playbooks_default_1474348723697_56934
==> default: Clearing any previously set forwarded ports...
==> default: Clearing any previously set network interfaces...
==> default: Preparing network interfaces based on configuration...
    default: Adapter 1: nat
==> default: Forwarding ports...
    default: 22 (guest) => 2222 (host) (adapter 1)
==> default: Booting VM...
==> default: Waiting for machine to boot. This may take a few minutes...
    default: SSH address: 127.0.0.1:2222
    default: SSH username: vagrant
    default: SSH auth method: private key
    default: Warning: Remote connection disconnect. Retrying...
    default: Warning: Remote connection disconnect. Retrying...
    default:
    default: Vagrant insecure key detected. Vagrant will automatically replace
    default: this with a newly generated keypair for better security.
    default:
    default: Inserting generated public key within guest...
    default: Removing insecure key from the guest if it's present...
    default: Key inserted! Disconnecting and reconnecting using new SSH key...
==> default: Machine booted and ready!
==> default: Checking for guest additions in VM...
    default: The guest additions on this VM do not match the installed version
    default: of VirtualBox! In most cases this is fine, but in rare cases it can
    default: prevent things such as shared folders from working properly. If you
    default: see shared folder errors, please make sure the guest additions
    default: within the virtual machine match the version of VirtualBox you have
    default: installed on your host and reload your VM.
    default:
    default: Guest Additions Version: 4.3.36
    default: VirtualBox Version: 5.0
==> default: Mounting shared folders...
    default: /vagrant => /Users/lorin/dev/ansiblebook/ch01/playbooks
```

Теперь можно попробовать зайти по SSH на вашу новую виртуальную машину Ubuntu 14.04, выполнив следующую команду:

```
$ vagrant ssh
```

Если все прошло благополучно, вы увидите экран с приветствием:

```
Welcome to Ubuntu 14.04.5 LTS (GNU/Linux 3.13.0-96-generic x86_64)

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

  System information as of Fri Sep 23 05:13:05 UTC 2016

  System load:  0.76              Processes:           80
```

```
  Usage of /:   3.5% of 39.34GB  Users logged in:     0
  Memory usage: 25%              IP address for eth0: 10.0.2.15
  Swap usage:   0%

  Graph this data and manage this system at:
    https://landscape.canonical.com/

  Get cloud support with Ubuntu Advantage Cloud Guest:
    http://www.ubuntu.com/business/services/cloud

0 packages can be updated.
0 updates are security updates.

New release '16.04.1 LTS' available.
Run 'do-release-upgrade' to upgrade to it.
```

Введите **exit**, чтобы завершить сеанс SSH.

Этот подход позволяет взаимодействовать с командной оболочкой. Однако Ansible требует подключения к виртуальной машине посредством SSH-клиента, а не команды `vagrant ssh`.

Попросите Vagrant вывести на экран детали SSH-подключения:

```
$ vagrant ssh-config
```

Я у себя получил такой результат:

```
Host default
  HostName 127.0.0.1
  User vagrant
  Port 2222
  UserKnownHostsFile /dev/null
  StrictHostKeyChecking no
  PasswordAuthentication no
  IdentityFile /Users/lorin/dev/ansiblebook/ch01/playbooks/.vagrant/
  machines/default/virtualbox/private_key
  IdentitiesOnly yes
  LogLevel FATAL
```

Вот самые важные строки:

```
HostName 127.0.0.1
User vagrant
Port 2222
IdentityFile /Users/lorin/dev/ansiblebook/ch01/playbooks/.vagrant/
machines/default/virtualbox/private_key
```

В Vagrant 1.7 изменился порядок работы с приватными SSH-ключами. Начиная с этой версии Vagrant генерирует новый приватный ключ для каждой машины. Более ранние версии использовали один и тот же ключ, который по умолчанию хранился в каталоге *~/.vagrant.d/insecure_private_key*. Примеры в этой книге основаны на Vagrant 1.7.

У вас строки должны выглядеть похоже, за исключением места хранения файла идентификации.

Убедитесь, что сможете начать новый SSH-сеанс из командной строки, используя эту информацию. В моем случае команда выглядит так:

```
$ ssh vagrant@127.0.0.1 -p 2222 -i /Users/lorin/dev/ansiblebook/ch01/
playbooks/.vagrant/machines/default/virtualbox/private_key
```

Вы должны увидеть экран входа в Ubuntu. Введите **exit**, чтобы завершить SSH-сеанс.

Передача информации о сервере в Ansible

Ansible может управлять только известными ей серверами. Передать информацию о серверах в Ansible можно в файле реестра.

Каждому серверу должно быть присвоено имя для идентификации в Ansible. С этой целью можно использовать имя хоста или выбрать другой псевдоним. С именем также должны определяться дополнительные параметры подключения. Присвоим нашему серверу псевдоним `testserver`.

Создайте в каталоге *playbooks* файл с именем *hosts*. Он будет служить реестром. Если в качестве тестового сервера вы используете виртуальную машину Vagrant, файл *hosts* должен выглядеть, как в примере 1.1. Я разбил содержимое файла на несколько строк, чтобы уместить его по ширине страницы. В действительности информация в файле представлена одной строкой без обратных косых.

Пример 1.1 ❖ Файл playbooks/hosts

```
testserver ansible_host=127.0.0.1 ansible_port=2222 \
  ansible_user=vagrant \
  ansible_private_key_file=.vagrant/machines/default/virtualbox/private_key
```

Здесь можно видеть один из недостатков использования Vagrant: мы вынуждены явно передать дополнительные аргументы, чтобы сообщить Ansible параметры подключения. В большинстве случаев в этих дополнительных данных нет необходимости.

Далее в этой главе вы увидите, как использовать файл *ansible.cfg*, чтобы избежать нагромождения информации в файле реестра. В последующих главах вы увидите, как с той же целью можно использовать переменные Ansible.

Если предположить, что у вас есть Ubuntu-машина в облаке Amazon EC2 с именем хоста `ec2-203-0-113-120.compute-1.amazonaws.com`, содержимое файла реестра будет выглядеть так (все в одну строку):

```
testserver ansible_host=ec2-203-0-113-120.compute-1.amazonaws.com \
  ansible_user=ubuntu ansible_private_key_file=/path/to/keyfile.pem
```

Ansible поддерживает программу ssh-agent, поэтому нет необходимости явно указывать файлы SSH-ключей в реестре. Если прежде вам не доводилось пользоваться этой программой, более детальную информацию о ней вы найдете в разделе «Агент SSH» в приложении A.

Чтобы проверить способность Ansible подключиться к серверу, используем утилиту командной строки `ansible`. Мы будем изредка пользоваться ею, в основном для решения специфических задач.

Попросим Ansible установить соединение с сервером `testserver`, указанным в файле реестра *hosts*, и вызвать модуль `ping`:

```
$ ansible testserver -i hosts -m ping
```

Если на локальном SSH-клиенте включена проверка ключей хоста, вы увидите нечто, похожее на первую попытку Ansible подключиться к серверу:

```
The authenticity of host '[127.0.0.1]:2222 ([127.0.0.1]:2222)' \
can't be established.
RSA key fingerprint is e8:0d:7d:ef:57:07:81:98:40:31:19:53:a8:d0:76:21.
Are you sure you want to continue connecting (yes/no)?
```

Просто введите **yes**.

В случае успеха появится следующий результат:

```
testserver | success >> {
  "changed": false,
  "ping": "pong"
}
```

Если Ansible сообщит об ошибке, добавьте в команду флаг `-vvvv`, чтобы получить больше информации об ошибке:

```
$ ansible testserver -i hosts -m ping -vvvv
```

Мы видим, что команда выполнилась успешно. Часть ответа `"changed": false` говорит о том, что выполнение модуля не изменило состояния сервера. Текст `"ping":"pong"` является характерной особенностью модуля `ping`.

Модуль `ping` не производит никаких изменений. Он лишь проверяет способность Ansible начать SSH-сеанс с сервером.

Упрощение задачи с помощью файла ansible.cfg

Нам пришлось ввести много текста в файл реестра, чтобы сообщить системе Ansible информацию о тестовом сервере. К счастью, Ansible поддерживает несколько способов передачи такой информации, и мы не обязаны группировать ее в одном месте. Сейчас мы воспользуемся одним из таких способов – файлом *ansible.cfg* – для определения некоторых настроек по умолчанию, чтобы потом нам не пришлось набирать так много текста.

Где лучше хранить файл ansible.cfg?

Ansible будет искать файл *ansible.cfg* в следующих местоположениях в указанном порядке:

1. Файл, указанный в переменной окружения `ANSIBLE_CONFIG`.
2. *./ansible.cfg* (*ansible.cfg* в текущем каталоге).

3. *~/.ansible.cfg* (*.ansible.cfg* в вашем домашнем каталоге).
4. */etc/ansible/ansible.cfg*.

Я обычно храню *ansible.cfg* в текущем каталоге, вместе со сценариями. Это позволяет хранить его в том же репозитории, где хранятся мои сценарии.

Пример 1.2 показывает, как в файле *ansible.cfg* определяются местоположение файла реестра (`inventory`), имя пользователя SSH (`remote_user`) и приватный ключ SSH (`private_key_file`). Эти настройки предполагают использование Vagrant. При использовании отдельного сервера необходимо установить только значения `remote_user` и `private_key_file`.

В нашем примере конфигурации проверка SSH-ключей хоста отключена. Это удобно при работе с Vagrant. В противном случае необходимо вносить изменения в файл *~/.ssh/known_hosts* каждый раз, когда удаляется имеющийся или создается новый Vagrant-сервер. Однако отключение проверки ключей для серверов в сети несет определенные риски. Если вы незнакомы с аутентификацией при помощи ключей хоста, то можете прочитать об этом в приложении А.

Пример 1.2 ❖ ansible.cfg

```
[defaults]
inventory = hosts
remote_user = vagrant
private_key_file = .vagrant/machines/default/virtualbox/private_key
host_key_checking = False
```

Ansible и система управления версиями

Ansible по умолчанию хранит реестр в файле */etc/ansible/hosts*. Однако лично я предпочитаю хранить его вместе с моими сценариями в системе управления версиями.

Хотя работа с такими системами не затрагивается в этой книге, я настоятельно рекомендую использовать для управления сценариями систему, подобную Git. Если вы разработчик программного обеспечения, то уже знакомы с системами управления версиями. Если вы системный администратор и прежде не пользовались ими, тогда это хороший повод начать знакомство.

С настройками по умолчанию отпадает необходимость указывать имя пользователя или файл с ключами SSH в файле *hosts*. Запись упрощается до:

```
testserver ansible_ host=127.0.0.1 ansible_ port=2222
```

Мы также можем запустить Ansible без аргумента `-i hostname`:

```
$ ansible testserver -m ping
```

Мне нравится использовать инструмент командной строки `ansible` для запуска произвольных команд на удаленных серверах. Произвольные команды также можно выполнять с помощью модуля `command`. При запуске модуля необходимо указать аргумент `-a` с запускаемой командой.

Например, вот как можно проверить время работы сервера с момента последнего запуска:

```
$ ansible testserver -m command -a uptime
```

Результат должен выглядеть примерно так:

```
testserver | success | rc=0 >>
 17:14:07 up 1:16,   1 user,  load average: 0.16, 0.05, 0.04
```

Модуль `command` настолько часто используется, что сделан модулем по умолчанию, то есть его имя можно опустить в команде:

```
$ ansible testserver -a uptime
```

Если команда в аргументе `-a` содержит пробелы, ее необходимо заключить в кавычки, чтобы командная оболочка передала всю строку как единый аргумент. Для примера вот как выглядит извлечение нескольких последних строк из журнала */var/log/dmesg*:

```
$ ansible testserver -a "tail /var/log/dmesg"
```

Вывод, возвращаемый машиной Vagrant, выглядит следующим образом:

```
testserver | success | rc=0 >>
[    5.170544] type=1400 audit(1409500641.335:9): apparmor="STATUS" operation=
"profile_replace" profile="unconfined" name="/usr/lib/NetworkManager/nm-dhcp-c
lient.act on" pid=888 comm="apparmor_parser"
[   5.170547] type=1400 audit(1409500641.335:10): apparmor="STATUS" operation=
"profile_replace" profile="unconfined" name="/usr/lib/connman/scripts/dhclientscript"
pid=888 comm="apparmor_parser"
[   5.222366] vboxvideo: Unknown symbol drm_open (err 0)
[   5.222370] vboxvideo: Unknown symbol drm_poll (err 0)
[   5.222372] vboxvideo: Unknown symbol drm_pci_init (err 0)
[   5.222375] vboxvideo: Unknown symbol drm_ioctl (err 0)
[   5.222376] vboxvideo: Unknown symbol drm_vblank_init (err 0)
[   5.222378] vboxvideo: Unknown symbol drm_mmap (err 0)
[   5.222380] vboxvideo: Unknown symbol drm_pci_exit (err 0)
[   5.222381] vboxvideo: Unknown symbol drm_release (err 0)
```

Чтобы выполнить команду с привилегиями root, нужно передать параметр `-b`. В этом случае Ansible выполнит команду от лица пользователя root. Например, для доступа к */var/log/syslog* требуются привилегии root:

```
$ ansible testserver -b -a "tail /var/log/syslog"
```

Результат будет выглядеть примерно так:

```
testserver | success | rc=0 >>
Aug 31 15:57:49 vagrant-ubuntu-trusty-64 ntpdate[1465]: /
adjust time server 91.189
94.4 offset -0.003191 sec
Aug 31 16:17:01 vagrant-ubuntu-trusty-64 CRON[1480]: (root) CMD ( cd /
&& run-p
rts --report /etc/cron.hourly)
Aug 31 17:04:18 vagrant-ubuntu-trusty-64 ansible-ping: Invoked with data=None
```

```
Aug 31 17:12:33 vagrant-ubuntu-trusty-64 ansible-ping: Invoked with data=None
Aug 31 17:14:07 vagrant-ubuntu-trusty-64 ansible-command: Invoked with executable
None shell=False args=uptime removes=None creates=None chdir=None
Aug 31 17:16:01 vagrant-ubuntu-trusty-64 ansible-command: Invoked with executable
None shell=False args=tail /var/log/messages removes=None creates=None chdir=None
Aug 31 17:17:01 vagrant-ubuntu-trusty-64 CRON[2091]: (root) CMD ( cd /
&& run-pa
rts --report /etc/cron.hourly)
Aug 31 17:17:09 vagrant-ubuntu-trusty-64 ansible-command: Invoked with /
executable=
N one shell=False args=tail /var/log/dmesg removes=None creates=None chdir=None
Aug 31 17:19:01 vagrant-ubuntu-trusty-64 ansible-command: Invoked with /
executable=
None shell=False args=tail /var/log/messages removes=None creates=None chdir=None
Aug 31 17:22:32 vagrant-ubuntu-trusty-64 ansible-command: Invoked with /
executable=
one shell=False args=tail /var/log/syslog removes=None creates=None chdir=None
```

Как видите, Ansible фиксирует свои действия в syslog.

Утилита `ansible` не ограничивается модулями `ping` и `command`: вы можете использовать любой модуль по желанию. Например, следующей командой можно установить Nginx в Ubuntu:

```
$ ansible testserver -b -m apt -a name=nginx
```

Если установить Nginx не удалось, возможно, нужно обновить список пакетов. Чтобы Ansible выполнила эквивалент команды `apt-get update` перед установкой пакета, замените аргумент `name=nginx` на “`name=nginx update_cache=yes`”.

Перезапустить Nginx можно так:

```
$ ansible testserver -b -m service -a “name=nginx \
    state=restarted”
```

Поскольку только пользователь root может установить пакет Nginx и перезапустить службы, необходимо указать аргумент `-b`.

Что дальше

Вспомним, о чем рассказывалось в этой главе. Здесь мы рассмотрели основные понятия системы Ansible, включая взаимодействия с удаленными серверами, и отличия от других систем управления конфигурациями. Мы также увидели, как пользоваться утилитой командной строки `ansible` для выполнения простых задач на единственном хосте.

Однако использование `ansible` для выполнения команд на одном хосте не особенно интересно. В следующей главе мы рассмотрим действительно полезные сценарии.