Оглавление

Вступительное слово от компании ITSumma **19**
Описанные/рассмотренные в книге ресурсы Kubernetes **20**
Благодарности **22**
Предисловие **23**
Признательности **25**
Об этой книге **27**
Кто должен читать эту книгу 27
Как организована эта книга: дорожная карта 27
О коде 29
Книжный форум 30
Другие интернет-ресурсы 30
Об авторе **32**
Об иллюстрации на обложке **33**
Глава 1. Знакомство с Kubernetes **34**
1.1 Объяснение необходимости системы наподобие Kubernetes 36
1.1.1 Переход от монолитных приложений к микросервисам 36
1.1.2 Обеспечение консистентного окружения для приложений 39
1.1.3 Переход к непрерывной доставке: DevOps и NoOps 40
1.2 Знакомство с контейнерными технологиями 42
1.2.1 Что такое контейнеры 42
1.2.2 Знакомство с контейнерной платформой Docker 46
1.2.3 Знакомство с rkt – альтернативой Docker 50
1.3 Первое знакомство с Kubernetes 51
1.3.1 Истоки 51
1.3.2 Взгляд на Kubernetes с вершины горы 52
1.3.3 Архитектура кластера Kubernetes 54
1.3.4 Запуск приложения в Kubernetes 55
1.3.5 Преимущества использования Kubernetes 57
1.4 Резюме 60
Глава 2. Первые шаги с Docker и Kubernetes **61**
2.1 Создание, запуск и совместное использование образа контейнера 61
2.1.1 Установка Docker и запуск контейнера Hello World 62
2.1.2 Создание тривиального приложения Node.js 64
2.1.3 Создание файла Dockerfile для образа 65

2.1.4 Создание контейнерого образа ... 66
2.1.5 Запуск образа контейнера ... 68
2.1.6 Исследование работающего контейнера изнутри ... 69
2.1.7 Остановка и удаление контейнера ... 71
2.1.8 Отправка образа в хранилище образов ... 72
2.2 Настройка кластера Kubernetes ... 73
2.2.1 Запуск локального одноузлового кластера Kubernetes с помощью minikube ... 74
2.2.2 Использование кластера Kubernetes, предоставляемого как сервис с Google Kubernetes Engine ... 76
2.2.3 Настройка псевдонима и автозавершение в командной строке для kubectl ... 79
2.3 Запуск первого приложения на Kubernetes ... 80
2.3.1 Развертывание приложения Node.js ... 80
2.3.2 Доступ к веб-приложению ... 83
2.3.3 Логические части вашей системы ... 85
2.3.4 Горизонтальное масштабирование приложения ... 87
2.3.5 Исследование узлов, на которых запущено приложение ... 90
2.3.6 Знакомство с панелью управления Kubernetes ... 91
2.4 Резюме ... 93

Глава 3. Модули: запуск контейнеров в Kubernetes ... 94
3.1 Знакомство с модулями ... 94
3.1.1 Зачем нужны модули ... 95
3.1.2 Общее представление о модулях ... 96
3.1.3 Правильная организация контейнеров между модулями ... 98
3.2 Создание модулей из дескрипторов YAML или JSON ... 100
3.2.1 Исследование дескриптора YAML существующего модуля ... 101
3.2.2 Создание простого дескриптора YAML для модуля ... 103
3.2.3 Использование команды kubectl create для создания модуля ... 105
3.2.4 Просмотр журналов приложений ... 106
3.2.5 Отправка запросов в модуль ... 107
3.3 Организация модулей с помощью меток ... 108
3.3.1 Знакомство с метками ... 108
3.3.2 Указание меток при создании модуля ... 110
3.3.3 Изменение меток существующих модулей ... 111
3.4 Перечисление подмножеств модулей посредством селекторов меток ... 112
3.4.1 Вывод списка модулей с помощью селектора меток ... 112
3.4.2 Использование нескольких условий в селекторе меток ... 113
3.5 Использование меток и селекторов для ограничения планирования модулей ... 114
3.5.1 Использование меток для классификации рабочих узлов ... 115
3.5.2 Приписывание модулей к определенным узлам ... 115

3.5.3 Планирование размещения на один конкретный узел 116
3.6 Аннотирование модулей 116
3.6.1 Поиск аннотаций объекта 117
3.6.2 Добавление и изменение аннотаций 117
3.7 Использование пространств имен для группирования ресурсов 118
3.7.1 Необходимость пространств имен 118
3.7.2 Обнаружение других пространств имен и их модулей 119
3.7.3 Создание пространства имен 120
3.7.4 Управление объектами в других пространствах имен 121
3.7.5 Изоляция, обеспечиваемая пространствами имен 121
3.8 Остановка и удаление модулей 122
3.8.1 Удаление модуля по имени 122
3.8.2 Удаление модулей с помощью селекторов меток 122
3.8.3 Удаление модулей путем удаления всего пространства имен 123
3.8.4 Удаление всех модулей в пространстве имен при сохранении пространства имен 123
3.8.5 Удаление (почти) всех ресурсов в пространстве имен 124
3.9 Резюме 125

Глава 4. Контроллер репликации и другие контроллеры: развертывание управляемых модулей 126
4.1 Поддержание модулей в здоровом состоянии 127
4.1.1 Знакомство с проверками живучести 128
4.1.2 Создание проверки живучести на основе HTTP 128
4.1.3 Просмотр проверки живучести в действии 129
4.1.4 Настройка дополнительных свойств проверки живучести 131
4.1.5 Создание эффективных проверок живучести 132
4.2 Знакомство с контроллерами репликации 134
4.2.1 Работа контроллера репликации 135
4.2.2 Создание контроллера репликации 137
4.2.3 Просмотр контроллера репликации в действии 138
4.2.4 Перемещение модулей в область и из области действия контроллера репликации 142
4.2.5 Изменение шаблона модуля 145
4.2.6 Горизонтальное масштабирование модулей 146
4.2.7 Удаление контроллера репликации 148
4.3 Использование набора реплик вместо контроллера репликации 149
4.3.1 Сравнение набора реплик с контроллером репликации 150
4.3.2 Определение набора реплик 150
4.3.3 Создание и исследование набора реплик 151
4.3.4 Использование более выразительных селекторов меток набора реплик 152
4.3.5 Подведение итогов относительно наборов реплик 153

4.4 Запуск ровно одного модуля на каждом узле с помощью набора демонов (DaemonSet) ... 153
4.4.1 Использование набора демонов для запуска модуля на каждом узле .. 154
4.4.2 Использование набора демонов для запуска модуля только на определенных узлах ... 155
4.5 Запуск модулей, выполняющих одну заканчиваемую задачу ... 158
4.5.1 Знакомство с ресурсом Job ... 158
4.5.2 Определение ресурса Job ... 159
4.5.3 Просмотр того, как задание управляет модулем ... 160
4.5.4 Запуск нескольких экземпляров модуля в задании ... 161
4.5.5 Ограничение времени, отпускаемого на завершение модуля задания .. 163
4.6 Планирование выполнения заданий периодически или один раз в будущем ... 163
4.6.1 Создание ресурса CronJob ... 163
4.6.2 Общие сведения о выполнении запланированных заданий ... 165
4.7 Резюме ... 165

Глава 5. Службы: обеспечение клиентам возможности обнаруживать модули и обмениваться с ними информацией ... 167
5.1 Знакомство со службами ... 168
5.1.1 Создание служб ... 169
5.1.2 Обнаружение служб ... 176
5.2 Подключение к службам, находящимся за пределами кластера ... 180
5.2.1 Знакомство с конечными точками служб ... 180
5.2.2 Настройка конечных точек службы вручную ... 181
5.2.3 Создание псевдонима для внешней службы ... 182
5.3 Предоставление внешним клиентам доступа к службам ... 183
5.3.1 Использование службы NodePort ... 184
5.3.2 Обеспечение доступа к службе через внешнюю подсистему балансировки нагрузки ... 188
5.3.3 Особенности внешних подключений ... 190
5.4 Обеспечение доступа к службам извне через ресурс Ingress ... 192
5.4.1 Создание ресурса Ingress ... 194
5.4.2 Доступ к службе через Ingress ... 195
5.4.3 Обеспечение доступа ко множеству служб через один и тот же Ingress .. 196
5.4.4 Настройка входа для обработки трафика TLS ... 197
5.5 Сигналы о готовности модуля к приему подключений ... 200
5.5.1 Знакомство с проверкой готовности ... 200
5.5.2 Добавление в модуль проверки готовности ... 202
5.5.3 Что должны делать реальные проверки готовности ... 204
5.6 Использование служб без обозначенной точки входа (Headless-сервисов) для обнаружения индивидуальных модулей ... 205
5.6.1 Создание службы без обозначенной точки входа ... 206

5.6.2 Обнаружение модулей через DNS ... 206
5.6.3 Обнаружение всех модулей – даже тех, которые не готовы ... 208
5.7 Устранение неполадок в службах ... 208
5.8 Резюме ... 209

Глава 6. Тома: подключение дискового хранилища к контейнерам ... 211
6.1 Знакомство с томами ... 212
6.1.1 Объяснение томов на примере ... 212
6.1.2 Знакомство с типами томов ... 214
6.2 Использование томов для обмена данными между контейнерами ... 215
6.2.1 Использование тома emptyDir ... 215
6.2.2 Использование репозитория Git в качестве отправной точки для тома ... 219
6.3 Доступ к файлам в файловой системе рабочего узла ... 222
6.3.1 Знакомство с томом hostPath ... 222
6.3.2 Исследование системных модулей с томами hostPath ... 223
6.4 Использование постоянного хранилища ... 224
6.4.1 Использование постоянного диска GCE Persistent Disk в томе модуля ... 225
6.4.2 Использование томов других типов с базовым постоянным хранилищем ... 228
6.5 Отделение модулей от базовой технологии хранения ... 230
6.5.1 Знакомство с томами PersistentVolume и заявками PersistentVolumeClaim ... 231
6.5.2 Создание ресурса PersistentVolume ... 232
6.5.3 Подача заявки на PersistentVolume путем создания ресурса PersistentVolumeClaim ... 233
6.5.4 Использование заявки PersistentVolumeClaim в модуле ... 236
6.5.5 Преимущества использования томов PersistentVolume и заявок ... 237
6.5.6 Повторное использование постоянных томов ... 238
6.6 Динамическое резервирование томов PersistentVolume ... 240
6.6.1 Определение доступных типов хранилища с помощью ресурсов StorageClass ... 240
6.6.2 Запрос на класс хранилища в заявке PersistentVolumeClaim ... 241
6.6.3 Динамическое резервирование без указания класса хранилища ... 243
6.7 Резюме ... 246

Глава 7. Словари конфигурации (ConfigMap) и секреты (Secret): настройка приложений ... 248
7.1 Конфигурирование контейнерных приложений ... 248
7.2 Передача в контейнеры аргументов командной строки ... 250
7.2.1 Определение команды и аргументов в Docker ... 250
7.2.2 Переопределение команды и аргументов в Kubernetes ... 252
7.3 Настройка переменных среды для контейнера ... 254

7.3.1 Указание переменных среды в определении контейнера 255
7.3.2 Ссылка на другие переменные среды в значении переменной 256
7.3.3 Отрицательная сторона жесткого кодирования переменных среды 256
7.4 Отсоединение конфигурации с помощью словаря конфигурации ConfigMap 257
7.4.1 Знакомство со словарями конфигурции 257
7.4.2 Создание словаря конфигурации 258
7.4.3 Передача записи словаря конфигурации в контейнер в качестве переменной среды 260
7.4.4 Одновременная передача всех записей словаря конфигурации как переменных среды 262
7.4.5 Передача записи словаря конфигурации в качестве аргумента командной строки 263
7.4.6 Использование тома configMap для обеспечения доступа к записям словаря конфигурации в виде файлов 264
7.4.7 Обновление конфигурации приложения без перезапуска приложения 271
7.5 Использование секретов для передачи чувствительных данных в контейнеры 274
7.5.1 Знакомство с секретами 274
7.5.2 Знакомство с секретом default-token 275
7.5.3 Создание секрета 277
7.5.4 Сравнение словарей конфигурации и секретов 277
7.5.5 Использование секрета в модуле 279
7.5.6 Секреты для выгрузки образов 284
7.6 Резюме 285

Глава 8. Доступ к метаданным модуля и другим ресурсам из приложений 287
8.1 Передача метаданных через нисходящий API 287
8.1.1 Доступные для использования метаданные 288
8.1.2 Предоставление доступа к метаданным через переменные среды 289
8.1.3 Передача метаданных через файлы в том downwardAPI 292
8.2 Обмен с сервером API Kubernetes 296
8.2.1 Исследование REST API Kubernetes 297
8.2.2 Обмен с сервером API изнутри модуля 302
8.2.3 Упрощение взаимодействия сервера API с контейнерами-посредниками 308
8.2.4 Использование клиентских библиотек для обмена с сервером API 310
8.3 Резюме 314

Глава 9. Развертывания: декларативное обновление приложений 315
9.1 Обновление приложений, работающих в модулях 316
9.1.1 Удаление старых модулей и замена их новыми 317
9.1.2 Запуск новых модулей, а затем удаление старых 317

9.2 Выполнение автоматического плавного обновления с помощью контроллера репликации ... 319
9.2.1 Запуск первоначальной версии приложения ... 319
9.2.2 Выполнение плавного обновления с помощью kubectl ... 321
9.2.3 Почему плавное обновление kubectl rolling-update устарело ... 326
9.3 Использование развертываний для декларативного обновления приложений ... 327
9.3.1 Создание развертывания ... 328
9.3.2 Обновление с помощью развертывания ... 331
9.3.3 Откат развертывания ... 335
9.3.4 Управление скоростью выкладки ... 338
9.3.5 Приостановка процесса выкладки ... 341
9.3.6 Блокировка раскруток плохих версий ... 342
9.4 Резюме ... 348

Глава 10. Ресурсы StatefulSet: развертывание реплицируемых приложений с внутренним состоянием ... 349
10.1 Репликация модулей с внутренним состоянием ... 349
10.1.1 Запуск множества реплик с отдельным хранилищем для каждой ... 350
10.1.2 Обеспечение стабильной долговременной идентификации для каждого модуля ... 352
10.2 Набор модулей с внутренним состоянием ... 353
10.2.1 Сопоставление наборов модулей с внутренним состоянием и наборов реплик ... 353
10.2.2 Обеспечение стабильной сетевой идентичности ... 355
10.2.3 Обеспечение стабильного выделенного хранилища для каждого экземпляра с внутренним состоянием ... 357
10.2.4 Гарантии набора StatefulSet ... 360
10.3 Использование набора StatefulSet ... 360
10.3.1 Создание приложения и образа контейнера ... 361
10.3.2 Развертывание приложения посредством набора StatefulSet ... 362
10.3.3 Исследование своих модулей ... 367
10.4 Обнаружение соседей в наборе StatefulSet ... 371
10.4.1 Реализация обнаружения соседей посредством DNS ... 373
10.4.2 Обновление набора StatefulSet ... 375
10.4.3 Опробование кластеризованного хранилища данных ... 376
10.5 Как наборы StatefulSet справляются с аварийными сбоями узлов ... 377
10.5.1 Симулирование отключения узла от сети ... 377
10.5.2 Удаление модуля вручную ... 379
10.6 Резюме ... 381

Глава 11. Внутреннее устройство Kubernetes ... 382
11.1 Архитектура ... 382

11.1.1 Распределенная природа компонентов Kubernetes ... 383
11.1.2 Как Kubernetes использует хранилище etcd ... 386
11.1.3 Что делает сервер API ... 390
11.1.4 Как сервер API уведомляет клиентов об изменениях ресурсов ... 392
11.1.5 Планировщик ... 393
11.1.6 Знакомство с контроллерами, работающими в менеджере контроллеров ... 396
11.1.7 Что делает агент Kubelet ... 401
11.1.8 Роль служебного сетевого прокси системы Kubernetes ... 403
11.1.9 Знакомство с надстройками Kubernetes ... 404
11.1.10 Все воедино ... 406
11.2 Взаимодействие контроллеров ... 406
11.2.1 Какие компоненты задействованы ... 406
11.2.2 Цепь событий ... 406
11.2.3 Наблюдение за событиями кластера ... 408
11.3 Что такое запущенный модуль ... 410
11.4 Интермодульное сетевое взаимодействие ... 411
11.4.1 Как должна выглядеть сеть ... 411
11.4.2 Более детальное рассмотрение работы сетевого взаимодействия ... 413
11.4.3 Знакомство с контейнерным сетевым интерфейсом ... 415
11.5 Как реализованы службы ... 416
11.5.1 Введение в kube-proxy ... 416
11.5.2 Как kube-proxy использует правила iptables ... 416
11.6 Запуск высокодоступных кластеров ... 418
11.6.1 Обеспечение высокой доступности приложений ... 418
11.6.2 Обеспечение высокой доступности компонентов плоскости управления Kubernetes ... 419
11.7 Резюме ... 423

Глава 12. Защита сервера API Kubernetes ... 424
12.1 Аутентификация ... 424
12.1.1 Пользователи и группы ... 425
12.1.2 Знакомство с учетными записями службы ... 426
12.1.3 Создание учетных записей ServiceAccount ... 427
12.1.4 Назначение модулю учетной записи службы ... 430
12.2 Защита кластера с помощью управления ролевым доступом ... 432
12.2.1 Знакомство с плагином авторизации RBAC ... 432
12.2.2 Знакомство с ресурсами RBAC ... 434
12.2.3 Использование ролей и привязок ролей ... 437
12.2.4 Применение кластерных ролей (ClusterRole) и привязок кластерных ролей (ClusterRoleBinding) ... 441
12.2.5 Кластерные роли и привязки кластерных ролей, существующие по умолчанию ... 452
12.2.6 Предоставление разумных авторизационных разрешений ... 455

12.3 Резюме 456

Глава 13. Защита узлов кластера и сети 457

13.1 Использование в модуле пространств имен хоста 457

13.1.1 Использование в модуле сетевого пространства имен узла 458

13.1.2 Привязка к порту хоста без использования сетевого пространства имен хоста 459

13.1.3 Использование пространств имен PID и IPC узла 461

13.2 Конфигурирование контекста безопасности контейнера 462

13.2.1 Выполнение контейнера от имени конкретного пользователя 464

13.2.2 Недопущение работы контейнера в качестве root 464

13.2.3 Выполнение модулей в привилегированном режиме 465

13.2.4 Добавление отдельных функциональных возможностей ядра в контейнер 467

13.2.5 Удаление функциональных возможностей из контейнера 469

13.2.6 Недопущение записи процессами в файловую систему контейнера 470

13.2.7 Совместное использование томов, когда контейнеры запущены под разными пользователями 471

13.3 Ограничение использования функциональности, связанной с безопасностью в модулях 474

13.3.1 Знакомство с ресурсами PodSecurityPolicy 474

13.3.2 Политики runAsUser, fsGroup и supplementalGroups 477

13.3.3 Конфигурирование разрешенных, стандартных и запрещенных возможностей 479

13.3.4 Ограничение типов томов, которые модули могут использовать 481

13.3.5 Назначение разных политик PodSecurityPolicy разным пользователям и группам 481

13.4 Изоляция сети модулей 485

13.4.1 Активация изоляции сети в пространстве имен 486

13.4.2 Разрешение подключения к серверному модулю только некоторых модулей в пространстве имен 486

13.4.3 Изоляция сети между пространствами имен Kubernetes 487

13.4.4 Изоляция с использованием обозначения CIDR 488

13.4.5 Лимитирование исходящего трафика набора модулей 489

13.5 Резюме 490

Глава 14. Управление вычислительными ресурсами модулей 491

14.1 Запрос на ресурсы для контейнеров модуля 491

14.1.1 Создание модулей с ресурсными запросами 492

14.1.2 Как ресурсные запросы влияют на назначение модуля узлу 493

14.1.3 Как запросы на ЦП влияют на совместное использование процессорного времени 498

14.1.4 Определение и запрос настраиваемых ресурсов 499

14.2 Лимитирование ресурсов, доступных контейнеру 500

14.2.1 Установка жесткого лимита на объем ресурсов, которые может использовать контейнер ... 500
14.2.2 Превышение лимитов ... 502
14.2.3 Как приложения в контейнерах видят лимиты ... 503
14.3 Классы QoS модулей ... 505
14.3.1 Определение класса QoS для модуля ... 505
14.3.2 Какой процесс уничтожается при нехватке памяти ... 508
14.4 Установка стандартных запросов и лимитов для модулей в расчете на пространство имен ... 510
14.4.1 Знакомство с ресурсом LimitRange ... 510
14.4.2 Создание объекта LimitRange ... 511
14.4.3 Обеспечение лимитов ... 513
14.4.4 Применение стандартных ресурсных запросов и лимитов ... 513
14.5 Лимитирование общего объема ресурсов, доступных в пространстве имен ... 514
14.5.1 Объект ResourceQuota ... 515
14.5.2 Указание квоты для постоянного хранилища ... 517
14.5.3 Лимитирование количества создаваемых объектов ... 517
14.5.4 Указание квот для конкретных состояний модулей и/или классов QoS ... 519
14.6 Мониторинг потребления ресурсов модуля ... 520
14.6.1 Сбор и извлечение фактических данных потребления ресурсов ... 520
14.6.2 Хранение и анализ исторической статистики потребления ресурсов ... 523
14.7 Резюме ... 526

Глава 15. Автоматическое масштабирование модулей и узлов кластера ... 528
15.1 Горизонтальное автомасштабирование модуля ... 529
15.1.1 Процесс автомасштабирования ... 529
15.1.2 Масштабирование на основе задействованности ЦП ... 533
15.1.3 Масштабирование на основе потребления памяти ... 540
15.1.4 Масштабирование на основе других, а также настраиваемых метрик ... 541
15.1.5 Определение метрик, подходящих для автомасштабирования ... 544
15.1.6 Уменьшение масштаба до нуля реплик ... 544
15.2 Вертикальное автомасштабирование модуля ... 545
15.2.1 Автоматическое конфигурирование ресурсных запросов ... 545
15.2.2 Модификация ресурсных запросов во время работы модуля ... 545
15.3 Горизонтальное масштабирование узлов кластера ... 546
15.3.1 Знакомство с кластерным автопреобразователем масштаба ... 546
15.3.2 Активация кластерного автопреобразователя масштаба ... 548
15.3.3 Ограничение прерывания службы во время уменьшения

масштаба кластера ... 549
15.4 Резюме ... 550

Глава 16. Продвинутое назначение модулей узлам ... 552
16.1 Использование ограничений и допусков для отделения модулей от определенных узлов ... 552
16.1.1 Знакомство с ограничениями и допусками ... 553
16.1.2 Добавление в узел индивидуально настроенных ограничений ... 555
16.1.3 Добавление в модули допусков ... 556
16.1.4 Для чего можно использовать ограничения и допуски ... 557
16.2 Использование сходства узлов для привлечения модулей к определенным узлам ... 558
16.2.1 Указание жестких правил сходства узлов ... 559
16.2.2 Приоретизация узлов при назначении модуля ... 561
16.3 Совместное размещение модулей с использованием сходства и антисходства модулей ... 565
16.3.1 Использование межмодульного сходства для развертывания модулей на одном узле ... 565
16.3.2 Развертывание модулей в одной стойке, зоне доступности или географическом регионе ... 568
16.3.3 Выражение предпочтений сходства модулей вместо жестких требований ... 569
16.3.4 Назначение модулей на удалении друг от друга с помощью антисходства модулей ... 571
16.4 Резюме ... 573

Глава 17. Рекомендации по разработке приложений ... 574
17.1 Соединение всего вместе ... 574
17.2 Жизненный цикл модуля ... 576
17.2.1 Приложения должны ожидать, что они могут быть удалены и перемещены ... 576
17.2.2 Переназначение мертвых или частично мертвых модулей ... 579
17.2.3 Запуск модулей в определенном порядке ... 581
17.2.4 Добавление обработчиков жизненного цикла ... 583
17.2.5 Выключение модуля ... 588
17.3 Обеспечение правильной обработки всех клиентских запросов ... 591
17.3.1 Предотвращение прерывания клиентских подключений при запуске модуля ... 592
17.3.2 Предотвращение прерванных подключений при выключении модуля ... 592
17.4 Упрощение запуска приложений и управления ими в Kubernetes ... 597
17.4.1 Создание управляемых образов контейнеров ... 597
17.4.2 Правильное тегирование образов и рациональное использование политики imagePullPolicy ... 598

17.4.3 Использование многомерных меток вместо одномерных ... 598
17.4.4 Описание каждого ресурса с помощью аннотаций ... 599
17.4.5 Предоставление информации о причинах прекращения процесса ... 599
17.4.6 Работа с журналами приложений ... 601
17.5 Рекомендации по разработке и тестированию ... 603
17.5.1 Запуск приложений за пределами Kubernetes во время разработки ... 603
17.5.2 Использование Minikube в разработке ... 605
17.5.3 Версионирование и автоматическое развертывание ресурсных манифестов ... 606
17.5.4 Знакомство с Ksonnet как альтернативой написанию манифестов YAML/JSON ... 607
17.5.5 Использование непрерывной интеграции и непрерывной доставки (CI/CD) ... 608
17.6 Резюме ... 608

Глава 18. Расширение системы Kubernetes ... 610
18.1 Определение своих собственных объектов API ... 610
18.1.1 Знакомство с определениями CustomResourceDefinition ... 611
18.1.2 Автоматизация пользовательских ресурсов с помощью пользовательских контроллеров ... 615
18.1.3 Валидация пользовательских объектов ... 620
18.1.4 Предоставление пользовательского сервера API для пользовательских объектов ... 621
18.2 Расширение Kubernetes с помощью каталога служб Kubernetes (Kubernetes Service Catalog) ... 623
18.2.1 Знакомство с каталогом служб ... 623
18.2.2 Знакомство с сервером API каталога служб и менеджером контроллеров ... 625
18.2.3 Знакомство с брокерами служб и API OpenServiceBroker ... 625
18.2.4 Резервирование и использование службы ... 627
18.2.5 Отвязывание и дерезервирование ... 630
18.2.6 Что дает каталог служб ... 631
18.3 Платформы, построенные поверх Kubernetes ... 631
18.3.1 Контейнерная платформа Red Hat OpenShift ... 631
18.3.2 Deis Workflow и Helm ... 635
18.4 Резюме ... 638

Приложение A. Использование kubectl с несколькими кластерами ... 639
A.1 Переключение между Minikube и Google Kubernetes Engine ... 639
A.2 Использование инструмента kubectl со множеством кластеров или пространств имен ... 640
A.2.1 Настройка расположения файла kubeconfig ... 640
A.2.2 Содержимое файла kubeconfig ... 640
A.2.3 Вывод списка, добавление и изменение записей в файле kubeconfig ... 642

A.2.4 Использование инструмента kubectl с разными кластерами, пользователями и контекстами ... 643
A.2.5 Переключение между контекстами ... 644
A.2.6 Перечисление контекстов и кластеров ... 644
A.2.7 Удаление контекстов и кластеров ... 644

Приложение B. Настройка многоузлового кластера с помощью kubeadm ... 645
B.1 Настройка ОС и необходимых пакетов ... 645
B.1.1 Создание виртуальной машины ... 645
B.1.2 Настройка сетевого адаптера для виртуальной машины ... 646
B.1.3 Инсталляция операционной системы ... 647
B.1.4 Установка Docker и Kubernetes ... 650
B.1.5 Клонирование виртуальной машины ... 652
B.2 Конфигурирование ведущего узла с помощью kubeadm ... 654
B.2.1 Как kubeadm запускает компоненты ... 655
B.3 Настройка рабочих узлов с помощью kubeadm ... 656
B.3.1 Настройка контейнерной сети ... 657
B.4 Использование кластера с локальной машины ... 658

Приложение C. Использование других контейнерных сред выполнения ... 659
C.1 Замена Docker на rkt ... 659
C.1.1 Настройка Kubernetes для использования rkt ... 659
C.1.2 Опробирование платформы rkt с Minikube ... 660
C.2 Использование других контейнерных сред выполнения посредством CRI ... 662
C.2.1 Знакомство с контейнерной средой выполнения CRI-O ... 663
C.2.2 Запуск приложений на виртуальных машинах вместо контейнеров ... 663

Приложение D. Кластерная федерация ... 664
D.1 Знакомство с кластерной федерацией Kubernetes ... 664
D.2 Архитектура ... 665
D.3 Федеративные объекты API ... 665
D.3.1 Знакомство с федеративными версиями ресурсов Kubernetes ... 665
D.3.2 Что делают федеративные ресурсы ... 666

Предметный указатель ... 669

Вступительное слово от компании ITSumma

Любимой кофейной кружке придется подвинуться – эта книга займет центральное место на вашем столе и станет главным навигатором в контейнерных технологиях. Мы очень рады, что книга выходит на русском языке. После своего триумфального появления в 2014 году Kubernetes основательно утвердился в качестве лидирующей программы для оркестровки контейнеров в мире. Но в России до сих пор не было ресурса, который дал бы исчерпывающую информацию о программе.

Данная книга дает пошаговое разъяснение принципов работы контейнеризации и устройства модулей в Kubernetes. Вы узнаете все о создании объектов верхнего уровня, развертывании кластера Kubernetes на собственной рабочей машине и построении федеративного кластера в нескольких дата-центрах.

Автор книги – инженер-программист с двадцатилетним стажем Марко Лукша. Последние пять лет он курировал новые разработки Kubernetes для компании Red Hat и изучил все тонкости создания приложений в данной среде. Сложные рабочие схемы Лукша преподносит в очень занимательной и последовательной манере. Вы узнаете о трех способах обеспечить доступ к службам извне и разберете варианты перехода к процессу обновления с нулевым временем простоя. А также сравните размещение в Kubernetes классических контейнеризированных микросервисов и систем, использующих внутреннее состояние.

Нас приятно удивило, что Лукша уделяет особое внимание извечному спору сисадминов и разработчиков и в том числе рассказывает о практике NoOps, которая разрешает конфликты двух лагерей. Отдельное спасибо автору за то, что он детально анализирует задачи обеспечения безопасности в Kubernetes – крайне актуальную и малоисследованную на сегодня задачу.

Наглядные схемы и «разбор полетов» в книге создают впечатление индивидуального мастер-класса. Автор предвосхищает большинство вопросов, заранее объясняя, в каком направлении двигаться, если что-то пойдет не так. Те, кто только начал знакомство с платформой, смогут уверенно перейти к практической работе в Kubernetes. А «бывалые» специалисты узнают причины и пути решения для ранее безвыходных ситуаций.

Эта книга – отличный компас в увлекательном изучении облачных технологий, которые с каждым годом набирают все больше последователей. Надеемся увидеть вас в их числе.

Ваша ITSumma

Описанные/рассмотренные в книге ресурсы Kubernetes

	Ресурс (сокр.) [версия API]	Описание	Раздел
	Namespace* (ns) [v1]	Позволяет организовывать ресурсы в неперекрывающиеся группы (например, для каждого потребителя ресурсов)	3.7
Развертывающие рабочие нагрузки	**Pod** (po) [v1]	Основная развертываемая единица, содержащая один или более процессов в расположенных рядом контейнерах	3.1
	ReplicaSet (rs) [apps/v1beta2**]	Поддерживает одну или несколько реплик модуля	4.3
	ReplicationController (rc) [v1]	Более старый, менее мощный эквивалент ресурса ReplicaSet	4.2
	Job [batch/v1]	Запускает модули, выполняющие завершаемую задачу	4.5
	CronJob [batch/v1beta1]	Запускает назначаемое задание один раз или периодически	4.6
	DaemonSet (ds) [apps/v1beta2**]	Запускает одну реплику модуля в расчете на узел (на всех узлах или только на тех, которые соответствуют селектору узлов)	4.4
	StatefulSet (sts) [apps/v1beta1**]	Запускает модули, имеющие внутреннее состояние, со стабильной идентичностью	10.2
	Deployment (deploy) [apps/v1beta1**]	Декларативное развертывание и обновление модулей	9.3
Службы	**Service** (svc) [v1]	Предоставляет доступ к одному или нескольким модулям на одной и стабильной паре IP-адреса и порта	5.1
	Endpoints (ep) [v1]	Определяет, к каким модулям (или другим серверам) предоставляется доступ через службу	5.2.1
	Ingress (ing) [extensions/v1beta1]	Предоставляет внешним клиентам доступ к одной или нескольким службам через один доступный извне IP-адрес	5.4
Конфигурация	**ConfigMap** (cm) [v1]	Словарь в формате «ключ-значение» для хранения незащищенных параметров конфигурации приложений и предоставления им доступа к ним	7.4
	Secret [v1]	Как и словарь конфигурации ConfigMap, но для конфиденциальных данных	7.5
Хранение	**PersistentVolume*** (pv) [v1]	Указывает на постоянное хранилище, которое можно смонтировать в модуль посредством заявки PersistentVolumeClaim	6.5
	PersistentVolumeClaim (pvc) [v1]	Запрос и заявка на PersistentVolume	6.5
	StorageClass* (sc) [storage.k8s.io/v1]	Определяет тип динамически резервируемого хранилища, заявляемого в PersistentVolumeClaim	6.6

(Окончание)

	Ресурс (сокр.) [версия API]	Описание	Раздел
Масштабирование	**HorizontalPodAutoscaler** (hpa) [autoscaling/v2beta1**]	Автоматически масштабирует количество реплик модулей на основе использования ЦП или другой метрики	15.1
	PodDisruptionBudget (pdb) [policy/v1beta1]	Определяет минимальное количество модулей, которые должны оставаться запущенными при эвакуации узлов	15.3.3
Ресурсы	**LimitRange** (limits) [v1]	Определяет мин, макс, ограничения и запросы по умолчанию (default) для модулей в пространстве имен	14.4
	ResourceQuota (quota) [v1]	Определяет объем вычислительных ресурсов, доступных для модулей в пространстве имен	14.5
Состояние кластера	**Node*** (нет)[v1]	Представляет рабочий узел Kubernetes	2.2.2
	Cluster* [federation/v1beta1]	Кластер Kubernetes (используемый в федерации кластеров)	Прил. D
	ComponentStatus* (cs) [v1]	Статус компонента контрольной панели	11.1.1
	Event (ev) [v1]	Отчет о том, что произошло в кластере	11.2.3
Безопасность	**ServiceAccount** (sa) [v1]	Учетная запись, используемая приложениями, запущенными в модулях	12.1.2
	Role [rbac.authorization.k8s.io/v1]	Определяет, какие действия субъект может выполнять с какими ресурсами (в расчете на пространство имен)	12.2.3
	ClusterRole* [rbac.authorization.k8s.io/v1]	Как Role, но для ресурсов уровня кластера или для предоставления доступа к ресурсам во всех пространствах имен	12.2.4
	RoleBinding [rbac.authorization.k8s.io/v1]	Определяет, кто может выполнять действия, определенные в Role или ClusterRole (в пространстве имен)	12.2.3
	ClusterRoleBinding* [rbac.authorization.k8s.io/v1]	Как RoleBinding, но по всем пространствам имен	12.2.4
	PodSecurityPolicy* (psp) [extensions/v1beta1]	Ресурс уровня кластера, который определяет, какие чувствительные для безопасности особенности могут использовать модули	13.3.1
	NetworkPolicy (netpol) [networking.k8s.io/v1]	Изолирует сеть между модулями, указывая, какие модули могут подключаться друг к другу	13.4
	CertificateSigningRequest* (csr) [certificates.k8s.io/v1beta1]	Запрос на подписание сертификата открытого ключа	5.4.4
Расш.	**CustomResourceDefinition*** (crd) [apiextensions.k8s.io/v1beta1]	Определяет настраиваемый ресурс, позволяющий пользователям создавать экземпляры настраиваемого ресурса	18.1

* Ресурс уровня кластера (без пространства имен).

** Также в других версиях API; указанная версия использована в этой книге.

Благодарности

Издательство «ДМК Пресс» благодарит компанию «ITSumma» и ее генерального директора Евгения Потапова за большую неоценимую помощь в подготовке этой книги.

Без них вы, возможно, не так хорошо поняли Kubernetes, но теперь мы полностью уверены, что все термины стоят на своих местах.

Предисловие

После работы в Red Hat в течение нескольких лет, в конце 2014 года я был назначен в недавно созданную команду под названием Cloud Enablement. Нашей задачей было вывести линейку продуктов компании промежуточного уровня на OpenShift Container Platform, которая затем разрабатывалась поверх платформы Kubernetes. В то время Kubernetes все еще находилась в зачаточном состоянии – версия 1.0 еще даже не была выпущена.

Наша команда должна была узнать все входы и выходы Kubernetes, чтобы быстро установить правильное направление для нашего программного обеспечения и воспользоваться всем, что система Kubernetes должна была предложить. Когда мы сталкивались с проблемой, нам было трудно сказать, делаем ли мы что-то неправильно или просто попали на одну из ранних ошибок Kubernetes.

С тех пор и Kubernetes, и мое понимание этой платформы прошли долгий путь. Когда я впервые начал ее использовать, большинство людей даже не слышало о Kubernetes. Теперь об этой платформе знает практически каждый инженер-программист, и она стала одним из самых быстрорастущих и широко распространенных способов запуска приложений как в облаке, так и в локальных центрах обработки данных.

В первый месяц работы с Kubernetes я написал двухсоставной блог-пост о том, как запускать кластер сервера приложений Jboss WildFly в OpenShift/Kubernetes. В то время я совсем не мог себе представить, что простой пост в блоге в конечном итоге приведет к тому, что представители издательства Manning свяжутся со мной по поводу того, хочу ли я написать книгу о Kubernetes. Конечно, я не мог сказать «нет» такому предложению, хотя я был уверен, что они также обратились к другим специалистам и в конечном итоге выбрали кого-то другого.

И все же вот мы здесь. После более чем полутора лет написания и исследования книга закончена. Это было потрясающее путешествие. Написание книги о технологии является идеальным способом узнать ее гораздо подробнее, чем если вы узнаете ее как простой пользователь. По мере того как во время процесса написания книги мои знания о Kubernetes расширялись, а сама платформа Kubernetes эволюционировала, я постоянно возвращался к предыдущим, уже написанным главам и добавлял дополнительную информацию. Я перфекционист, поэтому никогда не буду абсолютно доволен книгой, но я рад узнать, что многие читатели программы раннего доступа Manning (MEAP) считают ее отличным путеводителем по Kubernetes.

Моя цель состоит в том, чтобы помочь читателю понять саму технологию и научить их использовать ее инструментарий для эффективной и действенной разработки и развертывания приложений в кластерах Kubernetes. В книге я не уделяю особого внимания тому, как на самом деле создавать и поддерживать

надлежащий высокодоступный кластер Kubernetes, но последняя часть должна дать читателям очень четкое понимание того, из чего состоит такой кластер, и должна позволить им легко разбираться в дополнительных ресурсах, которые касаются этой темы.

Очень надеюсь, что вам понравится читать эту книгу и что она научит вас получать максимальную отдачу от удивительной системы под названием Kubernetes.

Признательности

До того, как я начал писать эту книгу, я понятия не имел, сколько людей будет вовлечено в ее перенос из грубой рукописи в опубликованную работу. А это означает, что я просто обязан поблагодарить очень многих людей.

Во-первых, я хотел бы поблагодарить Эрин Твохи за то, что она обратилась ко мне с просьбой написать эту книгу, и Майкла Стивенса из Мэннинга, который с первого дня был полностью уверен в моей способности написать ее. Его слова ободрения на раннем этапе действительно мотивировали меня и поддерживали меня в течение последних полутора лет.

Также хотел бы поблагодарить моего технического редактора Эндрю Уоррена, который помог мне выпустить мою первую главу, и Элешу Хайд, которая затем сменила Эндрю и работала со мной вплоть до последней главы. Спасибо за то, что вы со мной, хотя со мной трудно иметь дело, поскольку я, как правило, довольно регулярно выпадаю с радаров.

Кроме того, хотел бы поблагодарить Жанну Боярскую, ставшую первым рецензентом, которая читала и комментировала мои главы, пока я их писал. Жанна и Элеша сыграли важную роль в том, чтобы эта книга, как мы надеемся, стала недурной. Без их комментариев книга никогда бы не получила таких хороших отзывов от внешних рецензентов и читателей.

Хотел бы поблагодарить моего технического корректора, Антонио Мадгаги, и, конечно, всех моих внешних рецензентов: Эла Кринкера, Алессандро Кампэйса, Александра Мыльцевы, Чабу Сари, Дэвида Димариа, Элиаса Ранхеля, Эриса Зеленка, Фабрицио Куччи, Джареда Данкана, Кэйт Дональдсон, Майкла Брайта, Паоло Антинори, Питера Перлепеса и Тиклу Гангули. Их положительные комментарии меня поддерживали время от времени, когда я беспокоился, что мой стиль был совершенно ужасным и бесполезным. С другой стороны, их конструктивная критика помогла улучшить разделы, которые я быстро собрал без достаточных усилий. Спасибо, что указали на трудные для понимания разделы и предложили способы улучшения книги. Кроме того, спасибо за правильные вопросы, которые заставили меня понять свои ошибки в двух или трех вещах в первоначальных версиях рукописи.

Мне также нужно поблагодарить читателей, которые купили раннюю версию книги через программу MEAP издательства Мэннинга и озвучили свои комментарии на онлайн-форуме или обратились ко мне напрямую, особенно Вимала Кансала, Паоло Патьерно и Роланда Хаба, которые заметили немало несоответствий и других ошибок. И я хотел бы поблагодарить всех в издательстве Manning, кто участвовал в издании этой книги. Прежде чем я закончу, мне также нужно поблагодарить моего коллегу и друга школьных лет Эла Джастина, который привел меня в Red Hat, и моих замечательных коллег из команды Cloud Enablement. Если бы я не был в Red Hat или в этой команде, я бы не написал эту книгу.

Наконец, я хотел бы поблагодарить мою жену и моего сына, которые относились ко мне с пониманием и поддерживали меня в течение последних 18 месяцев, когда я запирался в своем офисе, вместо того чтобы проводить время с ними.

Спасибо всем!

Об этой книге

Основная задача книги «*Kubernetes в действии*» состоит в том, чтобы сделать вас опытным пользователем Kubernetes. Он научит вас практически всем понятиям, в которых вам нужно разбираться, чтобы эффективно разрабатывать и запускать приложения в среде Kubernetes.

Перед погружением в Kubernetes данная книга дает обзор контейнерных технологий, таких как Docker, в том числе того, как строить контейнеры, так что даже читатели, которые не использовали эти технологии раньше, могут с нуля организовать и вести свою работу. Затем она неспешно проведет вас через значительную часть того, что вам нужно знать о Kubernetes, – от элементарных понятий до вещей, скрытых под поверхностью.

Кто должен читать эту книгу

Данная книга посвящена в первую очередь разработчикам приложений, но также содержит обзор управляющих приложений с точки зрения системного администрирования. Она предназначена для тех, кто заинтересован в запуске и управлении контейнеризированными приложениями на более чем одном сервере.

Как начинающие, так и продвинутые инженеры-программисты, которые хотят познакомиться с контейнерными технологиями и организовать несколько связанных контейнеров в масштабе, получат опыт, необходимый для разработки, контейнеризации и запуска своих приложений в среде Kubernetes.

Предыдущий опыт работы с контейнерными технологиями либо Kubernetes вовсе не обязателен. Книга объясняет данный предмет, постепенно углубляясь в подробности, и не использует никакого исходного кода приложений, который было бы слишком трудно понять разработчикам, не являющимся экспертами.

Вместе с тем читатели должны иметь, по крайней мере, базовые знания в области программирования, компьютерных сетей и выполнения основных команд в Linux, а также понимание общеизвестных компьютерных протоколов, таких как HTTP.

Как организована эта книга: дорожная карта

Эта книга состоит из трех частей, которые включают в себя 18 глав.

Часть 1 дает краткое введение в Docker и Kubernetes, как настроить кластер Kubernetes и как запустить в нем простое приложение. Она содержит две главы:

- глава 1 объясняет, что такое Kubernetes, как эта система появилась и как она помогает решать сегодняшние задачи управления приложениями в масштабе;

- глава 2 представляет собой практическое руководство по созданию образа контейнера и по его запуску в кластере Kubernetes. В ней также объясняется, как запускать локальный одноузловой кластер Kubernetes и надлежащий многоузловой кластер в облаке.

В части 2 представлены ключевые понятия, в которых необходимо разбираться для запуска приложений в Kubernetes. Разделы этой части следующие:

- глава 3 вводит фундаментальный строительный блок в Kubernetes – модуль (Pod) и объясняет, как организовать модули и другие объекты Kubernetes посредством меток (Label);
- глава 4 знакомит вас с тем, как Kubernetes поддерживает приложения в рабочем состоянии путем автоматического перезапуска контейнеров. В ней также показывается, как правильно запускать управляемые модули, горизонтально масштабировать их, делать их устойчивыми к сбоям узлов кластера и запускать их в заданное время в будущем или периодически;
- в главе 5 показано, как модули могут обеспечивать доступ к службе (Service), которую они предоставляют клиентам, работающим как внутри, так и за пределами кластера. В ней также показано, как модули, работающие в кластере, могут обнаруживать службы и получать к ним доступ независимо от того, находятся они в кластере или нет;
- в главе 6 объясняется, как несколько контейнеров, работающих в одном модуле, могут совместно использовать файлы и как управлять постоянным хранилищем и делать его доступным для модулей;
- в главе 7 показано, как передавать конфигурационные данные и конфиденциальную информацию, такую как учетные данные, приложениям, работающим внутри модулей;
- глава 8 описывает, как приложения могут получать информацию о среде Kubernetes, в которой они работают, и как они могут общаться с Kubernetes для изменения состояния кластера;
- глава 9 знакомит с концепцией развертывания (Deployment) и объясняет, как правильно запускать и обновлять приложения в среде Kubernetes;
- в главе 10 представлен специальный способ запуска приложений с сохранением состояния, которые обычно требуют конкретной идентификации и хранения данных о собственном состоянии.

Часть 3 глубоко погружается во внутренние части кластера Kubernetes, вводит некоторые дополнительные понятия и рассматривает все, что вы узнали в первых двух частях, с более высокой точки зрения. Это последняя группа глав:

- глава 11 опускается под поверхность Kubernetes и объясняет все компоненты, которые составляют кластер Kubernetes, и что каждый из них делает. В ней также объясняется, как модули обмениваются по сети и как службы выполняют балансировку нагрузки между несколькими модулями;

- в главе 12 объясняется, как защитить API Kubernetes и в более общем смысле кластер с помощью аутентификации и авторизации;
- в главе 13 рассказывается, как модули могут получать доступ к ресурсам узла и как администратор кластера может это предотвратить;
- глава 14 посвящена ограничению вычислительных ресурсов, потребляемых каждым приложением, настройке гарантий качества обслуживания приложений и мониторингу использования ресурсов отдельными приложениями. Она также знакомит вас с тем, как препятствовать тому, чтобы пользователи потребляли слишком много ресурсов;
- в главе 15 рассказывается, как настроить Kubernetes на автоматическое масштабирование числа запущенных реплик приложения, а также как увеличивать размер кластера, если текущее число узлов кластера не может принимать дополнительные приложения;
- в главе 16 показано, как обеспечить назначение модулей только определенным узлам или как воспрепятствовать их назначению другим узлам. Она также показывает, как убедиться, что модули назначены вместе, или как не допускать этого;
- глава 17 знакомит вас с тем, как нужно разрабатывать свои приложения, чтобы сделать их добропорядочными гражданами вашего кластера. В ней также дается несколько советов о том, как настроить рабочие процессы разработки и тестирования для уменьшения накладных расходов;
- в главе 18 показано, как можно расширять Kubernetes с помощью собственных настраиваемых объектов и как это сделали другие и создали платформы приложений корпоративного уровня.

По мере прохождения этих глав вы не только узнаете об отдельных строительных блоках Kubernetes, но и постепенно улучшите свои знания об использовании инструмента командной строки `kubectl`.

О коде

Хотя эта книга не содержит массу фактического исходного кода, в ней действительно много манифестов ресурсов Kubernetes в формате YAML и команд оболочки вместе с их результатами. Все это отформатировано `вот таким моноширинным шрифтом`, чтобы отделить его от обычного текста.

Команды оболочки в основном выделены **жирным шрифтом**, чтобы четко отделить их от результатов их выполнения, но иногда жирным шрифтом для акцента выделены только наиболее важные части команды или части результатов команды. В большинстве случаев результаты команды были переформатированы, чтобы уместить их в ограниченное пространство книги. Кроме того, поскольку инструмент командного интерфейса `kubectl` системы Kubernetes постоянно развивается, новые версии могут печатать больше информации, чем показано в книге. Не смущайтесь, если они не совпадают в точности.

Листинги иногда включают маркер продолжения строки (➥), чтобы показать, что строка текста переносится на следующую строку. Они также включают аннотации, которые выделяют и объясняют наиболее важные части.

В текстовых абзацах некоторые очень распространенные элементы, такие как Pod, ReplicationController, ReplicaSet, DaemonSet и т. д., задаются обычным шрифтом, чтобы избежать чрезмерного распространения шрифта кода и способствовать читаемости. В некоторых местах слово «Pod» пишется в верхнем регистре для ссылки на ресурс Pod, а в нижнем регистре для ссылки на фактический модуль, то есть группу запущенных контейнеров.

Все образцы в книге были протестированы с помощью Kubernetes версии 1.8, работающей в Google Kubernetes Engine и в локальном кластере под управлением Minikube. Полный исходный код и YAML манифестов можно найти на https://github.com/luksa/kubernetes-in-action либо скачать с веб-сайта издателя в www.manning.com/books/kubernetes-in-action.

Книжный форум

Покупка книги «*Kubernetes в действии*» включает в себя бесплатный доступ к частному веб-форуму под управлением Manning Publications, где вы можете комментировать книгу, задавать технические вопросы и получать помощь от автора и других пользователей. Для того чтобы получить доступ к форуму, перейдите по ссылке https://forums.manning.com/forums/kubernetes-in-action. Вы также можете узнать больше о форумах издательства Manning и правилах поведения на https://forums.manning.com/forums/about.

Обязательство издательства Manning перед нашими читателями заключается в том, чтобы обеспечить место, где может иметь место содержательный диалог между отдельными читателями и между читателями и автором. Это не является обязательством какого-либо конкретного участия со стороны автора, чей вклад в форум остается добровольным (и неоплачиваемым). Мы предлагаем вам попробовать задать автору несколько сложных вопросов, чтобы его интерес не пропал! Форум и архив предыдущих обсуждений будут доступны с веб-сайта издателя до тех пор, пока книга находится в печати.

Другие интернет-ресурсы

Вы можете найти широкий спектр дополнительных ресурсов Kubernetes в следующих местах:

- сайт Kubernetes на https://kubernetes.io;
- блог Kubernetes, в котором регулярно публикуется интересная информация (http://blog.kuber-netes.io);
- канал Slack сообщества Kubernetes на http://slack.k8s.io;
- каналы YouTube системы Kubernetes и фонда Cloud Native Computing Foundation:

- https://www.youtube.com/channel/UCZ2bu0qutTOM0tHYa_jkIwg;
- https://www.youtube.com/channel/UCvqbFHwN-nwalWPjPUKpvTA.

Для того чтобы получить более глубокое понимание отдельных тем или даже помочь внести свой вклад в Kubernetes, вы также можете обратиться в любую из специальных групп по интересам (SIGs), связанных с Kubernetes на https://github.com/kubernetes/kubernetes/wiki/Special-Interest-Groups-(SIGs).

И наконец, поскольку Kubernetes является системой с открытым исходным кодом, есть множество информации, доступной в самом исходном коде Kubernetes. Вы найдете его на https://github.com/kubernetes/kubernetes и в родственных репозиториях.

Об авторе

Марко Лукша – инженер-программист с более чем 20-летним профессиональным опытом разработки всего, от простых веб-приложений до полных ERP-систем, вычислительных платформ и программного обеспечения среднего уровня. Он сделал свои первые шаги в программировании еще в 1985 году, в возрасте шести лет, на подержанном компьютере ZX Spectrum, который ему купил отец. В начальной школе он был национальным чемпионом в конкурсе программирования на языке Logo и посещал летние лагеря для программистов, где он научился программировать на Pascal. С тех пор он занимался разработкой программного обеспечения на широком спектре языков программирования.

В старших классах он начал создавать динамические веб-сайты, когда интернет был еще относительно молод. Затем он перешел к разработке программного обеспечения для отрасли здравоохранения и телекоммуникаций в местной компании, изучая информатику в университете Любляны, Словения. В конце концов, он перешел на работу в Red Hat, первоначально разрабатывая реализацию с открытым исходным кодом API Google App Engine, в которой использовались продукты Red Hat среднего уровня JBoss. Он также работал или участвовал в таких проектах, как CDI/Weld, Infinispan/Jboss DataGrid и др.

С конца 2014 года он является частью команды Red Hat Cloud Enablement, где его обязанности включают в себя обновление новых разработок в Kubernetes и связанных с ними технологий и обеспечение максимального задействования возможностей Kubernetes и OpenShift в программном обеспечении компании среднего уровня.

Глава 1

Знакомство с Kubernetes

Эта глава посвящена:

- описанию изменений в разработке и развертывании программного обеспечения за последние годы;
- изолированию приложений и уменьшению средовых различий с помощью контейнеров;
- описанию того, как контейнеры и Docker используются в Kubernetes;
- упрощению работы разработчиков и сисадминов с помощью Kubernetes.

Много лет назад многие программные приложения были большими монолитами, работающими либо как один процесс, либо как небольшое количество процессов, распределенных по нескольким серверам. Сегодня такие унаследованные системы по-прежнему широко распространены. Они имеют медленные циклы релизов и обновляются относительно редко. В конце каждого цикла релиза разработчики упаковывают всю систему и передают ее группе системных администраторов, которая затем ее развертывает и контролирует. В случае аварийных сбоев оборудования системные администраторы вручную переносят ее на оставшиеся работоспособные серверы.

Сегодня эти огромные монолитные устаревшие приложения медленно распадаются на более мелкие, самостоятельно работающие компоненты, называемые микросервисами. Поскольку микросервисы друг от друга отделены, их можно разрабатывать, развертывать, обновлять и масштабировать по отдельности. Это позволяет быстро изменять компоненты и так часто, как это необходимо, чтобы идти в ногу с сегодняшними быстро меняющимися бизнес-требованиями.

Но, учитывая большое количество развертываемых компонентов и все более крупные центры обработки данных, настраивать, управлять и поддер-

живать бесперебойную работу всей системы становится все труднее. Гораздо труднее понять, куда поместить каждый из этих компонентов, чтобы достичь высокой степени использования ресурсов и тем самым снизить затраты на оборудование. Делать все это вручную крайне тяжело. Нам нужна автоматизация, которая включает в себя автоматическое размещение этих компонентов на наших серверах, автоматическую настройку, контроль и обработку аварийных сбоев. Именно здесь в игру вступает Kubernetes.

Kubernetes позволяет разработчикам развертывать свои приложения самостоятельно и так часто, как они хотят, не требуя какой-либо помощи от системных администраторов. Но Kubernetes приносит пользу не только разработчикам. Эта платформа также помогает системным администраторам, автоматически отслеживая и перемещая эти приложения в случае аварийного сбоя оборудования. Основное внимание системных администраторов переключается с надзора за отдельными приложениями на надзор и контроль за платформой Kubernetes и остальной инфраструктурой, тогда как сама платформа Kubernetes заботится о приложениях.

ПРИМЕЧАНИЕ. *Kubernetes* на греческом языке означает «кормчий» или «рулевой» (лицо, держащее рулевое колесо корабля). Люди произносят слово Kubernetes по-разному. Многие произносят его как кубернетес, в то время как другие произносят больше как кубернетис. Независимо от того, какую форму вы используете, специалисты поймут, что вы имеете в виду.

Kubernetes абстрагирует аппаратную инфраструктуру и обеспечивает доступ ко всему вашему центру обработки данных (дата-центру) как одному огромному вычислительному ресурсу. Это позволяет развертывать и запускать программные компоненты без необходимости знать о фактических серверах, находящихся в основании. При развертывании многокомпонентного приложения с помощью Kubernetes он выбирает сервер для каждого компонента, развертывает его и позволяет легко находить и взаимодействовать со всеми другими компонентами приложения.

Все это делает Kubernetes отличным вариантом для большинства локальных дата-центров, но во всей красе эта система проявляет себя тогда, когда она используется в крупных центрах обработки данных, таких, которые построены и эксплуатируются облачными провайдерами. Kubernetes позволяет им предлагать разработчикам простую платформу для развертывания и запуска любого типа приложений, не требуя от облачного провайдера, чтобы собственные системные администраторы что-то знали о десятках тысяч приложений, работающих на их оборудовании.

Благодаря тому что все больше крупных компаний принимают модель Kubernetes как лучший способ запуска приложений, она становится стандартным способом запуска распределенных приложений как в облаке, так и в локальной инфраструктуре.

1.1 Объяснение необходимости системы наподобие Kubernetes

Прежде чем вы начнете подробно знакомиться с Kubernetes, давайте рассмотрим, как изменились разработка и развертывание приложений за последние годы. Это изменение является следствием как разделения больших монолитных приложений на более мелкие микросервисы, так и изменений в инфраструктуре, которая запускает эти приложения. Понимание данных изменений поможет вам лучше понять преимущества использования Kubernetes и контейнерных технологий, таких как Docker.

1.1.1 Переход от монолитных приложений к микросервисам

Монолитные приложения состоят из компонентов, которые тесно связаны друг с другом и должны разрабатываться, развертываться и управляться как одна сущность, поскольку все они выполняются как один процесс ОС. Изменения в одной части приложения требуют новой выкладки всего приложения, и со временем отсутствие жестких границ между частями приводит к увеличению сложности и последующему ухудшению качества всей системы из-за неограниченного роста взаимосвязей между этими частями.

Для запуска монолитного приложения обычно требуется небольшое количество мощных серверов, которые могут предоставить достаточно ресурсов для запуска приложения. Для того чтобы справиться с растущей нагрузкой на систему, вам нужно либо масштабировать серверы вертикально (так называемое масштабирование вверх), добавляя больше процессоров, оперативной памяти и других серверных компонентов, либо масштабировать всю систему по горизонтали, настраивая дополнительные серверы и запуская несколько копий (или реплик) приложения (масштабирование вширь). Хотя масштабирование вверх обычно не требует каких-либо изменений в приложении, оно относительно быстро становится дорогостоящим и на практике всегда имеет верхний предел. Масштабирование вширь, с другой стороны, является относительно дешевым аппаратно, но может потребовать больших изменений в программном коде приложения и не всегда возможно – некоторые части приложения с большим трудом поддаются горизонтальному масштабированию или почти невозможны для него (например, реляционные базы данных). Если какая-либо часть монолитного приложения не масштабируется, то все приложение становится немасштабируемым, если только каким-то образом этот монолит не разделить.

Разделение приложений на микросервисы

Эти и другие проблемы заставили нас начать разбиение сложных монолитных приложений на небольшие независимые развертывания компонентов, называемых микросервисами. Каждый микросервис выполняется как неза-

висимый процесс (см. рис. 1.1) и взаимодействует с другими микросервисами через простые, четко определенные интерфейсы (API).

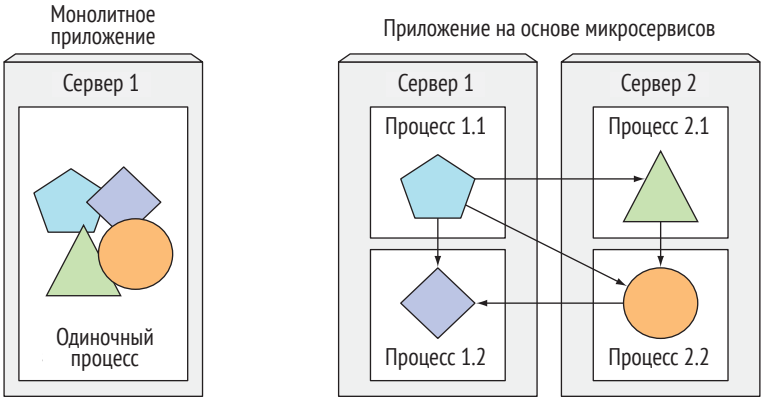

Рис. 1.1. Компоненты внутри монолитного приложения и независимых микросервисов

Микросервисы взаимодействуют через синхронные протоколы, такие как HTTP, используя которые, они обычно предоставляют RESTful, или через асинхронные протоколы, такие как AMQP (Advanced Message Queueing Protocol, расширенный протокол организации очереди сообщений). Эти протоколы просты, хорошо понятны большинству разработчиков и не привязаны к какому-либо конкретному языку программирования. Каждый микросервис может быть написан на языке, который наиболее целесообразен для реализации конкретных микросервисов.

Поскольку каждый микросервис представляет собой автономный процесс с относительно статическим внешним API, существует возможность разрабатывать и развертывать каждый микросервис отдельно. Изменение одной из них не требует изменений или повторного развертывания какого-либо другого сервиса, при условии что API не изменяется или изменяется только обратно совместимым образом.

Масштабирование микросервисов

Масштабирование микросервисов, в отличие от монолитных систем, где необходимо масштабировать систему целиком, выполняется отдельно для каждого сервиса. Это означает, что вы можете масштабировать только те сервисы, которые требуют больше ресурсов, оставляя другие сервисы в исходном масштабе. На рис. 1.2 показан пример. Некоторые компоненты реплицируются и выполняются как несколько процессов, развернутых на разных серверах, в то время как другие выполняются как один процесс приложения. Когда монолитное приложение не может быть промасштабировано вширь по причине того, что одна из ее частей немасштабируема, разделение приложения на микросервисы позволяет горизонтально масштабировать те составные части, которые позволяют масштабировать вширь, и вертикально масштабировать те части, которые нельзя масштабировать горизонтально.

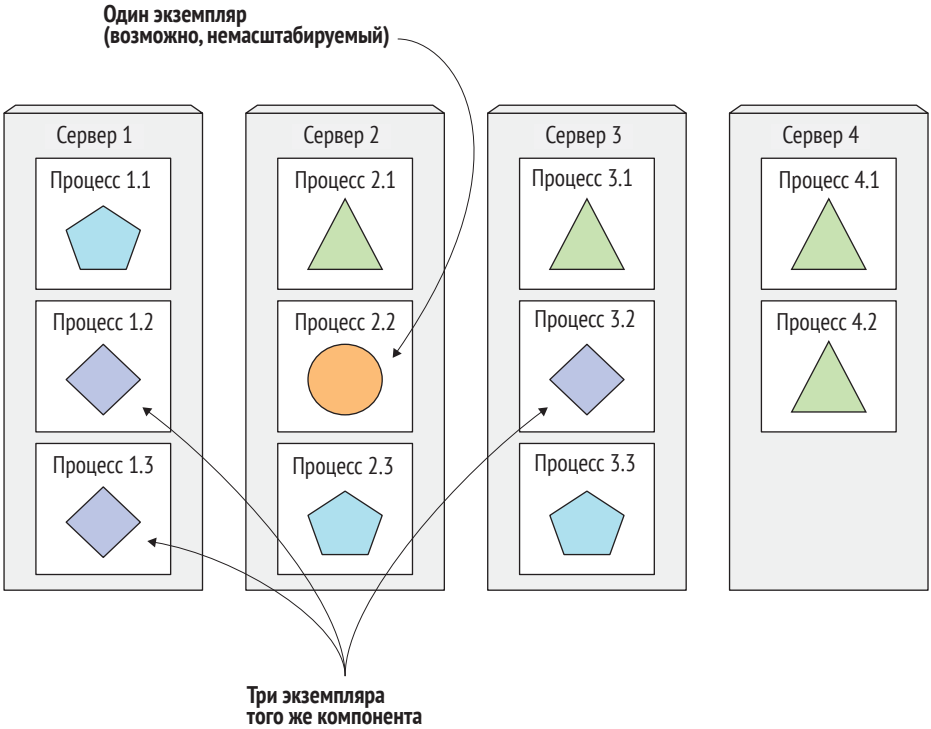

Рис. 1.2. Каждый микросервис может масштабироваться по отдельности

Развертывание микросервисов

Как это обычно бывает, микросервисы имеют свои недостатки. Если система состоит лишь из небольшого количества развертываемых компонентов, управлять этими компонентами очень просто. Вопрос, где развертывать каждый компонент, решается тривиальным образом, потому что вариантов не так много. Когда количество этих компонентов увеличивается, решения, связанные с развертыванием, становятся все более трудными, поскольку не только количество комбинаций развертывания увеличивается, но и количество взаимосвязей между компонентами увеличивается еще больше.

Микросервисы выполняют свою работу вместе, как команда, поэтому им нужно найти друг друга и пообщаться. При их развертывании кто-то или что-то должно правильно настроить их все, чтобы они могли работать вместе как одна система. С увеличением количества микросервисов это становится утомительным и подверженным ошибкам, в особенности если учесть то, что сисадмины должны делать, когда сервер аварийно завершает свою работу.

Микросервисы также создают другие проблемы, к примеру затрудняют отладку и трассировку вызовов выполнения, поскольку они охватывают несколько процессов и машин. К счастью, эти проблемы теперь решаются с помощью систем распределенного сбора трассировок, таких как Zipkin.

Понимание различий в требованиях к окружению

Как я уже упоминал, компоненты в архитектуре микросервисов не только развертываются независимо, но и разрабатываются таким же образом. Из-за их независимости и того факта, что, как правило, каждый компонент разрабатывается отдельными командами, ничто не мешает каждой команде использовать разные библиотеки и заменять их всякий раз, когда возникает необходимость. Расхождение в библиотеках между компонентами приложения, как показано на рис. 1.3, где в любом случае требуются разные версии одной библиотеки, – это неизбежность.

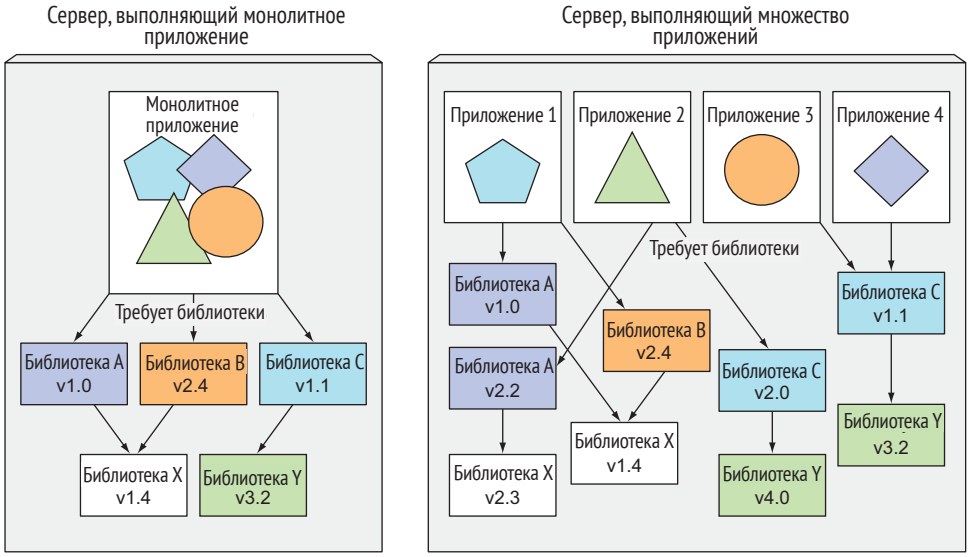

Рис. 1.3. Несколько приложений, работающих на одном хосте, могут иметь конфликтующие версии библиотек, от которых они функционально зависимы

Развертывание динамически связанных приложений, для которых требуются различные версии совместно используемых библиотек и/или другие особенности среды, может быстро стать кошмаром для сисадминов, которые развертывают их и управляют ими на рабочих серверах. Чем больше количество компонентов необходимо развернуть на одном хосте, тем сложнее будет управлять всеми их зависимостями, чтобы удовлетворить все их требования.

1.1.2 Обеспечение консистентного окружения для приложений

Независимо от того, сколько отдельных компонентов вы разрабатываете и развертываете, одна из самых больших проблем, с которой всегда приходится сталкиваться разработчикам и системным администраторам, – это различия в окружениях, в которых они выполняют свои приложения. Мало того, что существует огромная разница между окружением разработки и рабочим окру-

жением, различия даже существуют между отдельными машинами в рабочем окружении. Еще одним неизбежным фактом является то, что окружение одной рабочей машины будет меняться с течением времени.

Эти различия варьируются от аппаратного обеспечения до операционной системы и библиотек, доступных на каждой машине. Рабочие окружения управляются системными администраторами, в то время как разработчики занимаются своими ноутбуками, на которых ведется разработка, самостоятельно. Разница в том, насколько эти две группы людей знают о системном администрировании, и это, по понятным причинам, приводит к относительно большим различиям между этими двумя системами, не говоря уже о том, что системные администраторы уделяют гораздо больше внимания последним обновлениям в системе безопасности, в то время как многих разработчиков это не интересует в такой же мере.

Кроме того, рабочие системы могут выполнять приложения от нескольких разработчиков или групп разработчиков, что не обязательно верно для компьютеров разработчиков. Рабочая система должна обеспечивать надлежащую среду для всех размещаемых на ней приложений, даже если для них могут потребоваться разные, порой конфликтующие версии библиотек.

Для того чтобы уменьшить количество проблем, которые проявляются только в рабочем окружении, было бы идеально, если бы приложения могли работать в одной и той же среде во время разработки и в «бою», чтобы они имели одну и ту же операционную систему, библиотеки, конфигурацию системы, сетевую среду и все остальное. Вы также не хотите, чтобы эта среда слишком сильно менялась с течением времени, если вообще менялась. Кроме того, если это возможно, требуется способность добавлять приложения на тот же сервер, не затрагивая ни одно из существующих приложений на этом сервере.

1.1.3 Переход к непрерывной доставке: DevOps и NoOps

В последние несколько лет мы также наблюдаем изменения во всем процессе разработки приложений и в том, как приложения обслуживаются в рабочем окружении. В прошлом работа команды разработчиков состояла в создании приложения и ее передаче группе системных администраторов, которая затем его развертывала, сопровождала и поддерживала его работу. Однако теперь организации понимают, что лучше, чтобы та же команда, которая разрабатывает приложение, также принимала участие в его развертывании и сопровождала его на протяжении всей его жизни. Это означает, что команды разработчиков, QA и системные администраторы теперь должны сотрудничать на протяжении всего процесса. Подобная практика называется *DevOps*.

Преимущества

Когда разработчики принимают более активное участие в выполнении приложения в производстве, это приводит к тому, что они имеют лучшее понимание потребностей и нужд пользователей и проблем, с которыми сталкиваются

системные администраторы при сопровождении приложения. Разработчики приложений теперь также гораздо более склонны передавать пользователям приложение раньше, а затем использовать их отзывы, чтобы определять пути дальнейшего развития приложения.

Для того чтобы чаще выпускать новые версии приложений, необходимо оптимизировать процесс развертывания. В идеале требуется, чтобы разработчики развертывали приложения сами, не дожидаясь системных администраторов. Однако развертывание приложения часто требует понимания базовой инфраструктуры и организации оборудования в дата-центре. Разработчики не всегда знают эти детали и в большинстве случаев даже не хотят знать о них.

Позвольте разработчикам и сисадминам делать то, что они делают лучше всего

Несмотря на то что разработчики и системные администраторы работают для достижения одной и той же цели – запуск успешного программного приложения в качестве службы для своих клиентов, они имеют различные индивидуальные цели и мотивирующие факторы. Разработчики любят создавать новые функциональные средства и улучшать пользовательский опыт. Обычно они не хотят следить за тем, чтобы своевременно вносились исправления в систему безопасности базовой операционной системы, и т. п. Они предпочитают оставлять это на усмотрение системных администраторов.

Системные администраторы отвечают за процесс развертывания программного обеспечения в рабочем окружении и аппаратную инфраструктуру, на которой они работают. Они занимаются обеспечением безопасности системы, задействованием ресурсов и другими аспектами, которые не являются приоритетными для разработчиков. Системные администраторы не хотят иметь дело с неявными взаимозависимостями всех компонентов приложения и не хотят думать о том, как изменения операционной системы или инфраструктуры могут повлиять на работу приложения в целом, но они влияют.

В идеале требуется, чтобы разработчики развертывали приложения сами, ничего не зная об аппаратной инфраструктуре и не имея дела с системными администраторами. Это называется *NoOps*. Очевидно, что вам все же нужно, чтобы кто-то занимался аппаратной инфраструктурой, при этом, в идеале, не занимаясь особенностями каждого приложения, работающего на нем.

Как вы увидите, Kubernetes позволяет нам достичь всего этого. Абстрагируясь от фактического оборудования и обеспечивая к нему доступ как к единой платформе для развертывания и выполнения приложений, эта инфраструктура позволяет разработчикам настраивать и развертывать свои приложения без какой-либо помощи от системных администраторов и дает возможность системным администраторам сосредоточиваться на поддержании базовой инфраструктуры в рабочем состоянии, не зная ничего о реальных приложениях, работающих поверх него.