АДМИНИСТРИРОВАНИЕ ОПЕРАЦИОННЫХ СИСТЕМ

Екатеринбург
2023

Министерство науки и высшего образования Российской Федерации
Уральский государственный экономический университет

АДМИНИСТРИРОВАНИЕ ОПЕРАЦИОННЫХ СИСТЕМ

Рекомендовано
Советом по учебно-методическим вопросам и качеству образования
Уральского государственного экономического университета
в качестве учебного пособия

Екатеринбург
2023

УДК 004.451(075)
ББК 32.973я73
А31

Рецензенты:

Ученый совет Уральского института фондового рынка
(протокол № 8 от 12 апреля 2023 г.);

директор учебно-научного центра «Информационная безопасность»
Института радиоэлектроники и информационных технологий — РтФ
Уральского федерального университета имени первого Президента России Б. Н. Ельцина,
доктор технических наук, профессор
С. В. Поршнев

Авторский коллектив:
В. П. Часовских, *В. Г. Лабунец*, *Е. Н. Стариков*, *Г. А. Акчурина, Е. В. Кох*

А31 **Администрирование операционных систем** : учебное пособие / В. П. Часовских, В. Г. Лабунец, Е. Н. Стариков [и др.] ; Министерство науки и высшего образования Российской Федерации, Уральский государственный экономический университет. — Екатеринбург : УрГЭУ, 2023. — 128 с.

В учебном пособии рассматривается администрирование операционных систем Windows Server, Windows и Linux в приложениях сквозных технологий цифровой экономики Российской Федерации. В Windows предусмотрено множество инструментов, предназначенных для администрирования или управления компьютером. Изложены основные задачи системного администрирования, описаны базовые протоколы, даны рекомендации по выбору оборудования и проведению ежедневных рутинных операций. Подробно раскрыты технологии, используемые при построении информационных систем, описаны средства мониторинга и обслуживания как малых, так и распределенных сетей. Рассмотрены методы централизованного управления, основы создания безопасной среды. Даны рекомендации по поиску неисправностей, обеспечению защиты данных. Параллельно рассмотрены решения на основе операционных систем Windows и Linux с использованием как проприетарных, так и открытых технологий.

Предназначено для студентов очной и заочной форм обучения направлений подготовки бакалавриата 02.03.03 «Математическое обеспечение и администрирование информационных систем», 09.03.01 «Информатика и вычислительная техника», 09.03.03 «Прикладная информатика», а также может быть полезно направлениям 10.03.01 «Информационная безопасность» и 09.04.03 «Прикладная информатика».

УДК 004.451(075)
ББК 32.973я73

Введение

Федеральные программы развития информационного общества в Российской Федерации на 2017–2030 гг. направлены на создание условий для развития общества знаний, улучшение благосостояния и качества жизни путем повышения доступности и качества товаров и услуг, произведенных в цифровой экономике с использованием современных цифровых технологий, повышения степени информированности и цифровой грамотности, улучшения доступности и качества государственных услуг для граждан, а также безопасности как внутри страны, так и за ее пределами. Важное место в цифровой экономике занимают вопросы администрирования операционных систем для сквозных технологий.

Операционная система — главный компонент компьютера или мобильного устройства. Она отвечает за управление всеми программами и ресурсами устройства, такими как процессор, память, хранение данных и др. Примерами операционных систем являются Windows, Windows Server, Linux, Mac OS, Android и iOS.

Администрирование операционной системы — управление ее компонентами. Администрирование является важным и ответственным делом, так как один человек, он же администратор операционной системы, не покидая своего рабочего места, влияет на работу целой сети, в которую может входить от нескольких до многих тысяч компьютеров.

В настоящее время операционная система предназначена не только для настольных компьютеров или смартфонов, но и для реализации сквозных технологий национальной программы цифровой экономики Российской Федерации, прежде всего Big Data и машинного обучения технологий искусственного интеллекта.

Управление операционной системой для локального компьютера определяется при установке и дальнейшем изменении параметров в интересах пользователя и выполняется самим пользователем.

В учебном пособии рассматривается SQL Server 2022 — инновационная версия популярной системы управления базами данных. Будет уделено внимание вопросам производительности и безопасности, использования контейнеров и технологии Kubernetes, работе с кластерами больших данных и средствах машинного обучения.

Изучение новых функций SQL Server 2022 позволит студентам расширить свои навыки в области управления и извлечения информации из больших данных, машинного обучения ML.NET и технологий искусственного интеллекта.

Материал учебного пособия знакомит студентов, изучающих проблемы вычислительной техники, с инженерными методами защиты программ и данных в современных вычислительных системах. Наряду с описанием стандартных методов в учебное пособие вошли также материалы, освещающие сквозные технологии цифровой экономики Российской Федерации.

В учебном пособии приведены основные определения и дан всесторонний обзор современных принципов администрирования операционных систем, которыми следует руководствоваться при проектировании и эксплуатации информационных систем, актуальных для цифровой экономики России.

Глава 1

Операционная система

1.1. Понятие и функции операционных систем

Операционная система (ОС) — комплекс управляющих и обрабатывающих программ, которые, с одной стороны, выступают как интерфейс между устройствами вычислительной системы и прикладными программами, а с другой — предназначены для управления устройствами и вычислительными процессами, для эффективного распределения вычислительных ресурсов между вычислительными процессами и для организации надежных вычислений.

В большинстве вычислительных систем операционная система является основной и наиболее важной частью системного программного обеспечения. С 1990-х гг. наиболее распространенными операционными системами являются системы семейства Windows и системы класса UNIX (особенно Linux и Mac OS). С 2000-х гг. большое распространение получили мобильные компьютеры (смартфоны и планшеты) и ОС Android и iOS соответственно.

ОС является необходимой составляющей программного обеспечения (ПО), без которой компьютер не будет работать.

Операционная система выполняет следующие базовые функции (рис. 1.1):

1) управляет файловой системой (просмотр, удаление, копирование, перемещение, переименование);

2) запуск и завершение прикладных программ;

3) всевозможные виды сервиса (информация о параметрах, их настройка, оптимизация работы и т. д.)

4) предоставляет минимальный необходимый набор функций по реализации и функционированию устройств.

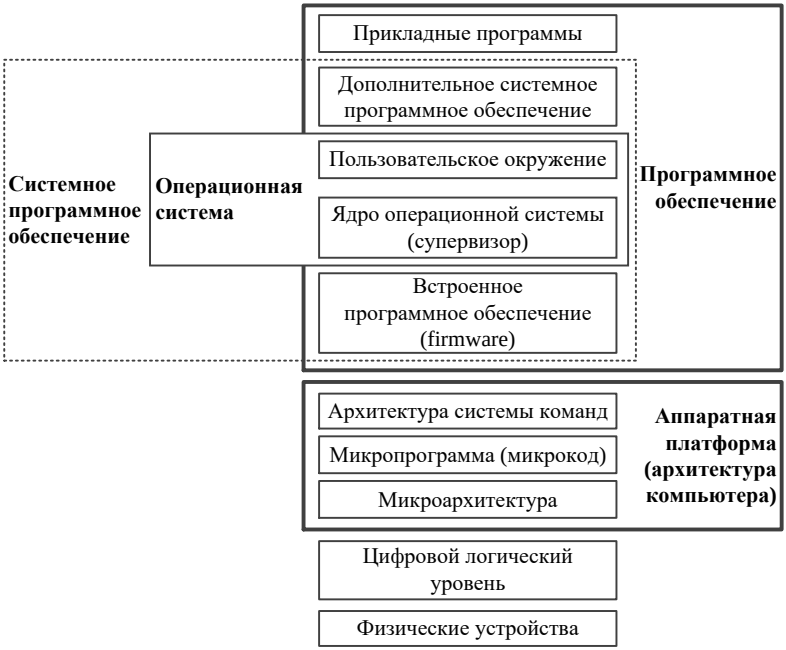

Рис. 1.1. Место операционной системы в структуре компьютера

Разработчикам прикладных программ ОС позволяет не думать о деталях реализации и функционирования устройств, предоставляя минимально необходимый набор функций по работе с ним.

Рассмотрим **основные функции ОС**:

– выполнение по запросу программ тех низкоуровневых действий, которые являются общими для большинства программ и часто встречаются почти во всех (ввод и вывод данных, запуск и остановка других программ, выделение и освобождение дополнительной памяти и др.);

– загрузка программ в оперативную память и их выполнение;

– стандартизованный доступ к периферийным устройствам (устройства ввода-вывода);

– управление оперативной памятью (распределение между процессами, организация виртуальной памяти);

– управление доступом к данным на энергонезависимых носителях (таких как жесткий диск, оптические диски и др.), организованным в той или иной файловой системе;

– обеспечение пользовательского интерфейса;

– сетевые операции, поддержка стека сетевых протоколов.

Дополнительные функции:

– параллельное или псевдопараллельное выполнение задач (многозадачность);

– эффективное распределение ресурсов вычислительной системы между процессами;

– разграничение доступа различных процессов к ресурсам;

– организация надежных вычислений (невозможность одного вычислительного процесса намеренно или по ошибке повлиять на вычисления в другом процессе), основанная на разграничении доступа к ресурсам;

– взаимодействие между процессами: обмен данными, взаимная синхронизация;

– защита самой системы, а также пользовательских данных и программ от действий пользователей или приложений;

– многопользовательский режим работы и разграничение прав доступа.

1.2. Состав и загрузка операционных систем

В состав ОС входят следующие элементы.

1. **Программный модуль**, управляющий файловой системой. Процесс работы компьютера сводится к обмену файлами между устройствами.

2. **Командный процессор** — специальная программа, которая запрашивает у пользователя команды и выполняет их. Пользователь может дать команду запуска программы, выполнения ка-

кой-либо операции над файлами (копирование, удаление, переименование), вывода документа на печать и т. д.

3. **Драйверы** — программы, которые управляют работой устройств. Каждому устройству соответствует свой драйвер. Технология «Plug and Play» (подключи и играй) позволяет автоматизировать подключение новых устройств. В процессе установки Windows определяет тип и конкретную модель установленного устройства и подключает необходимый для его функционирования драйвер. При включении компьютера производится загрузка драйверов в оперативную память. Пользователь имеет возможность вручную установить или переустановить драйверы.

4. **Программные модули графического интерфейса** — программы, позволяющие пользователю вводить команды с помощью мыши.

5. **Графический интерфейс пользователя** — разновидность пользовательского интерфейса, в котором элементы интерфейса (меню, кнопки, значки, списки и т. п.), представленные пользователю на дисплее, исполнены в виде графических изображений.

В отличие от интерфейса командной строки, в графическом интерфейсе пользователь имеет произвольный доступ (с помощью устройств ввода: клавиатуры, мыши, джойстика и т. п.) ко всем видимым экранным объектам (элементам интерфейса) и осуществляет непосредственное манипулирование ими. Чаще всего элементы интерфейса реализованы на основе метафор и отображают их назначение и свойства, что облегчает понимание и освоение программ неподготовленными пользователями. Одним из требований к хорошему графическому интерфейсу программной системы является концепция «Делай то, что я имею в виду», или DWIM (Do What I Mean). DWIM требует, чтобы система работала предсказуемо, чтобы пользователь заранее интуитивно понимал, какое действие выполнит программа после получения его команды.

6. **Утилиты** — сервисные программы для обслуживания дисков (проверять, сжимать, дефрагментировать и т. д.), выполнения операций с файлами (архивировать, копировать и т. д.) и работы в компьютерных сетях.

7. **Справочная система** — получение информации о функционировании ОС в целом и о работе ее отдельных модулей.

8. **Файлы** операционной системы хранятся во внешней, долговременной памяти. Но программы могут выполняться, только если они находятся в оперативной памяти, поэтому файлы ОС необходимо загрузить туда.

Диск, на котором находятся файлы операционной системы и с которого производится ее загрузка, называется системным. После включения ПК производится загрузка ОС с системного диска в оперативную память. Она должна выполняться в соответствии с программой загрузки, однако для того, чтобы компьютер выполнял программу, ей необходимо уже находиться в оперативной памяти. Разрешение этого противоречия состоит в последовательной, поэтапной загрузке операционной системы.

В постоянном запоминающем устройстве содержатся программы тестирования ПК и первого этапа загрузки ОС — это BIOS (Basic Input/Output System — базовая система ввода/вывода). После включения питания процессор начинает выполнение программы самотестирования компьютера POST (Power-ON Self Test). Производится тестирование работоспособности процессора, памяти и других аппаратных средств компьютера.

После проведения самотестирования специальная программа в BIOS начинает поиск загрузчика ОС. Происходит поочередное обращение к имеющимся дискам и поиск на определенном месте (в первом загрузочном секторе диска) наличия специальной программы Master Boot (программы-загрузчика ОС).

Master Boot загружается в ОС, и ей передается управление работой компьютера. Программа ищет файлы операционной системы на системном диске и загружает их в оперативную память в качестве программных модулей.

Инсталляция программ — процедура установки большинства программных продуктов, при которой используется специальная дистрибутивная копия.

Копирование программного продукта на жесткий диск называется установкой. Файл с программой имеет расширение .exe, .com, .bat. Он работает автономно или в сопровождении служебных файлов. Запустить программу — значит начать ее работу,

но этот программный продукт должен быть совместим с аппаратными средствами.

ОС выполняет две группы функций:

– предоставление пользователю или программисту вместо реальной аппаратуры компьютера расширенной виртуальной машины, с которой удобней работать и которую легче программировать;

– повышение эффективности использования компьютера путем рационального управления его ресурсами в соответствии с некоторым критерием.

Если говорить подробнее, то в функции ОС будут включены:

– управление процессами;
– управление памятью;
– управление файлами и внешними устройствами;
– защита данных и администрирование;
– интерфейс прикладного программирования;
– пользовательский интерфейс и т. д.

Если говорить о пользовательском интерфейсе, то операционная система должна обеспечивать удобный интерфейс не только для прикладных программ, но и для человека, работающего за терминалом.

В ранних операционных системах пакетного режима функции пользовательского интерфейса были сведены к минимуму и не требовали наличия терминала. Команды языка управления заданиями набивались на перфокарты, а результаты выводились на печатающее устройство. Современные же ОС поддерживают развитые функции пользовательского интерфейса для интерактивной работы за терминалами двух типов: алфавитно-цифровыми и графическими.

При работе за алфавитно-цифровым терминалом пользователь имеет в своем распоряжении систему команд, мощность которых отражает функциональные возможности этой ОС. Обычно командный язык ОС позволяет запускать и останавливать приложения, выполнять различные операции с файлами и каталогами, получать информацию о состоянии ОС (количество работающих процессов, объем свободного пространства на дисках и т. п.), ад-

министрировать систему. Команды могут вводиться не только в интерактивном режиме с терминала, но и считываться из так называемого командного файла, содержащего некоторую последовательность команд.

Программный модуль ОС, ответственный за чтение отдельных команд или же последовательности команд из командного файла, иногда называют командным интерпретатором.

Ввод команды может быть упрощен, если операционная система поддерживает графический пользовательский интерфейс. В этом случае пользователь для выполнения нужного действия с помощью мыши выбирает на экране нужный пункт меню или графический символ.

Выводы по главе 1

Определены основные компоненты операционной системы и ее администрирование. Сформулированы следующие понятия операционной системы: базовые, основные и дополнительные функции, состав и группы функций. Рассмотрена процедура загрузки операционной системы.

Вопросы для самоконтроля

1. В чем заключается понятие операционной системы?
2. Каковы базовые функции операционной системы?
3. Каковы основные функции операционной системы?
4. Каковы дополнительные функции операционной системы?
5. В чем заключается состав операционной системы?
6. В чем заключается загрузка операционной системы?
7. В чем заключаются группы функций операционной системы?

Глава 2

Администрирование операционных систем

Цель администрирования ОС — обеспечение ее надежного функционирования.

Одна из важнейших задач администрирования — разграничение прав доступа к различным ресурсам.

Настройка операционной системы относится к задачам администрирования ПК и состоит:

– в добавлении пользователей в систему или удалении из нее;

– выборе настроек пользовательского окружения по умолчанию (политики групп);

– управлении правами пользователей;

– установке и конфигурации драйверов устройств;

– разбиении винчестера на разделы;

– выборе размера виртуальной памяти, используемой системой;

– управлении списком сервисов, запускаемых автоматически при загрузке ОС;

– конфигурации сетевых настроек;

– настройке брандмауэра (firewall);

– выборе стандартных переменных среды.

Требования к современным операционным системам

Главным требованием, предъявляемым к операционной системе, является выполнение ею основных функций эффективного

управления ресурсами и обеспечение удобного интерфейса для пользователя и прикладных программ.

Современная операционная система должна поддерживать мультипрограммную обработку, виртуальную память, свопинг, многооконный графический интерфейс пользователя, а также удовлетворять следующим эксплуатационным требованиям.

1. **Расширяемость.** В то время как аппаратная часть компьютера устаревает за несколько лет, полезная жизнь операционных систем может измеряться десятилетиями. Если код ОС написан таким образом, что дополнения и изменения могут вноситься без нарушения целостности системы, такую ОС называют расширяемой.

2. **Переносимость.** В идеале код ОС должен легко переноситься с процессора одного типа на процессор другого типа и с аппаратной платформы (которые различаются не только типом процессора, но и способом организации всей аппаратуры компьютера) одного типа на аппаратную платформу другого типа.

3. **Совместимость.** Если ОС имеет средства для выполнения прикладных программ, написанных для других операционных систем, она обладает совместимостью с этими ОС.

4. **Надежность и отказоустойчивость.** Система должна быть защищена как от внутренних, так и от внешних ошибок, сбоев и отказов. Ее действия должны быть всегда предсказуемыми, а приложения не должны иметь возможности наносить вред ОС.

5. **Безопасность.** Современная ОС должна защищать данные и другие ресурсы вычислительной системы от несанкционированного доступа.

6. **Производительность.** Операционная система должна обладать настолько хорошим быстродействием и временем реакции, насколько это позволяет аппаратная платформа.

Назначение и основные функции операционных систем

Операционная система компьютера обычно понимается как комплекс взаимосвязанных программ, который является интерфейсом между приложениями и пользователями, с одной стороны, и аппаратурой компьютера, с другой стороны.

Основное назначение ОС состоит в предоставлении пользователю или программисту некоторой расширенной виртуальной машины, которую легче программировать и с которой легче работать, чем непосредственно с аппаратурой, составляющей реальный компьютер или реальную сеть.

Управление ресурсами вычислительной системы с целью наиболее эффективного их использования является основным назначением любой операционной системы.

Основные ресурсы современных вычислительных систем: процессоры и основная память. Ресурсы распределяются между процессами.

Процесс (задача) — программа в стадии выполнения.

Программа — это статический объект, представляющий собой файл с кодами и данными.

Процесс — это динамический объект, который возникает в операционной системе после того, как пользователь или сама операционная система решает запустить программу на выполнение, т. е. создать новую единицу вычислительной работы.

Основные критерии эффективности, в соответствии с которыми ОС организует управление ресурсами компьютера:

– пропускная способность вычислительной системы;

– время реакции.

Управление ресурсами включает решение следующих общих, не зависящих от типа ресурса задач:

– планирование ресурса, т. е. определение, какому процессу, когда и в каком количестве (если ресурс может выделяться частями) следует выделить этот ресурс;

– удовлетворение запросов на ресурсы;

– отслеживание состояния и учет использования ресурса, т. е. поддержание оперативной информации о том, занят или свободен ресурс и какая доля ресурса уже распределена;

– разрешение конфликтов между процессами.

Большинство функций управления ресурсами выполняются операционной системой автоматически и прикладному программисту недоступны.

2.1. Сетевые операционные системы

Операционная система компьютерной сети — комплекс взаимосвязанных программ, который обеспечивает удобство работы пользователям и программистам путем предоставления им некоторой виртуальной вычислительной системы и реализует эффективный способ разделения ресурсов между множеством выполняемых в сети процессов.

Компьютерная сеть — это набор компьютеров, связанных коммуникационной системой и снабженных соответствующим программным обеспечением, позволяющим пользователям сети получать доступ к ресурсам этого набора компьютеров. Компьютерная сеть позволяет пользователю работать со своим компьютером как с автономным и добавляет к этому возможность доступа к информационным и аппаратным ресурсам других компьютеров сети.

Коммуникационная система может включать в себя кабели, повторители, коммутаторы, маршрутизаторы и другие устройства, обеспечивающие передачу сообщений между любой парой компьютеров сети.

Термин «сетевая операционная система» используется в двух значениях: во-первых, как совокупность ОС всех компьютеров сети и, во-вторых, как операционная система отдельного компьютера, способного работать в сети.

Функциональные компоненты сетевой ОС:

1) средства управления локальными ресурсами компьютера реализуют все функции ОС автономного компьютера (распределение оперативной памяти между процессами, планирование и диспетчеризацию процессов, управление процессорами в мультипроцессорных машинах, управление внешней памятью, интерфейс с пользователем и т. д.);

2) сетевые средства, в свою очередь, можно разделить на три компонента:

– средства предоставления локальных ресурсов и услуг в общее пользование — серверная часть ОС;

– средства запроса доступа к удаленным ресурсам и услугам — клиентская часть ОС;

– транспортные средства ОС, которые совместно с коммуникационной системой обеспечивают передачу сообщений между компьютерами сети.

Совокупность серверной и клиентской частей ОС, предоставляющих доступ к конкретному типу ресурса компьютера через сеть, называется сетевой службой. Каждая служба связана с определенным типом сетевых ресурсов и (или) определенным способом доступа к этим ресурсам.

Сетевая служба обычно предоставляет пользователям сети некоторый набор услуг. Эти услуги часто называют сетевым сервисом.

На практике сложилось несколько подходов к построению сетевых операционных систем, различающихся глубиной внедрения сетевых служб в операционную систему:

– сетевые службы глубоко встроены в ОС;

– сетевые службы объединены в виде некоторого набора — оболочки;

– сетевые службы производятся и поставляются в виде отдельного продукта.

Если сетевые функции глубоко встраиваются в основные модули опреационной системы, то это обеспечивает простоту эксплуатации и модификации, а также высокую производительность. При таком подходе отсутствует избыточность. Если все сетевые службы хорошо интегрированы, т. е. рассматриваются как неотъемлемые части ОС, все внутренние механизмы такой операционной системы могут быть оптимизированы для выполнения сетевых функций.

2.2. Функции операционных систем по управлению ресурсами компьютера

Управление ресурсами вычислительной системы с целью наиболее эффективного их использования является основным назначением любой операционной системы.

Основными ресурсами современных вычислительных систем являются процессоры и основная память. Ресурсы распределяются между процессами.

Управление процессами

Подсистема управления процессами планирует выполнение процессов, т. е. распределяет процессорное время между несколькими одновременно существующими в системе процессами, занимается созданием и уничтожением процессов, обеспечивает процессы необходимыми системными ресурсами, поддерживает синхронизацию, обеспечивает взаимодействие между процессами.

Для каждого вновь создаваемого процесса ОС генерирует системные информационные структуры процесса, которые содержат данные о потребностях процесса в ресурсах вычислительной системы, а также о фактически выделенных ему ресурсах.

Чтобы процесс мог быть выполнен, операционная система должна назначить ему область оперативной памяти, в которой будут размещены коды и данные процесса, а также предоставить ему необходимое количество процессорного времени.

В мультипрограммной операционной системе одновременно может существовать несколько процессов. Часть процессов порождается по инициативе пользователей и их приложений, такие процессы обычно называют пользовательскими. Другие процессы, называемые системными, инициализируются самой операционной системой для выполнения своих функций.

Важной задачей операционной системы является защита ресурсов, выделенных этому процессу, от остальных процессов. Одним из наиболее тщательно защищаемых ресурсов процесса являются области оперативной памяти, в которой хранятся коды и данные процесса. Совокупность всех областей оперативной памяти, выделенных операционной системой процессу, называется его адресным пространством.

На протяжении периода существования процесса его выполнение может быть многократно прервано и продолжено. Для возобновления выполнения процесса, необходимо восстановить состояние его контекста.

Контекст процесса — состояние операционной среды, которое идентифицируется состоянием регистров и программного счетчика, режимом работы процессора, указателями на открытые файлы, информацией о незавершенных операциях ввода-вывода, кодами ошибок, выполняемых этим процессом системных вызовов и т. д.

Управление памятью

Управление памятью включает распределение имеющейся физической памяти между всеми существующими в системе в конкретный момент процессами, загрузку кодов и данных процессов в отведенные им области памяти, настройку адресно-зависимых частей кодов процесса на физические адреса выделенной области, а также защиту областей памяти каждого процесса.

Функциями ОС по управлению памятью являются:

– отслеживание свободной и занятой памяти;

– выделение памяти процессам и освобождение памяти при завершении процессов;

– защита памяти;

– вытеснение процессов из оперативной памяти на диск, когда размеры основной недостаточны для размещения в ней всех процессов, и возвращение их в оперативную память, когда в ней освобождается место;

– настройка адресов программы на конкретную область физической памяти.

Одним из наиболее популярных способов управления памятью в современных операционных системах является использование механизма виртуальной памяти, при котором все данные, используемые программой, хранятся на диске и при необходимости частями (сегментами или страницами) отображаются в физической памяти. Это позволяет программисту писать программу так, как будто в его распоряжении имеется однородная оперативная память большого объема, часто существенно превышающего объем имеющейся физической памяти. При перемещении кодов и данных между оперативной памятью и диском подсистема виртуальной памяти выполняет трансляцию виртуальных адресов, полученных в результате компиляции и компоновки программы, в физические адреса ячеек оперативной памяти. Очень важно, что все операции по перемещению кодов и данных между оперативной памятью и дисками, а также трансляция адресов выполняются ОС прозрачно для программиста.

Защита памяти — это избирательная способность предохранять выполняемую задачу от записи или чтения памяти, назначенной другой задаче. Средства защиты памяти, реализованные

в операционной системе, должны пресекать несанкционированный доступ процессов к чужим областям памяти.

Управление файлами и внешними устройствами

Файл — простая неструктурированная последовательность байтов, имеющая символьное имя. Для удобства работы файлы группируются в каталоги, которые, в свою очередь, образуют группы — каталоги более высокого уровня. Пользователь может с помощью ОС выполнять над файлами и каталогами такие действия, как поиск по имени, удаление, вывод содержимого на внешнее устройство (например, на дисплей), изменение и сохранение содержимого.

Файловая система ОС выполняет преобразование символьных имен файлов, с которыми работает пользователь или прикладной программист, в физические адреса данных на диске, организует совместный доступ к файлам, защищает их от несанкционированного доступа.

При выполнении своих функций файловая система тесно взаимодействует с подсистемой управления внешними устройствами, которая по запросам файловой системы осуществляет передачу данных между дисками и оперативной памятью.

Подсистема управления внешними устройствами, называемая также подсистемой ввода-вывода, исполняет роль интерфейса ко всем устройствам, подключенным к компьютеру. Программа, управляющая конкретной моделью внешнего устройства и учитывающая все его особенности, обычно называется драйвером этого устройства. Для пользователя очень важно, чтобы операционная система включала как можно больше разнообразных драйверов, так как это гарантирует возможность подключения к компьютеру большого числа внешних устройств различных производителей.

Прикладные программисты могут пользоваться интерфейсом драйверов при разработке своих программ, но такой интерфейс обычно представляет собой низкоуровневые операции, обремененные большим количеством деталей.

Поддержание высокоуровневого унифицированного интерфейса прикладного программирования к разнородным устройствам ввода-вывода является одной из наиболее важных задач ОС.

Операционная система должна обеспечивать удобный интерфейс не только для прикладных программ, но и для человека, работающего за компьютером. Современные ОС поддерживают разнообразные функции пользовательского интерфейса для интерактивной работы: алфавитно-цифровые терминалы, графические интерфейсы пользователя, модули голосовых команд, речевое управление, интерфейсы на основе анализа биологических сигналов мозга человека и т. д.

При работе за алфавитно-цифровым терминалом пользователь имеет в своем распоряжении систему команд, которая отражает функциональные возможности этой ОС. Команды могут вводиться не только в интерактивном режиме с терминала, но и считываться из так называемого командного файла, содержащего некоторую последовательность команд.

Программный модуль ОС, ответственный за чтение отдельных команд или же последовательности команд из командного файла, иногда называют командным интерпретатором.

Ввод команды может быть упрощен, если операционная система поддерживает графический пользовательский интерфейс. В этом случае пользователь для выполнения нужного действия с помощью мыши выбирает на экране нужный пункт меню или графический символ.

2.3. Обобщенная структура операционной системы

Структурную организацию ОС на основе различных программных модулей называют архитектурой. Обычно в состав ОС входят исполняемые и объектные модули стандартных для этой ОС форматов, библиотеки разных типов, модули исходного текста программ, программные модули специального формата (например, загрузчик ОС, драйверы ввода-вывода), конфигурационные файлы, файлы документации, модули справочной системы и т. д.

Наиболее общим подходом к структуризации операционной системы является разделение всех ее модулей на две группы:

– ядро, выполняющее основные функции ОС;

– модули, выполняющие вспомогательные функции ОС.

Ядро — центральная часть операционной системы, управляющая выполнением процессов, ресурсами вычислительной си-

стемы и предоставляющая процессам координированный доступ к этим ресурсам. Основными ресурсами являются процессорное время, память и устройства ввода-вывода. Доступ к файловой системе и сетевое взаимодействие также могут быть реализованы на уровне ядра.

Модули ядра выполняют такую базовую функцию ОС, как управление процессами, памятью, устройствами ввода-вывода и т. п. Ядро составляет сердцевину операционной системы, без него ОС является полностью неработоспособной и не сможет выполнить ни одну из своих функций. В состав ядра входят функции, решающие внутрисистемные задачи организации вычислительного процесса, такие как переключение контекстов, загрузка или выгрузка станиц, обработка прерываний. Эти функции недоступны для приложений. Другой класс функций ядра служит для поддержки приложений, создавая для них так называемую прикладную программную среду. Приложения могут обращаться к ядру с запросами (системными вызовами) для выполнения тех или иных действий, например, для открытия и чтения файла, вывода графической информации на дисплей, получения системного времени и т. д. Функции ядра, которые могут вызываться приложениями, образуют интерфейс прикладного программирования — API.

Для обеспечения высокой скорости работы ОС все модули ядра или большая их часть постоянно находятся в оперативной памяти, т. е. являются резидентными.

Обычно ядро оформляется в виде программного модуля некоторого специального формата, отличающегося от формата пользовательских приложений. Термин «ядро» в разных ОС трактуется по-разному. Одним из определяющих свойств ядра является работа в привилегированном режиме.

Вспомогательные модули ОС выполняют менее обязательные функции: программы архивирования данных, дефрагментации диска, текстового редактора и т. п. Оформляются либо в виде приложений, либо в виде библиотек процедур. Обычно они подразделяются на следующие группы:

– утилиты — программы, решающие отдельные задачи управления и сопровождения компьютерной системы, такие как программы сжатия дисков, архивирования данных;

– системные обрабатывающие программы — текстовые или графические редакторы, компиляторы, компоновщики, отладчики;

– программы предоставления пользователю дополнительных услуг — специальный вариант пользовательского интерфейса, калькулятор и даже игры;

– библиотеки процедур различного назначения, упрощающие разработку приложений, например, библиотека математических функций, функций ввода-вывода и т. д.

Разделение операционной системы на ядро и модули-приложения обеспечивает легкую расширяемость ОС. Чтобы добавить новую высокоуровневую функцию, достаточно разработать новое приложение, и при этом не требуется модифицировать ответственные функции, образующие ядро системы.

Важным свойством архитектуры ОС, основанной на ядре, является возможность защиты кодов и данных операционной системы за счет выполнения функций ядра в привилегированном режиме.

2.4. Новые функции Windows Server 2022 на базе Windows Server 2019

Во время анонса новой операционной системы компания Microsoft особо подчеркнула инновационные возможности, нацеленные на повышение безопасности. Новшества позволяют реализовать концепцию сервера с защищенным ядром, основанную на сочетании оборудования, микропрограмм и драйверов. Все заявленные функции можно вручную настроить в центре администрирования Windows.

Более того, впервые HTTPS и TLS 1.3 включены по дефолту. Также частью новой ОС стал Secure DNS (DNS поверх HTTPS), который включен еще и в клиентские Windows 10 21H2, а также Windows 11.

Улучшения в сетевом протоколе SMB. С появлением Server 2022 изменилась процедура шифрования для протокола Server Message Block, где с недавнего времени внедрены два надежных алгоритма AES-256. Алгоритм AES-128, как и раньше, будет работать для обеспечения совместимости.

Шифрование SMB настраивается в том числе отдельно для связи между узлами кластера, что сказывается как на CSV, так и на SSD.

Эти функции безопасности теперь также совместимы с SMB Direct, тогда как в предыдущих версиях Windows Server они вызывали снижение производительности при использовании сетевых адаптеров RDMA.

Еще одна новая функция — это возможность сжимать SMB-трафик. В Windows 10, начиная с выпуска 20H2, можно было включить сжатие SMB для xcopy и robocopy с отдельными переключателями для этих программ. В Server 2022 эту функцию теперь можно включить для обмена файлами через Центр администрирования Windows или PowerShell.

Еще одна новая функция для доступа к общим файловым ресурсам — поддержка SMB через QUIC. Протокол QUIC может использоваться как альтернатива TCP, а в сочетании с TLS 1.3 он также может использоваться для замены VPN. Однако эта функция доступна только в Windows Server 2022 Datacenter: Azure Edition.

Раздел конфигурации SMB в центре администрирования Windows не содержит настроек для SMB через QUIC в Server 2022 Datacenter.

В случае «горячего исправления» Microsoft резервирует для Azure еще одну интересную новую функцию. Она позволяет внедрять обновления без перезапуска сервера. Операционная система использует для этого службу автоматического управления Azure.

Помимо новых вариантов гибридных конфигураций (таких как управление локальными серверами через Azure Arc) и расширенной поддержки контейнеров, новая операционная система достигла определенных результатов, соответствующих традиционному использованию ОС.

Вложенная виртуализация для AMD. Еще одна немаловажная функция гарантирует поддержку вложенной виртуализации на чипах производства AMD, которая прежде была доступна лишь для процессоров Intel, начиная с версии ОС Windows Server 2016.

Касаясь поддержки центральных процессоров новых поколений, следует отметить новую версию ОС, которая поддержи-

вает процессоры Intel Ice Lake. На этой платформе система может оперировать объемами оперативной памяти до 48 ТБ и обеспечивать работу до 2048 ядер логического процессора.

Edge в Server Core. С прекращением поддержки IE 15 июня 2022 г. Microsoft Edge заменил устаревший браузер на сервере. Таким образом, Edge включен в Server 2022 и также может опционально использоваться при установке Server Core. Такая конфигурация ранее уже поддерживалась, однако при ручной установке возникали некоторые препятствия.

Улучшение для Storage Spaces Direct. Для работы с гиперконвергентной инфраструктурой все будущие инновации войдут в Azure Stack HCI, однако в Windows Server по-прежнему будут улучшаться существующие функции.

Теперь это отражено в версиях Server 2022, которые не располагают такой функцией, как расширяемые кластеры, но при этом получили новую возможность восстановления для локальных хранилищ. Администраторы могут использовать эту функцию для контроля за ресурсами, необходимыми для восстановления копий данных или активных рабочих нагрузок.

Другие нововведения в области хранения данных. Пока Storage Spaces Direct соединяет хранилище узлов кластера в пул, Storage Spaces оперирует хранилищем лишь одного сервера. В обновленной версии операционной системы для кеширования доступно многоуровневое хранилище с поддержкой высокоскоростных накопителей по типу твердотельных SSD или NVMe.

Наконец, в Server 2022 Microsoft расширила службу миграции хранилища, представленную в выпуске 2019 г. Первоначально он был предназначен для переноса общих файловых ресурсов из устаревших систем на более новые версии Windows Server. Теперь он поддерживает отказоустойчивые кластеры, серверы Samba и NetApp FAS в качестве источников, а также переносит локальных пользователей и группы.

Формы издания. Начиная с версии 2012 г., Microsoft предлагает серверную ОС в двух основных редакциях, которые различаются в первую очередь правами на виртуализацию. Однако, начиная с Server 2016, Datacenter Edition получил эксклюзивные функции, которые отсутствуют в стандартной версии. К ним от-

носятся Shielded VMs, Storage Replica и программно-определяемое хранилище с локальными дисковыми пространствами.

Это различие сохраняется в версии 2022 г., где Standard Edition ограничивается двумя виртуальными экземплярами и включает только урезанную версию Storage Replica, которая ограничена максимальным объемом томов в 2 ТБ.

К двум выпускам теперь присоединяется третий, который называется Windows Server 2022 Datacenter: Azure Edition. Как следует из названия, он предназначен только для работы в облаке Microsoft.

Microsoft Windows Server 2022 Standard. Стандартная версия предназначена для пользователей, которым требуется всего несколько виртуальных машин Windows Server. Помимо установки на физическое оборудование, с серверной лицензией также возможны две виртуальные машины под управлением Windows Server. На виртуальные машины Linux нет ограничений. Почти все функции и роли серверов из Datacentre Edition также включены в стандартную версию.

Microsoft Windows Server 2022 Essentials. Версия Essentials, рассчитанная на 25 пользователей или 50 устройств, является идеальной серверной ОС для малого бизнеса. Она включает в себя многие функции старших выпусков, таких как Windows Admin Center и System Insights, однако ограничена поддержкой одного процессора с максимум 10 ядрами.

Microsoft Windows Server 2022 Datacenter. С выпуском Datacenter пользователи получили максимально возможную гибкость в развертывании серверов и смогли реализовывать крупные, быстро меняющиеся рабочие нагрузки. Количество виртуальных машин неограниченно, и программно определяемое хранилище может быть реализовано с помощью Storage Spaces Direct.

Системные требования

Процессор. На эффективность процессора влияют два ключевых фактора: число ядер и тактовая частота. Для установки Windows Server система должна быть оборудована как минимум одним 64-разрядным процессором с тактовой частотой 1,4 ГГц,

совместимым с набором инструкций x64. В числе прочего должна быть обеспечена поддержка функций безопасности DEP и NX Bit.

Оперативная память. Минимальные требования к оперативной памяти на сервере — не менее 512 МБ. Кроме того, требуется поддержка ECC.

Сетевые адаптеры. Адаптер Ethernet должен обеспечивать скорость не менее 1 Гбита/с. Также сетевые адаптеры должны соответствовать спецификации архитектуры PCI Express.

Дисковое пространство. В конфигурацию сервера должен быть включен жесткий диск объемом не менее 32 ГБ данных для работы Windows Server, в то время как для установки графического интерфейса потребуются дополнительные 4 ГБ.

Windows Server 2022 не привносит новых ролей или функций, но вместе с тем дополняет и вносит коррективы в ряд существующих функций и протоколов, некоторые из которых существенно повышают безопасность сервера.

Тем не менее, продукт от Microsoft остается одним из наиболее востребованных и надежных серверных ОС на рынке. Выбор лицензии и издания Windows Server 2022 следует делать на основе специфики бизнеса, численности штата и имеющегося оборудования.

В основе операционной системы Windows Server 2022 лежит Windows Server 2019 и надежная платформа Windows Server 2016. Она предоставляет множество инновационных возможностей для работы с четырьмя основными областями: гибридное облако, безопасность, платформа приложений и гиперконвергентная инфраструктура (HCI).

Windows Admin Center. Windows Admin Center представляет собой локально развертываемое браузерное приложение для управления серверами, кластерами, гиперконвергентной инфраструктурой и ПК под управлением Windows 10. Он не требует дополнительных затрат, помимо Windows, и готов к использованию в рабочей среде.

Возможности рабочего стола. Поскольку Windows Server 2022 получил развитие из Windows Server 2019, выпуске Long-Term Servicing Channel (LTSC), он включает возможности рабочего стола. Как и в случае с Windows Server 2016, во время настройки операционной системы Windows Server 2022 можно

выбрать установку основных серверных компонентов или установку сервера с возможностями рабочего стола.

Системная аналитика. Системная аналитика — это новая функция, за счет которой в Windows Server реализуется встроенная поддержка локальных возможностей прогнозной аналитики. Эти возможности прогнозирования, каждая из которых поддерживается моделью машинного обучения, локально анализируют системные данные Windows Server, такие как счетчики производительности и события. Системная аналитика позволяет понять, как работают серверы, и помогает сократить эксплуатационные расходы, связанные с реактивным управлением проблемами в развертываниях Windows Server.

Функция совместимости приложений основных серверных компонентов по требованию. Компонент совместимости основных серверных приложений по запросу значительно повышает совместимость приложений, включив подмножество двоичных файлов и компонентов из Windows Server с возможностями рабочего стола. Основные серверные компоненты поддерживаются как можно более экономичными, не добавляя графическую среду Windows Server Desktop Experience, что повышает функциональность и совместимость.

Эта дополнительная функция по требованию доступна в отдельном ISO-файле, ее можно добавлять только в образы и установки основных серверных компонентов Windows с помощью DISM.

Роль транспортного сервера служб развертывания Windows (WDS), добавленная в Server Core. Транспортный сервер содержит только основные сетевые компоненты службы развертывания Windows. Теперь можно использовать основные серверные компоненты с ролью транспортного сервера для создания многоадресных пространств имен, которые передают данные (включая образы операционной системы) с изолированного сервера. Можно использовать его, если требуется предоставить PXE-сервер, который позволяет клиентам выполнять PXE-загрузку и скачивать собственное приложение для установки.

Интеграция служб удаленных рабочих столов с Azure AD. Благодаря интеграции Azure AD (облачная служба для управления удостоверениями и доступом) можно использовать поли-

тики условного доступа, многофакторную проверку подлинности, встроенную проверку подлинности с другими приложениями SaaS (облачная модель предоставления программного обеспечения с помощью Azure AD) и др.

Сеть. Выполнены улучшения основного сетевого стека, такие как быстрое открытие TCP (TFO), автонастройка окна получения, IPv6 и др.

Безопасность. Включена Defender Advanced Threat Protection (Microsoft Defender ATP) — платформа, выполняющая функции поиска, идентификации, приоритизации и реагирования на угрозы безопасности на уровне конечных точек в Защитнике Windows (ATP). Датчики глубокого анализа и ответные действия платформы ATP выявляют атаки на уровне памяти и ядра и реагируют на них путем подавления вредоносных файлов и завершения вредоносных процессов.

Включена новая функция безопасности Windows ATP Exploit Guard — это новый набор возможностей предотвращения вторжений в узел, позволяющий сбалансировать риски безопасности и требования к производительности. Защитник Windows Exploit Guard предназначен для блокировки устройства от широкого спектра векторов атак и блокировки поведения, обычно используемого в атаках с вредоносными программами.

Безопасность программно-конфигурируемых сетей (SDN). Функция безопасности для SDN предоставляет множество возможностей для безопасного выполнения рабочих нагрузок клиентами как в локальной среде, так и в качестве поставщика услуг в облаке.

Поддержка Linux. Теперь Windows Server 2022 поддерживает выполнение систем Ubuntu, Red Hat Enterprise Linux и SUSE Linux Enterprise Server внутри экранированных виртуальных машин при работе в средах со смешанными ОС.

HTTP/2 для более быстрого и безопасного просмотра веб-страниц. Улучшенное объединение подключений исключает сбои при работе в Интернете, а также обеспечивает правильное шифрование веб-сеансов. Обновленный процесс согласования наборов шифров на стороне сервера в HTTP/2 обеспечивает автоматическое устранение сбоев подключений и удобство развертывания.

Хранение. Служба миграции хранилища — это новая технология, которая упрощает перенос серверов в более новую версию Windows Server. Предоставляется графическое средство, которое выполняет инвентаризацию данных на серверах, а затем передает данные и конфигурацию на более новые серверы. Служба миграции хранилища также при необходимости переместит удостоверения старых серверов на новые, чтобы приложениям и пользователям не нужно было ничего изменять.

Платформы приложений

Контейнеры Linux в Windows. Появилась возможность запускать контейнеры Windows и Linux на одном узле контейнеров с помощью одной и той же управляющей программы Docker. Сформировалась разнородная среда узла контейнера, обеспечивающая гибкость для разработчиков приложений.

Встроенная поддержка Kubernetes. В Windows Server 2022 улучшены функции вычислений, сети и хранилища выпусков Semi-Annual Channel, необходимых для реализации поддержки платформы Kubernetes в Windows.

Улучшения контейнеров. Улучшены интегрированные удостоверения, что упростило процесс встроенной проверки подлинности Windows в контейнерах и повысило ее надежность, устранив некоторые ограничения предыдущих выпусков Windows Server.

Улучшенная совместимость приложений. Упрощено создание контейнеров приложений Windows: улучшена совместимость приложений для имеющегося образа windowsservercore. Для приложений с большими зависимостями API теперь существует третий базовый образ: windows.

Уменьшение размера и повышение производительности. Размеры загрузки базового образа контейнера, размер на диске и время запуска были улучшены для ускорения рабочих процессов контейнеров.

Интерфейс администрирования в Windows Admin Center. Был значительно упрощен мониторинг контейнеров, запущенных на компьютере, а также управление отдельными контейнерами с помощью нового расширения для Windows Admin Center.

Улучшения вычислений

Упорядочение на виртуальной машине. Порядок запуска виртуальной машины также улучшен благодаря повышению осведомленности об ОС и приложениях, в результате чего добавлены улучшенные триггеры, когда виртуальная машина считается запущенной перед запуском следующей.

Поддержка памяти класса хранилища для виртуальных машин. Позволяет создавать тома с прямым доступом и форматированием с файловой системой NTFS на энергонезависимых модулях DIMM и предоставлять их виртуальным машинам Hyper-V. Виртуальные машины Hyper-V теперь могут использовать преимущества производительности с низкой задержкой, предоставляемые устройствами памяти класса хранения.

Поддержка постоянной памяти для виртуальных машин Hyper-V. Чтобы использовать высокую пропускную способность и низкую задержку постоянной памяти (также известной как память класса хранения) на виртуальных машинах, теперь ее можно проецировать непосредственно на виртуальные машины. Постоянная память может помочь значительно сократить задержку транзакций базы данных или сократить время восстановления для баз данных с низкой задержкой в памяти при сбое.

Хранилище контейнеров — постоянные тома данных. Контейнеры приложений теперь имеют постоянный доступ к томам.

Зашифрованные сети. Шифрования виртуальных сетей, позволяющие шифровать трафик виртуальной сети между виртуальными машинами, которые обмениваются данными между собой в подсетях с пометкой «Включено шифрование». Для шифрования пакетов с помощью этой возможности также используется протокол DTLS в виртуальной подсети. Протокол DTLS обеспечивает защиту от перехвата, несанкционированных изменений и подделки со стороны любых лиц, имеющих доступ к физической сети.

Подсистема Windows для Linux (WSL). WSL позволяет администраторам сервера использовать имеющиеся средства и сценарии с Linux в Windows Server.

Выводы по главе 2

Сформулирована основная цель администрирования ОС и требования к современным операционным системам. Рассмотрены назначение и основные функции современных операционных систем. Определены сетевые операционные системы, их функциональные компоненты, встроенные сетевые службы и сетевые оболочки. Проанализированы функции операционных систем по управлению ресурсами компьютера, функции операционных систем по управлению ресурсами компьютера, управление файлами и внешними устройствами. Предложена обобщенная структура операционной системы и характеристика оценки возможностей новой MS Windows Server 2022.

Понятия ОС, рассчитанные на обработку больших данных и машинного обучения, определяются Admin Center, новыми возможностями рабочего стола, функциями совместимости приложений основных серверных компонентов по требованию. Акцентировано внимание на службах развертывания Windows, службах удаленных рабочих столов с Azure AD. Рассмотрены контейнеры Linux в Windows, встроенная поддержка Kubernetes и подсистема Windows для Linux.

Вопросы для самоконтроля

1. Какова цель администрирования операционных систем?
2. Каковы требования к современным операционным системам?
3. Каковы назначение и основные функции современных операционных систем?
4. В чем заключаются сетевые операционные системы?
5. Каковы функциональные компоненты сетевой операционной системы?
6. Чем характеризуются встроенные сетевые службы и сетевые оболочки?
7. Каковы функции операционных систем по управлению ресурсами компьютера?
8. В чем заключается управление файлами и внешними устройствами?
9. В чем заключается обобщенная структура операционной системы?

10. Перечислите новшества в MS Windows Server 2022.
11. Перечислите формы издания MS Windows Server 2022.
12. В чем заключается понятие Windows Admin Center?
13. В чем заключаются новые возможности рабочего стола?
14. В чем заключается функция совместимости приложений основных серверных компонентов по требованию?
15. Опишите службы развертывания Windows.
16. Опишите службы удаленных рабочих столов с Azure AD.
17. Опишите контейнеры Linux в Windows.
18. Опишите встроенную поддержку Kubernetes.
19. Опишите подсистему Windows для Linux.

Глава 3

SQL Server и контейнеры

3.1. Концепция контейнеров

В цифровой экономике организации сталкиваются с большими объемами данных (Big Data) из широкого спектра постоянно увеличивающихся наборов данных, размещенных в неконтролируемых источниках данных.

В рассматриваемой операционной системе можно получать ценную информацию практически в реальном времени из всех данных с помощью кластеров больших данных SQL Server 2019, которые обеспечивают полномасштабную среду для работы с большими наборами данных, в том числе с использованием машинного обучения и возможностей искусственного интеллекта.

В Microsoft Server понятие виртуальных машин (ВМ) было известно уже достаточно давно. В качестве новых определены контейнеры.

Контейнеры определяются как виртуализация операционной системы. Некая виртуальная машина состоит из полной операционной системы (гостевой), запущенной в машине хоста. Следовательно, для машины хоста характерно размещение определенного числа виртуальных машин. Всякая виртуальная машина способна работать под управлением иной гостевой операционной системы.

Контейнеры опираются на единственную операционную систему хоста (которая может в свою очередь представлять какую-то виртуальную машину), совместно используя ресурсы ядра. Это делает контейнеры более легковесными, нежели вирту-

альные машины. Хотя контейнеры и разделяют некое единственное ядро операционной системы, они изолированы друг от друга. Основное различие контейнеров и виртуальных машин заключается в том, что виртуальные машины виртуализируют весь компьютер вплоть до аппаратных уровней, а контейнеры — только программные уровни выше уровня операционной системы.

Отличия между контейнерами и виртуальными машинами показаны на рис. 3.1.

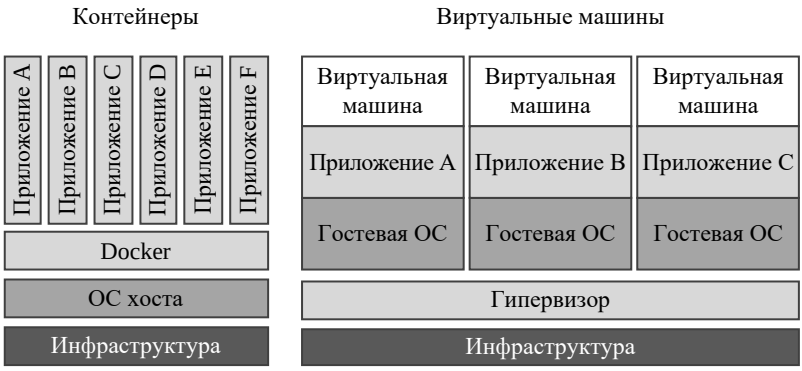

Рис. 3.1. Отличия контейнеров и виртуальных машин

Контейнеры являются абстракцией на уровне прикладного приложения, которые сообща пакуют код и зависимости. Множество контейнеров может исполняться в одной и той же машине и совместно с прочими контейнерами применять ядро ОС хоста, причем каждый исполняется как изолированный процесс в пространстве пользователя. Контейнеры требуют меньше места, чем виртуальные машины (образы контейнеров обычно в размере представлены десятками МБ) и запускаются почти моментально.

Виртуальные машины представляют некий уровень абстракции физического оборудования, который превращает один сервер во множество серверов. Имеющийся гипервизор позволяет множеству ВМ запускаться в отдельной машине. Всякая ВМ содержит какую-то полную копию некоторой операционной системы, одно или более приложений, необходимые двоичные файлы и библиотеки, требуя десятки ГБ.

Docker — это проект с открытым исходным кодом для автоматизации развертывания приложений в виде переносимых автономных контейнеров, выполняемых в облаке или локальной среде.

Контейнеры решают проблему, которую виртуальные машины не позволяют решить сегодня. Виртуальные машины — удивительная технология, позволяющая абстрагировать приложения от базовой аппаратной платформы, но они требуют загрузки и запуска всей операционной системы, для того чтобы приложение могло работать.

Виртуальные машины позволяют приложениям запускаться изолированно друг от друга на хост-системе, и для SQL Server это было отличным решением для сценариев консолидации, даже несмотря на то, что SQL Server допускает размещение нескольких экземпляров на одном компьютере. Контейнеры обеспечивают ту же концепцию изоляции, но они намного легче, чем виртуальные машины. Контейнеры часто считаются отдельным уровнем абстракции от операционной системы.

Контейнеры не заменяют виртуальные машины, а дополняют их. Фактически одна из самых распространенных сред для запуска контейнеров — это виртуальная машина.

Образ контейнера представляет собой двоичный файл, который описывает набор файлов, организованных в файловой системе, и программу, запускаемую из этих файлов.

Контейнер является экземпляром программы, запускаемой изолированным способом в образе контейнера вместе с файловой системой.

Виды контейнеров

Переносимые контейнеры. Контейнеры являются переносимыми, потому что образ контейнера можно запускать везде, где работает Docker, т. е. практически везде, включая компьютеры, функционирующие под управлением ОС Windows, Linux и macOS, а также в облачных системах, поддерживающих эти операционные системы или Kubernetes. Можно взять образ контейнера SQL Server в виде двоичного файла и поместить его в любую из систем, где он будет работать точно так же.

Облегченные контейнеры. Контейнер — это запущенный экземпляр образа контейнера, процесс, представляющий собой приложение, работающее изолированно. Это делает его намного более легким, чем виртуальная машина, предназначенная для размещения приложения. Требования контейнеров к ресурсам также оптимизированы, потому что если запускается более одного контейнера из образа, часть файлов образов (называемых читаемым слоем, или слоем, доступным для чтения) распределяется между контейнерами. Это уменьшит объем ресурсов, необходимый для запуска нескольких экземпляров SQL Server на хост-системе или виртуальной машине.

Согласованные контейнеры. Это один из аспектов контейнеров, который решает много проблем, и особенно для SQL Server. В течение многих лет в организациях сложилась практика, когда администраторы устанавливали отдельные серверы с SQL Server, формируя платформы для тестирования и разработки. Это может вызвать сложности, так как несколько разработчиков, использующих один и тот же SQL Server (при этом им обычно требуется более высокий уровень доступа к SQL Server), могут создать множество проблем для администраторов баз данных.

Контейнеры предоставляют разработчикам согласованный способ использования SQL Server, при этом отказавшись от обязательного совместного использования экземпляра SQL Server. Например, если есть необходимость использования разработчиками определенного образа версии SQL Server вместе с определенной базой данных, можно создать образ контейнера. Поскольку контейнеры переносимы, и SQL Server теперь работает на Linux, можно предоставить один и тот же образ контейнера SQL Server для разработчиков, использующих разные платформы.

Эффективные контейнеры. Контейнеры предоставляют новые эффективные возможности для обновления программного обеспечения, в том числе SQL Server. На рис. 3.2. показана схема работы контейнеров.

Контейнер в середине представляет собой остановленный контейнер SQL Server.

Контейнер слева — это новая версия SQL Server, которая запускается, но указывает на те же системные и пользовательские

базы данных, размещенные в постоянном хранилище (при этом используется концепция томов).

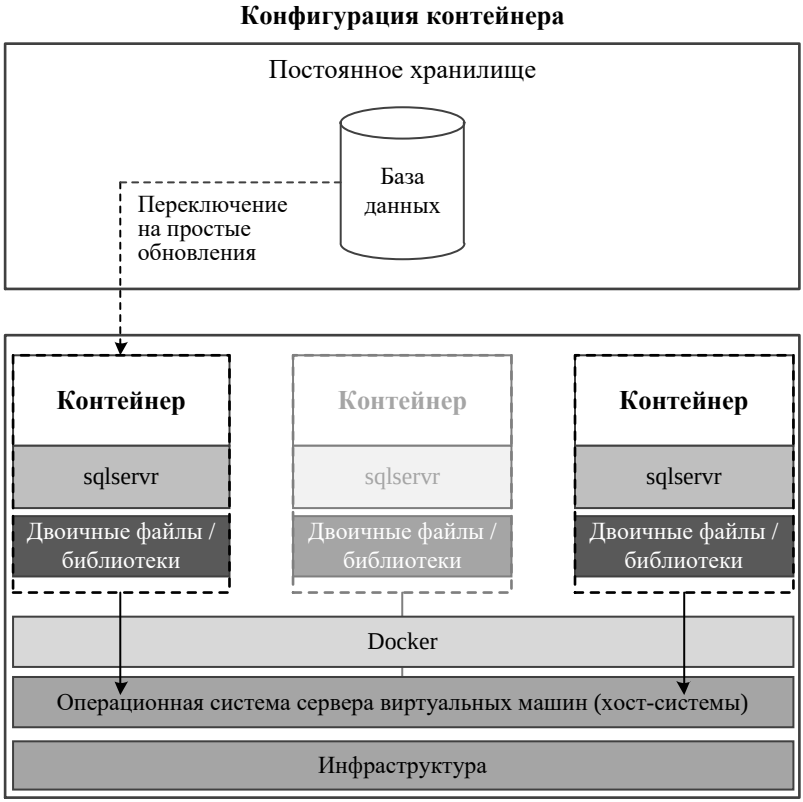

Рис. 3.2. Схема работы контейнеров

Элемент «Двоичные файлы / библиотеки» представляет собой двоичные файлы, необходимые для запуска SQL Server. Они иллюстрируют аспект «облегченности» контейнера, отличающий его от виртуальной машины. Обратим внимание, что если все контейнеры созданы на основе одного и того же образа, эти файлы совместно используются всеми контейнерами, однако на рисунке это не изображено.

Элемент Docker обозначает программное обеспечение, которое используется для запуска и управления контейнерами. Изу-

чим стрелки, проходящие через Docker к операционной системе хост-узла.

Docker не является слоем между контейнером и операционной системой хост-узла. Другими словами, SQL Server не нужно взаимодействовать через некоторый отдельный уровень для выполнения операций ОС ядра. Именно поэтому контейнеры считаются более легкими — они содержат программное обеспечение, напрямую взаимодействующее с операционной системой хост-узла. Уникальной особенностью контейнеров является то, что они работают изолированно друг от друга, отсюда и появился термин «контейнер».

Размещение контейнеров

Контейнеры — это программные модули, которые взаимодействуют с операционной системой хост-узла, которая может находиться на виртуальной машине или непосредственно на аппаратной платформе.

Поскольку программа в контейнере взаимодействует напрямую с операционной системой хост-узла, так же, как и любая другая программа в системе (за исключением того, что контейнеры запускаются специальным изолированным способом), она должна быть скомпилирована и запущена на этой операционной системе.

Еще в версии SQL Server 2017 для Linux представлены образы контейнеров SQL Server на основе Linux, а именно Ubuntu Linux. Почти каждый образ контейнера, существующий в мире, создан на базе операционной системы хост-узла, и абсолютное большинство из них использует Linux. Для хост-узлов, работающих на Linux, запуск контейнера SQL Server на Linux не является проблемой. Контейнеры с SQL Server в Linux могут использоваться независимо от того, установлена система Linux непосредственно на аппаратной платформе или на виртуальной машине.

В случае Windows и macOS ключевой концепцией является Docker как среда выполнения для контейнера. Docker поддерживает контейнеры, работающие в Windows, посредством Docker Desktop для Windows. Любой контейнер, основанный на образе Linux, будет запускаться в контексте виртуальной машины с Linux, называемой DockerDesktopVM.

В последнее время были достигнуты определенные успехи в реализации контейнеров с версией Docker для Windows благодаря поддержке концепции, называемой Linux Containers for Windows (LCOW). Команда Windows описывает эту концепцию как более легкий метод запуска контейнеров Linux в Windows, чем полноценная виртуальная машина. Кроме того, Docker Desktop для Windows может использовать оптимизированный метод с использованием новой подсистемы Windows для Linux. Если можно создать образ контейнера на основе операционной системы Linux, существуют образы контейнеров на основе Windows и macOS.

Docker, как и другие среды выполнения контейнеров, принял концепцию контейнеров и создал платформу и экосистему, которые используются повсеместно, но сам использует возможности операционной системы для поддержки контейнеров (в Linux аналогичные концепции применяются к контейнерам Windows).

3.2. Поддержка контейнеров

Поддержка контейнеров включает в себя следующие основные понятия:

– **пространство имен** — обеспечение механизма, позволяющего запускать процессы изолированно друг от друга;

– **контрольные группы** — предоставление механизма для управления использованием ресурсов для процессов или множества процессов. Контейнеры по умолчанию имеют доступ ко всем вычислительным ресурсам, таким как память и процессор, но контрольные группы предоставляют метод для ограничения использования ресурсов для контейнера;

– **объединенная файловая система** — представление нескольких каталогов как один. Эта концепция является ключевой для минимизации размера контейнеров и поддержки слоев для чтения и для записи. В системе Linux файловая система OverlayFS поддерживает объединенную файловую систему.

Ключевая концепция для контейнеров базируется на следующих понятиях.

Слой для чтения — образ контейнера доступен только для чтения и состоит из набора файлов, размещенных в файловой си-

стеме. Для SQL Server это включает минимум поддерживаемых файлов из образа базовой операционной системы и файлов для SQL Server, включая двоичные файлы и системные базы данных.

Слой для записи — это любые изменения, внесенные в файловую систему контейнера после его запуска. Они могут включать любые изменения в файлах из слоя для чтения или добавление новых файлов. Слой для записи сохраняется в течение всего срока службы контейнера. После удаления контейнера слой для записи также удаляется. Для пользовательских баз данных SQL Server это представляет проблему.

Том — это место постоянного хранилища на хост-узле, которое связано с каталогом в доступном для записи слое контейнера. Для SQL Server обычной практикой является использование тома, связываемого с каталогом в контейнере для хранения баз данных. Тома сохраняются независимо от времени существования контейнера, поэтому если контейнер удален, том все еще существует.

Docker — представляет свой собственный API, называемый libcontainer, абстрагирующий концепции ОС, который поддерживает контейнеры.

Важно отметить, что Docker является примером среды выполнения контейнера — одной из самых популярных в отрасли. Существует среда выполнения контейнера с открытым исходным кодом, которая называется containerd.

Жизненный цикл контейнера

При установке Docker в Linux, Windows или macOS устанавливаются следующие компоненты, которые обеспечивают работу контейнеров:

– **механизм Docker (docker engine)** — состоит из демона Docker (который является «сервисом»), контролирующего все операции по созданию и запуску контейнеров. Механизм Docker поддерживает API для программ, взаимодействующих с механизмом построения и запуска контейнеров;

– **клиент Docker (docker client)** — это приложение, которое использует API механизма Docker для создания и запуска контейнеров. Клиент Docker — это согласованная программа, которая

поддерживает все возможности и ведет себя одинаково в Windows, macOS и Linux;

– **инструмент для управления набором контейнеров Docker (docker compose)** — это приложение, позволяющее создавать и запускать многоконтейнерные приложения.

На рис. 3.3 показан жизненный цикл контейнера.

Рис. 3.3. Жизненный цикл контейнера

Рассмотрим каждый из его элементов более подробно.

Build — команда сборки Docker, которая используется для создания нового образа контейнера. Хотя командой поддерживается SDK, стандартный подход заключается в определении образа для построения с использованием файла с именем Dockerfile.

Push — после создания образа нужно предоставить другим пользователям возможность использовать его для отправки или публикации образа контейнера, в реестре используется команда docker push. Этот реестр может находиться на локальном сервере или в открытом доступе. Одним из наиболее распространенных реестров в публичном доступе является Docker Hub или hub.docker.com.

Pull — любой, кто хочет использовать образ контейнера, должен получить его по запросу, даже если этот образ хранится на локальном сервере. Получить образ контейнера можно с помощью команды docker pull. Механизм Docker (docker engine) будет хранить копию образа локально на хост-узле.

Run — чтобы запустить контейнер на основе образа, используется команда docker run. Если вы запускаете контейнер на основе образа, который еще не был получен по запросу, Docker сначала получает образ, а затем запускает контейнер.

После того как запущен контейнер, появляется возможность управлять им. С помощью клиента Docker можно остановить, запустить, перезапустить и удалить контейнер. Кроме того, клиент Docker позволит управлять образами, в том числе удалять их.

Клиента Docker также можно использовать для мониторинга и управления экосистемой контейнеров: он позволяет полу-

чить списки запущенных и остановленных контейнеров, а также выводить данные статистики и журналов для запущенных и остановленных контейнеров.

Наконец, клиент Docker позволяет взаимодействовать с запущенными контейнерами, копируя файлы в слой для записи на хост-узле и запуская программу, которая существует в файловой системе контейнеров (которая будет выполняться в том же пространстве имен, что и основная программа-контейнер). Эти команды будут очень полезны для контейнеров SQL Server.

Контейнер SQL Server

Образы контейнеров SQL Server содержат необходимые файлы для ядра SQL Server, SQL Server Agent, всех функций, включенных в механизм SQL Server, таких как репликация и средства командной строки SQL Server (sqlcmd и bcp). Когда запускается контейнер, SQL Server уже предварительно установлен. Другими словами, когда вы получаете и запускаете контейнер SQL Server, вы готовы его использовать. Это одно из основных преимуществ использования контейнера. После запуска контейнера установка SQL Server не требуется.

Образ контейнера создается с помощью команды сборки docker build с использованием файла с названием Dockerfile. Чтобы понять, как предварительно установлен контейнер SQL Server, приведем примерное описание команд в Dockerfile для SQL Server:

```
FROM <базовый образ на основе ubuntu или rhel>
LABEL <информация о метке Microsoft>
EXPOSE 1433
COPY <библиотеки и двоичные файлы SQL Server>
RUN ./install.sh
CMD ["/opt/mssql/bin/sqlservr"]
```

Команда `FROM` задает базовый образ ОС, на котором построен образ контейнера SQL Server. Одной из особенностей контейнеров является возможность создавать новые образы на основе других, создавая эффект наложения образов. Рассмотрим, как создать свой собственный образ на основе образа SQL Server (который основан на образе базовой ОС).

Команда EXPOSE позволяет контейнеру SQL Server устанавливать соединения, необходимые приложениям для обмена данными с использованием порта 1433 внутри контейнера. Это важно, поскольку по умолчанию контейнеры изолированы. Часто этот порт отображается на другой порт хост-узла, что позволяет нескольким контейнерам SQL Server работать на одном хост-узле, что приводит обычно к сбою, поскольку два приложения не могут быть привязаны к одному и тому же порту.

Команды COPY и RUN являются лишь частью процесса сборки для копирования всех двоичных файлов SQL Server в файловую систему образа контейнера и установки любых программных зависимостей.

Все эти команды в Dockerfile SQL Server до сих пор являются шагами создания образа контейнера. Когда выполняется команда сборки docker build, каждый из этих операторов используется для построения образа. Оператор CMD сообщает Docker имя программы, запускаемой при запуске контейнера, в нашем случае sqlservr. Это означает, что контейнер SQL Server не работает как «служба» (например, служба systemd в Linux). Интересным является то, что, приложение SQL Server (это работает таким же образом в Windows) создано как программа-демон, а это означает, что оно работает в фоновом режиме, пока не получит сигнал о том, что должно остановиться.

3.3. Запуск контейнера

Основной синтаксис для запуска контейнера SQL Server выполняется с помощью команды docker run (в Linux перед командами запуска контейнера необходимо вводить команду sudo):

```
docker run
-e 'ACCEPT_EULA=Y' -e 'MSSQL_SA_PASSWORD=Sql2017isfast'
-p 1401:1433
-v sqlvolume:/var/opt/mssql
--hostname sql2019latest
--name sql2019latest
-d
mcr.microsoft.com/mssql/rhel/server:2019-latest
```

Параметры `-e` обозначают переменные среды, которые используются для запуска и работы контейнера. В случае с SQL Server необходимо как минимум принять пользовательское соглашение и ввести пароль `sa`. Также в команде могут использоваться другие переменные среды, позволяющие указать версию SQL Server или включить SQL Server Agent. Любая переменная среды, поддерживаемая SQL Server, может использоваться для предварительной настройки установки SQL Server при запуске контейнера.

Параметр `-p 1401:1433` не потребуется, если необходимо запускать только один контейнер SQL Server на одном хост-узле. Если есть несколько экземпляров SQL Server, нужно связать порт 1433 с другим портом. Любое приложение, желающее подключиться к этому контейнеру SQL Server, теперь будет использовать порт 1401 вместо порта по умолчанию.

Параметр `-v sqlvolume:/var/opt/mssql` указывает, какой том следует применять для связывания с каталогом SQL Server, где хранятся базы данных. Это необязательный параметр, однако если есть необходимость сохранения БД независимо от времени жизни контейнера, придется использовать тома для их хранения. Так, тома определенно будут использоваться для любого контейнера SQL Server, используемого в промышленной среде.

Параметр `-hostname sql2019latest` тоже необязательный, но его удобство заключается в том, что указанное имя вычислительного узла, на котором размещен контейнер, станет значением @@SERVERNAME в SQL Server.

Параметр `-name sql2019latest` не является обязательным, но удобен для управления контейнером. Задав для контейнера имя, можно идентифицировать контейнер по имени и управлять им. Например, после запуска этого контейнера можно остановить его, выполнив команду `docker stop sql2019latest`.

Параметр `-D` определяет, что контейнер будет запускаться в фоновом режиме. Обычно этот параметр используется для контейнера SQL Server. Тем не менее хорошим способом отладки является удаление этого параметра, если нет возможности запустить контейнер SQL Server. Это связано с тем, что программа sqlservr запускается из командной строки, поведение по умолчанию — передать содержимое ERRORLOG в стандартный вывод

(stdout), выведенные таким образом данные затем отображаются при запуске контейнера. Также можно использовать команду `docker logs`, чтобы вывести ERRORLOG контейнера SQL Server.

mcr.microsoft.com/mssql/rhel/server:2019-latest — тег образа контейнера, который необходимо запустить.

Одним из аспектов работы контейнера SQL Server является последовательность запуска. Когда программа sqlservr запускается в контейнере, каталог /var/opt/mssql не существует, однако она создает этот каталог, чтобы извлечь системные базы данных из установленных файлов образа контейнера при запуске. Кроме того, sqlservr понимает, как принимать переменные среды и использовать их в качестве параметров запуска, чтобы было возможно принять пользовательское соглашение, указать пароль sa и задать значения других переменных среды.

3.4. SQL Server и Kubernetes

Если контейнеры — новые виртуальные машины, то Kubernetes — новые серверы.

Kubernetes — это портативная расширяемая платформа с открытым исходным кодом для управления контейнеризованными рабочими нагрузками и сервисами, которая облегчает как декларативную настройку, так и автоматизацию. У платформы есть большая, быстро растущая экосистема. Сервисы, поддержка и инструменты Kubernetes широко доступны.

Сокращенное название Kubernetes — K8s — достаточно часто встречается в специализированной литературе.

Рассмотрим **основные термины и объекты K8s.**

Кластер — это основной вычислительный узел для всех программ, работающих на K8s. Обычно хосты, на которых размещены все объекты, называются кластером K8s.

Узел — виртуальная машина, работающая в кластере. Узел будет хостом для запуска модулей, которые имеют контейнеры, в кластере. Как правило, в кластере K8s имеется несколько узлов.

Под (pod) — логическая коллекция контейнеров, работающих на узле в кластере. Под — это модуль развертывания, управления и отработки отказа контейнеров, работающих в кластере.

Служба — как описывает документация Kubernetes, это абстракция, которая определяет логический набор модулей и политику доступа к ним. Для целей SQL Server служба будет выполнять функции балансировщика нагрузки и абстракции внутреннего IP-адреса модуля, на котором размещен SQL Server. Это очень похоже на приемник (прослушивающий узел) в классической модели кластеризации SQL Server. K8s предоставляет концепцию сервиса, встроенного в программное обеспечение K8s, и такие приложения, как SQL Server, могут связываться с ним, так что независимо от того, где именно модуль SQL Server размещен в кластере, приложения всегда могут подключаться к сервису с использованием одного и того же IP-адреса и порта.

Секрет (secret) — объект K8s, который позволяет хранить конфиденциальную информацию. Для SQL Server это очень удобный способ для хранения пароля sa.

Класс хранилищ (storage class) — объект K8s, который представляет хранилище, аналогичное дисковой системе.

Persistent Volume Claim (PVC) — хранилище, резервное копирование которого осуществляется с использованием постоянного тома, который связан с классом хранилищ.

Изучим **общие сведения о внутреннем устройстве K8s.**

Один из аспектов K8s, который необходимо понимать, — это Application Programing Interface (API).

Вся внутренняя часть K8s базируется на **сервере API** — это часть программного обеспечения, которая обрабатывает запросы и выполняет работу. Рассмотрим пример с SQL Server. Примем сервер API как SQL Server. API SQL Server — это T-SQL: приложения могут отправлять команды T-SQL на SQL Server, и он выполняет действие. K8s работает аналогично.

Это означает, что при написании кода можно развертывать и управлять контейнерами в K8s с помощью API или использовать интерфейс командной строки с названием **kubectl**, который взаимодействует с API-интерфейсом K8s. Необходимо программировать API K8s с помощью kubectl и декларативного протокола, использующего файлы YAML.

Поскольку K8s является системой с открытым исходным кодом, можно развернуть кластер различными способами на разных платформах и системах.

Перечислим основные преимущества Kubernetes:

– быстрое развертывание приложений;

– удобное масштабирование развернутых приложений;

– внутреннее самовосстановление;

– нулевой простой при обновлениях приложений.

Установка кластера. В дальнейшем будем использовать созданный кластер для публикации различных сервисов, и поэтому установка будет производиться на VPS. Для кластера потребуется несколько виртуальных машин:

– Master node (CentOS 7, IvCPU, RAM 2 ГБ, HDD 10 ГБ);

– Worker node (CentOS 7, IvCPU, RAM 2 ГБ, HDD 10 ГБ);

– Worker node (CentOS 7, IvCPU, RAM 2 ГБ, HDD 10 ГБ);

– Ingress node (CentOS 7, 2vCPU, RAM 8 ГБ, HDD 10 ГБ).

На всех узлах нашего кластера K8s установлена хост-система CentOS 7. Рассмотрим вариант развертывания с помощью Kubespray — набора ролей Ansible для установки и конфигурирования K8s.

Развертывание K8s. Для начала отключим файл (раздел) подкачки и межсетевой экран на всех узлах кластера, а также сгенерируем и скопируем ключи SSH.

Отключаем SWAP:

```
swapoff -a
```

Отключаем firewall (на учебном стенде это допустимо, но на проде не стоит так делать):

```
firewall-cmd --state
systemctl stop firewalld
systemctl disable firewalld
```

Следующая задача — сгенерировать ключ SSH и скопировать его на все узлы будущего кластера, включая master, где ключ был изначально сгенерирован:

```
ssh-keygen
ssh-copy-id root@<master host IP>
```

Теперь копируем ключ на оставшиеся хосты (команды выполняются на мастер-хосте):

```
ssh-copy-id root@remote_host
```

Далее на master устанавливаем pip и git:

```
curl "https://bootstrap.pypa.io/get-pip.py" -o "get-pip.py"
python get-pip.py
yum install git
```

Если используется старая версия Python, pip можно поставить следующим способом:

```
curl "https://bootstrap.pypa.io/pip/2.7/get-pip.py" -o "get-
pip.py"
python get-pip.py
```

Подготовка платформы завершена. Мы по-прежнему находимся на мастер-хосте и до конца развертывания кластера с него не уйдем. За нас будет ходить Ansible. Для дальнейших действий потребуется репозиторий Kubespray. Клонируем его:

```
git clone https://github.com/kubernetes-sigs/kubespray.git
cd kubespray
```

Устанавливаем зависимости для Kubespray, описанные в файле requirements.txt:

```
pip install --ignore-installed requests= =2.23.0
pip install -r requirements.txt
```

Итак, было получено следующее:

– на ноде master установлен Ansible;

– сгенерирован и скопирован SSh ключ на все узлы кластера;

– установлены необходимые зависимости из файла requirementS.txt.

Конфигурация Kubespray. Чтобы Kubespray понимал, на какие узлы нужно устанавливать K8s, необходимо создать директорию с описанием конфигурации будущего кластера. Потребуется изменить IP-адреса. Ниже приведем пример inventory.ini:

```
master-1.root.local.io ansible_host=192.168.0.3
ip=192.168.0.3
ingress-1.root.local.io ansible_host=192.168.0.6
ip=192.168.0.6
node-1.root.local.io ansible_host=192.168.0.4 ip=192.168.0.4
node-2.root.local.io ansible_host=192.168.0.5 ip=192.168.0.5
```

```
[kube-master]
master-1.root.local.io

[etcd]
master-1.root.local.io

[kube-node]
node-1.root.local.io
node-2.root.local.io
ingress-1.root.local.io
```

Кластер состоит из четырех узлов:

– один master с компонентами Control Plane;

– один Ingress для маршрутизации трафика;

– два Workers, на которых будут запускаться сервисы.

Control Plane (API server, etcd, Sheduler, Controle manager) собран на одной ноде. Обычно в секции `[kube-master]` больше одного узла, а в секции `[etcd]` — более трех. Это связано с тем, что quorum собирается не меньше чем на трех узлах при нечетном общем количестве узлов в etcd. Поскольку речь идет об учебной площадке, этот вариант тоже имеет право на существование. Отредактируем K8s-cluster/K8s-cluster.yml. Нужно поменять network plugin на `flannel` и имя кластера на `root.local`:

```
kube_network_plugin: flannel
cluster_name: root.local
```

Теперь поправим K8s-cluster/K8s-net-flannel.yml, обычный regexp на сеть провайдера VPS (в нашем случае это 192.168.0.0/24, но у вас подсеть может отличаться).

Исправляем:

```
flannel_interface_regexp: '192\.168\.0\.\d{1,9}'
```

Сборка кластера. Поскольку inventory уже подготовлен, остается только запустить playbook на исполнение и 15–20 минут подождать, пока соберется кластер. Если не получится соединение SSH, и сборка прервется, ее можно будет запустить повторно — это не вызовет ошибок:

```
ansible-playbook -u root -i inventory/inventory.ini
cluster.yml -b --diff
```

Проверяем собранный кластер:

```
[root@master-1 kubespray] kubectl  get      nodes
NAME                      STATUS   ROLES    AGE
ingress-1.root.local.io   Ready    <none>   2d1h
master-1.root.local.io    Ready    master   2d1h
node-1.root.local.io      Ready    <none>   2d1h
node-2.root.local.io      Ready    <none>   2d1h
```

Последний штрих — добавляем роль нашей ноде с ingress:

```
[root@master-1 kubespray] kubectl label node ingress-1.root.lo-
cal.io node-role.kubernetes.io/ingress=
```

Если повторно посмотреть на ingress, появится роль:

```
root@master-1 kubespray]  kubectl  get       nodes
NAME                      STATUS   ROLES     AGE
ingress-1.root.local.io   Ready    ingress   2d1h
master-1.root.local.io    Ready    master    2d1h
node-1.root.local.io      Ready    <none>    2d1h
node-2.root.local.io      Ready    <none>    2d1h
```

Выводы по главе 3

Для решения задач цифровой экономки Российской Федерации определена новая модель данных — кластеры больших данных SQL Server 2019. Проанализированы отличия между контейнерами и виртуальными машинами в концепции Docker. Определены образ, концепция, размещение контейнеров и их поддержка. Продемонстрированы ключевые аспекты концепции для контейнеров и их жизненный цикл. Применительно к реляционным базам данных, определен контейнер SQL Server.

Рассмотрены действия запуска контейнера. Подробно рассмотрены объекты K8s, установка кластера и его сборка.

Вопросы для самоконтроля

1. В чем заключается понятие кластеров больших данных SQL Server 2019?
2. Каковы отличия между контейнерами и виртуальными машинами?
3. В чем заключается понятие Docker?

4. В чем заключается образ контейнера?
5. В чем заключается концепция контейнеров?
6. В чем заключается размещение контейнеров?
7. В чем заключается поддержка контейнеров?
8. В чем заключается ключевая концепция для контейнеров?
9. В чем заключается жизненный цикл контейнера?
10. В чем заключается понятие контейнера SQL Server?
11. Назовите технологию запуска контейнера.
12. Какие новые функции образуются с помощью SQL Server и Kubernetes?
13. Назовите основные термины и объекты K8s.
14. Какие вы знаете сведения о внутреннем устройстве K8s?
15. Перечислите этапы установки кластера.
16. Перечислите этапы сборки кластера.

Глава 4

Виртуализация данных

4.1. Поддерживаемые продукты и службы SQL

Один из основных сценариев для SQL Server 2019 — это возможность выполнять виртуализацию данных. Такой процесс позволяет сохранить данные в исходном расположении. Можно виртуализировать данные в экземпляре SQL Server и запрашивать их так же, как любую другую таблицу в SQL Server, это уменьшает необходимость в процессах извлечения, преобразования и загрузки. Реализовать это можно с помощью соединителей PolyBase.

PolyBase позволяет экземпляру SQL Server запрашивать данные с помощью T-SQL непосредственно из SQL Server, Oracle, Teradata, MongoDB, кластеров Hadoop, Cosmos DB и S3-совместимых хранилищ объектов без необходимости устанавливать клиентское программное обеспечение для подключения. Можно также использовать универсальный соединитель ODBC для подключения к дополнительным поставщикам с помощью сторонних драйверов ODBC. PolyBase позволяет с помощью запросов T-SQL объединить данные из внешних источников с данными из реляционных таблиц в экземпляре SQL Server.

Чаще всего виртуализация данных с помощью PolyBase используется для того, чтобы оставить данные в исходном расположении и формате. Вы можете виртуализировать внешние данные в экземпляре SQL Server и запрашивать их на месте так же, как любую другую таблицу в SQL Server. Это минимизирует необходимость использовать процессы ETL для перемещения данных. Такой сценарий виртуализации можно реализовать с помощью соединителей PolyBase.

PolyBase предоставляет одинаковые функции для следующих продуктов SQL от корпорации Microsoft:

– SQL Server 2016 (13.x) и более поздние версии (Windows);

– SQL Server 2019 (15.x) и более поздние версии (Windows и Linux);

– SQL Server Analytics Platform System (PDW);

– Azure Synapse Analytics.

Виртуализация данных с помощью функции PolyBase доступна для Управляемого экземпляра SQL Azure, который ограничен запросами внешних данных, хранящихся в файлах в Azure Data Lake Storage (ADLS) второго поколения и хранилище BLOB-объектов Azure.

Структура и свойства PolyBase в SQL Server 2022:

1) S3-совместимое хранилище объектов. SQL Server 2022 (16.x) добавляет новый соединитель, совместимое с S3 хранилище объектов с помощью REST API S3. Возможно использовать и OPENROWSETEXTERNAL TABLES для запроса файлов данных в хранилище объектов, совместимом с S3;

2) некоторые соединители отделены от служб PolyBase. Соединители S3-совместимого хранилища объектов, а также ADSL второго поколения и хранилища BLOB-объектов Azure больше не зависят от служб PolyBase. Они по-прежнему должны работать для поддержки подключения к Oracle, Teradata, MongoDB и универсальному подключению ODBC. Компонент PolyBase должен быть установлен на экземпляре SQL Server;

3) пример файла Parquet. PolyBase теперь может запрашивать данные из файлов Parquet, хранящихся в S3-совместимом хранилище объектов;

4) формат разностной таблицы. PolyBase теперь может запрашивать (только для чтения) данные из формата разностной таблицы, находящиеся в хранилище объектов, совместимом с S3, учетной записи хранения Azure второй версии и Azure Data Lake Storage второго поколения;

5) создание внешней таблицы как выбор (CETAS). PolyBase теперь может использовать CETAS для создания внешней таблицы и последующего параллельного экспорта результатов инструкции Transact-SQL SELECT в Azure Data Lake Storage второго поколения, учетной записи хранения Azure второй версии и хранилища объектов, совместимого с S3.

Соединители PolyBase

Компонент PolyBase обеспечивает подключение к внешним источникам данных (табл. 1).

Таблица 1

Режимы подключения к внешним источникам данных с помощью PolyBase

Внешние источники данных	SQL Server 2016–2019 с PolyBase	SQL Server 2022 (16.x) с PolyBase	APS PDW	Azure Synapse Analytics
Oracle, MongoDB, Teradata	Чтение	Чтение	Нет	Нет
Универсальные данные ODBC	Чтение (только Windows)	Чтение (только Windows)	Нет	Нет
Хранилище Azure	Чтение/запись	Чтение/запись	Чтение/запись	Чтение/запись
Hadoop	Чтение/запись	Нет	Чтение/запись	Нет
SQL Server	Чтение	Чтение	Нет	Нет
S3-совместимое хранилище объектов	Нет	Чтение/запись	Нет	Нет

В SQL Server 2016 (13.x) появилась поддержка PolyBase с поддержкой подключений к Hadoop и хранилище BLOB-объектов Azure.

В SQL Server 2019 (15.x) представлены дополнительные соединители, в том числе для SQL Server, Oracle, Teradata и MongoDB, накопительный пакет обновления 19 ввел поддержку Oracle TNS.

SQL Server 2022 (16.x) не поддерживает Hadoop, но в нем появился соединитель хранилища, совместимый с S3, накопительный пакет обновления 2 ввел поддержку Oracle TNS.

В число внешних соединителей PolyBase входят:

– SQL Server;
– Oracle;
– Teradata;
– MongoDB;
– Hadoop;
– S3-совместимое хранилище объектов.

Чтобы использовать PolyBase, в экземпляре SQL Server сделаем следующее:

1) установим PolyBase в Windows или в Linux;

2) начиная с SQL Server 2019 (15.x), может потребоваться включить компонент PolyBase в sp_configure;

3) создадим внешний источник данных;

4) создадим внешнюю таблицу.

Запросы T-SQL на основе PolyBase также можно использовать для импорта данных из хранилища BLOB-объектов Azure и экспорта данных в него. Кроме того, PolyBase позволяет Azure Synapse Analytics импортировать данные из Хранилища BLOB-объектов Azure и Azure Data Lake Store, а также экспортировать в них данные.

PolyBase позволяет объединять данные из экземпляра SQL Server с внешними данными. Прежде чем PolyBase объединит данные во внешних источниках, можно выполнить одно из следующих действий:

– передать часть данных, чтобы все они находились в одном месте;

– запросить данные из двух источников, а затем написать пользовательскую логику запроса для объединения и интеграции данных на уровне клиента.

PolyBase позволяет легко объединять данные, используя Transact-SQL и не требует установки дополнительного программного обеспечения в среде Hadoop. При запросе внешних данных используется такой же синтаксис T-SQL, как и при запросе таблицы базы данных. Все вспомогательные действия, реализуемые PolyBase, выполняются прозрачно. Автору запроса не требуется знать, как работает внешний источник.

PolyBase поддерживает следующие сценарии в SQL Server.

1. **Запрос данных, хранящихся в Хранилище BLOB-объектов Azure.** Хранилище BLOB-объектов Azure — удобное место хранения данных для использования службами Azure.

2. **Запрос данных, хранящихся в Hadoop, из экземпляра SQL Server или PDW.** Пользователи хранят данные в более экономичных распределенных и масштабируемых системах, таких как Hadoop.

3. **Импорт данных из Hadoop, хранилища BLOB-объектов Azure или Azure Data Lake Store.** Используется скорость технологии Microsoft SQL columnstore и возможности анализа при импорте данных из Hadoop, хранилища BLOB-объектов Azure или Azure Data Lake Store в реляционные таблицы. Отсутствует необходимость в отдельном средстве ETL или импорта.

4. **Экспорт данных в Hadoop, хранилище BLOB-объектов Azure или Azure Data Lake Store.** Позволяет создать экономичное хранилище и обеспечить его подключение к сети для удобного доступа к данным.

5. **Интеграция со средствами бизнес-аналитики.** Можно использовать PolyBase со средствами бизнес-аналитики и стеком технологий анализа Microsoft, а также применять любые сторонние средства, совместимые с SQL Server.

6. **Принудительная отправка вычислений в Hadoop** (Применимо только к SQL Server 2016 (13.x), SQL Server 2017 (14.x) и SQL Server 2019 (15.x)). PolyBase отправляет некоторые вычисления во внешний источник, чтобы оптимизировать запрос в целом. Оптимизатор запросов принимает решение принудительно отправить вычисления в Hadoop, если это улучшит производительность запросов, для чего он использует статистику из внешних таблиц. При включении вычислений создаются задания MapReduce и применяются распределенные вычислительные ресурсы Hadoop.

7. **Масштабирование вычислительных ресурсов** (Применимо только к SQL Server 2016 (13.x), SQL Server 2017 (14.x) и SQL Server 2019 (15.x)). Для повышения производительности запросов можно использовать группы горизонтального увеличения масштаба PolyBase в SQL Server. Это обеспечивает параллельную передачу данных между экземплярами SQL Server и узлами Hadoop, а также добавляет вычислительные ресурсы для работы с внешними данными.

Начиная с SQL Server 2022 (16.x), Hortonworks Data Platform (HDP) и Cloudera Distributed Hadoop (CDH) больше не поддерживаются. Из-за этих изменений перед миграцией на SQL Server 2022 (16.x) необходимо вручную удалить внешние источники данных PolyBase, созданные в предыдущих версиях SQL Server, которые используют службу хранилища Azure (переменная TYPE

имеет значение HADOOP). Для удаления внешних источников данных также требуется удалить связанные объекты базы данных, такие как учетные данные для базы данных и внешние таблицы.

Чтобы начать использовать компонент PolyBase, его необходимо установить в Windows или Linux и включить, если требуется, в sp_configure.

4.2. Запуск мастера внешних таблиц

Подключимся к основному экземпляру по IP-адресу и номеру порта конечной точки sql-server-master, получив их с помощью команды `azdata cluster endpoints list`. Развернем узел **Базы данных** в обозревателе объектов. Затем выберем одну из баз данных, где необходимо виртуализировать данные из существующего экземпляра SQL Server. Щелкнем правой кнопкой мыши базу данных и выберем **Создать внешнюю таблицу** (**Create External Table**), чтобы запустить мастер виртуализации данных (рис. 4.1). Также можно запустить его в палитре команд, для этого нажмем клавиши CTRL+SHIFT+P в Windows или CMD+SHIFT+P на Mac.

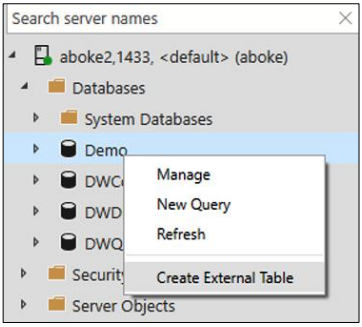

Рис. 4.1. Запуск мастера внешних таблиц

Выбор источника данных. Если мастер запущен в одной из баз данных, раскрывающийся список назначения заполняется автоматически. На этой странице также можно ввести или изменить базу данных назначения (рис. 4.2). Мастер поддерживает

внешние источники данных SQL Server, Oracle, MongoDB и Teradata.

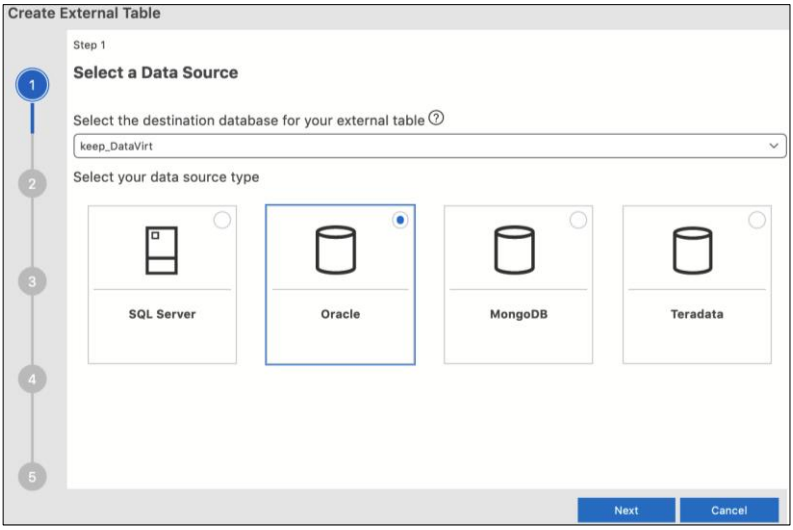

Рис. 4.2. Стартовая страница создания внешней таблицы

Чтобы продолжить, нужно выбрать **Далее** (**Next**).

Создание главного ключа базы данных. На этом шаге нужно создать главный ключ базы данных (рис. 4.3.), что является обязательным условием. Главный ключ позволяет защитить учетные данные, используемые внешним источником. Введем надежный пароль для главного ключа, а также создадим резервную копию главного ключа с помощью команды **BACKUP MASTER KEY**. Сохраним ее в надежном месте вне системы. Если уже есть главный ключ, то этот шаг будет автоматически пропущен.

Указание учетных данных внешнего источника. На этом шаге вводится внешний источник и его учетные данные, чтобы создать объект внешнего источника данных. Учетные данные необходимы для подключения объекта базы данных к источнику. Введем имя внешнего источника данных. Например, Test. Укажем сведения о подключении к внешнему источнику данных SQL Server. Введем **Имя сервера** и **Имя базы данных**, в которой нужно создать внешний источник.

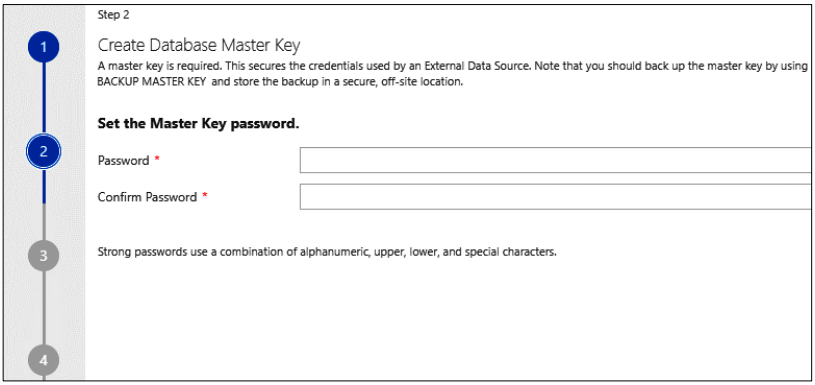

Рис. 4.3. Создание главного ключа базы данных

Следующий шаг — настройка учетных данных. Введем имя пользователя, которое будет использоваться в качестве учетных данных в области базы. Оно позволяет надежно хранить данные для входа в созданный внешний источник. Введем имя пользователя и пароль для подключения к источнику данных (рис. 4.4).

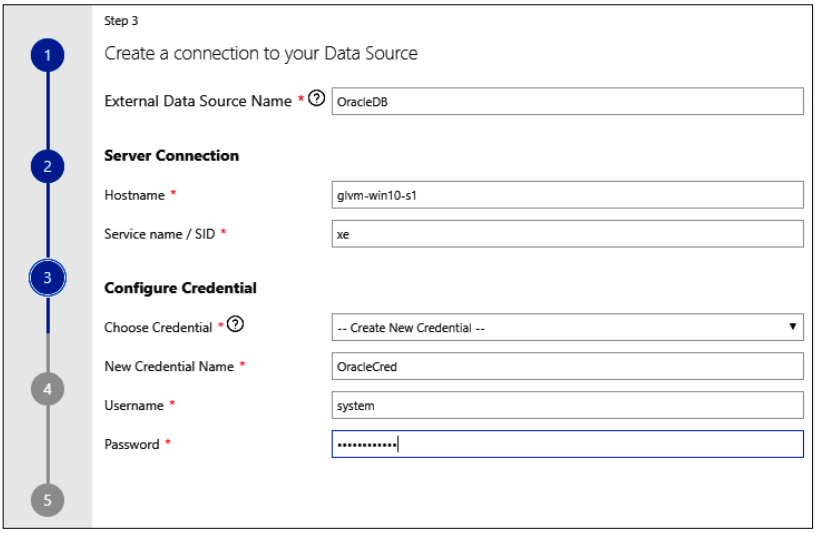

Рис. 4.4. Настройка учетных данных

Создание внешних представлений для таблиц данных. На следующей странице выберем таблицы, для которых нужно создать внешние представления (рис. 4.5). При выборе родительских баз данных также включаются дочерние таблицы. После выбора таблиц справа появится таблица сопоставления. Здесь можно внести изменения в типы, а также изменить имя выбранной внешней таблицы.

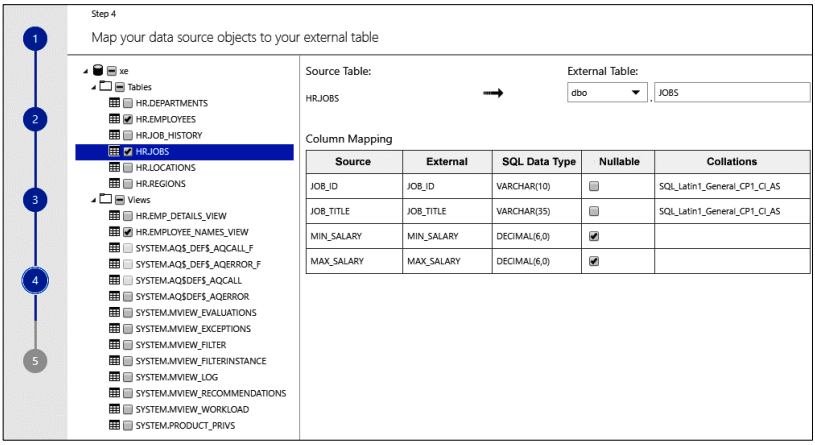

Рис. 4.5. Выбор таблиц,
для которых необходимы внешние представления

Чтобы изменить представление сопоставления, нужно дважды щелкнуть другую выбранную таблицу.

Инструмент внешних таблиц не поддерживает тип фотографии. При создании внешнего представления, содержащего тип фотографии, после создания таблицы отобразится ошибка, однако таблица все равно создастся.

Сводка. На этом шаге отображается сводка выбранных значений, которая содержит имя учетных данных в области базы и объекты внешнего источника данных, созданные в целевой базе (рис. 4.6).

Выберем **Создать скрипт**, чтобы сгенерировать на T-SQL синтаксис для создания внешнего источника данных. Выберем **Создать**, чтобы сформировать объект внешнего источника дан-

ных. Тогда в базе данных назначения будет создан объект внешнего источника данных (рис. 4.7.).

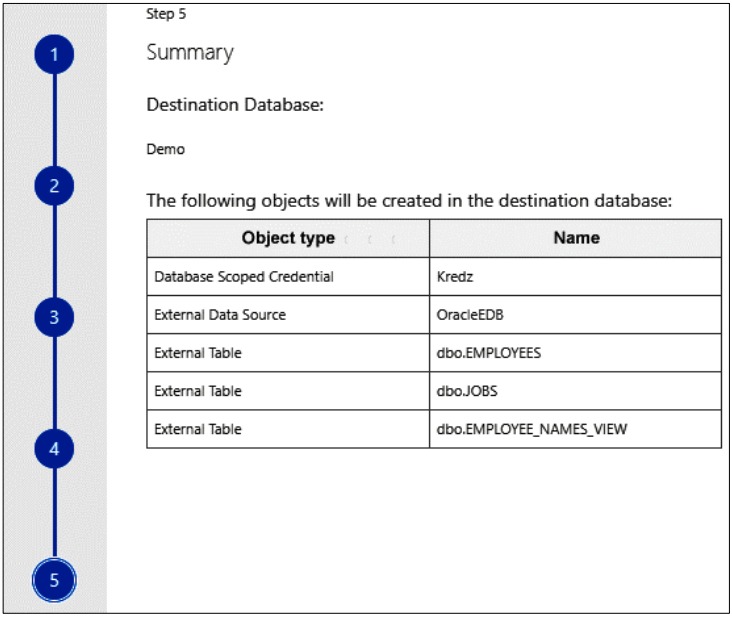

Рис. 4.6. Сводка выбранных значений

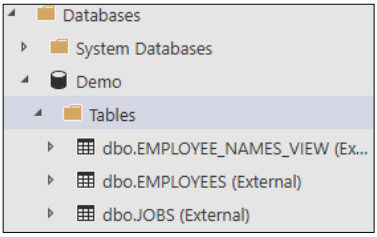

Рис. 4.7. Создание объекта внешнего источника данных

При выборе **Создать скрипт** будет сформирован запрос T-SQL для создания объекта внешнего источника данных (рис. 4.8).

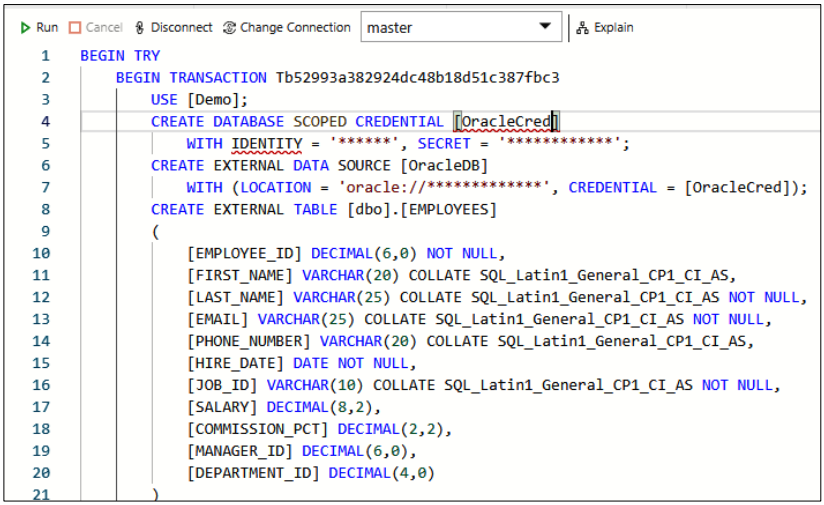

Рис. 4.8. Сформированный запрос T-SQL для создания объекта внешнего источника данных

Кнопка **Создать скрипт** должна отображаться только на последней странице мастера. Сейчас же она отображается на всех.

Выводы по главе 4

Определен сценарий для SQL Server 2019, обеспечивающий виртуализацию данных. Такой процесс позволяет сохранить данные в исходном расположении.

Показано, что можно виртуализировать данные в экземпляре SQL Server и запрашивать их так же, как любую другую таблицу в SQL Server. Это уменьшает необходимость в процессах извлечения, преобразования и загрузки. Реализовать это можно с помощью соединителей PolyBase.

Рассмотрены основные сценарии для SQL Server 2019 с возможностью выполнять виртуализацию данных, применяя соединители PolyBase. Определены поддерживаемые продукты и службы SQL и усовершенствования PolyBase в SQL Server 2022. Рассмотрены облачные технологии (Azure) и интеграция с этой средой.

Перечислены варианты использования PolyBase. Раскрыта возможность обращаться к базе данных Big Data Hadoop.

Разобраны варианты запуск мастера внешних таблиц, создание главного ключа базы данных и сопоставление внешних таблиц данных.

Вопросы для самоконтроля

1. Какие основные сценарии для SQL Server 2019 вы знаете?
2. Каково назначение соединителей PolyBase?
3. Какие поддерживаемые продукты и службы SQL вы знаете?
4. В чем заключаются усовершенствования PolyBase в SQL Server 2022?
5. В чем заключаются понятие Azure и интеграция с Azure?
6. В чем заключаются варианты использования PolyBase?
7. В чем заключается цель принудительной отправки вычислений в Hadoop?
8. В чем заключаются этапы запуска мастера внешних таблиц?
9. В чем заключаются этапы создания главного ключа базы данных?
10. В чем заключаются этапы создания внешних представлений для таблиц данных?

Глава 5

Кластеры больших данных в SQL Server

5.1. Технологии, поддерживающие кластеры больших данных SQL Server

Кластеры больших данных SQL Server (SQL Server Big Data Clusters) — решение, объединяющее три основные технологии:

– *SQL Server* — центр доступа к данным в кластере. Это полная версия SQL Server, работающая в контейнере на основе образа ОС Linux;

– *большие данные* — технологии больших данных, такие как HDFS и Spark;

– *кластер* — кластер Kubernetes для развертывания и запуска различных контейнеров, чтобы обеспечить единую, полную, целостную систему.

Большие данные — это любые данные, которые невозможно обработать за приемлемое время с помощью имеющихся систем. Для SQL Server это означает, что в организации могут иметься данные, для хранения которых не подходит система управления реляционными базами данных (Relational Database Management System, RDBMS), такая как SQL Server. Подобная ситуация может быть следствием многих разнообразных причин, в числе которых можно назвать объем, структуру, происхождение данных и сложность их преобразования в реляционные таблицы.

Кластер больших данных (Big Data Cluster, BDC) SQL Server существует как продукт, являющийся частью другого продукта. Это связано с тем, что при развертывании BDC становится до-

ступным следующее множество ценных инструментов и функциональных возможностей.

SQL Server 2019. BDC поставляется с экземпляром SQL Server 2019, работающим в контейнере, использующем образ ОС Linux. Это означает, что все возможности SQL Server 2019 для Linux также включены в BDC. В их число входят аутентификация Active Directory и высокая доступность с поддержкой групп доступности Always On (Always On Availability Groups).

Polybase. Компонент Polybase для SQL Server с BDC устанавливается и включается автоматически. Это означает, что вы получаете встроенные коннекторы для SQL Server, Oracle, Teradata, MongoDB и Hadoop. Кроме того, BDC поставляется со специальными коннекторами для оптимизированного доступа к файлам HDFS и кешам данных в кластере. Более того, даже несмотря на то, что SQL Server 2019 в Linux не поддерживает группы масштабирования Polybase (Polybase Scale-Out Groups), BDC включает реализацию групп масштабирования Polybase с использованием концепции, называемой пулом вычислений (Compute Pool).

Распределенная файловая система Hadoop (HDFS). Во время использования решения BDC, оно развернет кластер хранения данных HDFS при помощи набора инструментов Apache Hadoop (Apache Hadoop — свободно распространяемый набор утилит, библиотек и фреймворк с открытым исходным кодом). В распоряжении будет несколько разных способов доступа к файлам, хранящимся в кластере HDFS в BDC, включая способ, использующий Polybase для обеспечения возможности работы с данными в SQL Server. Также есть метод для подключения собственного внешнего хранилища HDFS к локальному хранилищу HDFS в BDC, многоуровневую концепцию хранения HDFS (HDFS Tiering).

Spark. При развертывании BDC устанавливается фреймворк Apache Spark, предоставляющий еще один способ для анализа и обработки данных. Взаимодействие со Spark будет осуществляться посредством Spark Jobs, которые будут получать данные, размещенные внутри кластера.

Кеш данных. Предоставляется специализированным набором экземпляров SQL Server, оптимизированных для хранения

результатов запросов к внешним источникам данных Polybase. Нужно представить сценарий, в котором необходимо сохранить набор результатов, обновляемых еженедельно, для целей отчетности. Эти результаты могут быть получены из запросов Polybase с использованием множества различных источников данных, и кеш данных в BDC является идеальным решением для этого, так как реализуется в компоненте, называемом пулом данных (Data Pool).

Инструменты и сервисы. Для облегчения развертывания, использования и управления BDC предоставляется набор инструментов, доступных как часть решения. Инструмент Azure Data Studio станет ключевой частью общего решения BDC, включая поддержку записных книжек (Notebooks).

Кроме того, развертывается набор контейнеров в качестве сервисов, которые помогают координировать и управлять BDC. Эти сервисы названы контроллером (Controller).

Конечные точки. Потребуется возможность подключения к BDC для всех типов задач, поэтому предоставляется ряд конечных точек сервисов. В их число входят конечные точки для подключения к SQL Server, HDFS и Spark, а также несколько сервисов управления и мониторинга.

Развертывание приложений. Кластеры больших данных SQL Server позволяют исполнять программный код посредством операторов T-SQL и Spark Jobs. Службы машинного обучения SQL Server (SQL Server Machine Learning Services) и платформа расширяемости (включая языковые расширения) также позволяют запускать код на R, Python и Java, интегрированный с SQL Server. Поскольку BDC развертывается в кластере K8s, предоставляется удобный способ развертывания приложений в BDC, а также открытый интерфейс для взаимодействия с этими приложениями и обеспечить приложению доступ к источникам данных, подключенным к BDC, таким как таблицы SQL Server и внешние таблицы.

Таким образом, BDC предоставляет концепцию развертывания приложений (Application Deployment) для приложений, написанных на R, Python, MLeap и SSIS. Развертывание приложений является ключевой концепцией использования BDC в качестве комплексной платформы машинного обучения.

Машинное обучение. Одним из решений, предоставляемых BDC, является комплексная платформа для машинного обучения. Это становится возможным благодаря таким элементам BDC, как:

– службы машинного обучения SQL Server;
– SparkML;
– MLeap;
– пакеты машинного обучения;
– развертывание приложений.

Развертывание кластеров больших данных. Все программное обеспечение в BDC развертывается в виде контейнеров в кластере Kubernetes. При работе с BDC предполагается, что развертывается собственный кластер Kubernetes. Также опционально предоставляются инструменты, помогающие развернуть K8s.

Рассмотрим планирование развертывания BDC.

Первое решение, которое необходимо принять при развертывании BDC, — это выбор варианта размещения Kubernetes (K8s). BDC поддерживает развертывание на K8s в общедоступном облачном провайдере **Azure Kubernetes Service** (AKS), или на сервере Linux, или на виртуальной машине, где развернут K8s (например, если K8s самостоятельно развернут с помощью kubeadm). Вероятно, что список других известных поставщиков K8s, которые будут поддерживаться BDC, будет расширяться с выходом SQL Server 2019 и более поздних версий и будет включать Azure Stack, Red Hat OpenShift и другие платформы. На настоящий момент времени можно развернуть BDC при развертывании K8s на Windows Server, но для этого сценария потребуются виртуальные машины Linux, работающие на Windows Server.

Инструменты для развертывания BDC создадут серию объектов pod с контейнерами (в большинстве случаев они будут иметь несколько контейнеров) в K8s для поддержки системы BDC.

Как только определен вариант размещения K8s, можно либо самостоятельно развернуть K8s, либо использовать сценарии, которые созданы для совместного развертывания K8s и BDC.

Для любого варианта основным требованием для развертывания BDC в среде dev/test являются виртуальная машина Linux или компьютер (в случае развертывания решения на AKS нужно

выбрать размер виртуальной машины), удовлетворяющие следующим требованиям к ресурсам:

– 64 ГБ оперативного запоминающего устройства;

– 8 процессоров (могут быть логическими объектами);

– для AKS — размер виртуальной машины Azure, поддерживающей не менее 24 дисков;

– если планируется развернуть более одного узла BDC, каждый узел должен будет соответствовать этим требованиям к ресурсам.

В предоставленных сценариях и записных книжках по умолчанию выбирается размер виртуальной машины Azure: Standard_L8s_v2, но если выбрать виртуальную машину Azure для AKS с 64 ГБ, 8 процессорами и 24 дисками, все сценарии должны работать.

Для нашего варианта развертывания будем использовать AKS и предоставленный сценарий, который развернет кластер AKS и BDC за один шаг.

5.2. Выбор клиентского приложения и загрузка необходимого инструментария

После определения стратегии K8s нужны инструменты для развертывания BDC. Очень важно убедиться, что на клиентском узле установлены все необходимые инструменты, прежде чем пытаться развернуть BDC. Необходимо установить Windows 10 на виртуальной машине Azure и использовать все ее ресурсы для установки инструментария BDC.

Этот инструментарий включает в себя следующие компоненты.

Python — является ключевым компонентом, используемым несколькими различными инструментами, и доступен на всех платформах ОС. Инструмент azdata, необходимый для установки BDC, написан на Python. Python необходим, поскольку будет использоваться сценарий Python для развертывания AKS и BDC за один шаг.

Kubectl — это инструмент, специально разработанный для отправки запросов на сервер API K8s, программный интерфейс

для Kubernetes. Необходимо установить kubectl на компьютере, работающем под Windows.

Azdata — инструмент, известный под названием mssqlctl в ранних версиях SQL Server 2019, предназначавшихся для предварительного пользовательского тестирования. Он крайне важен для развертывания и управления BDC, написан на Python, и его можно рассматривать как kubectl для BDC.

Чтобы убедиться в правильности установки инструмента **azdata**, запускаем azdata из командной строки и видим, как выглядит его интерфейс. Результаты показаны на рис. 5.1.

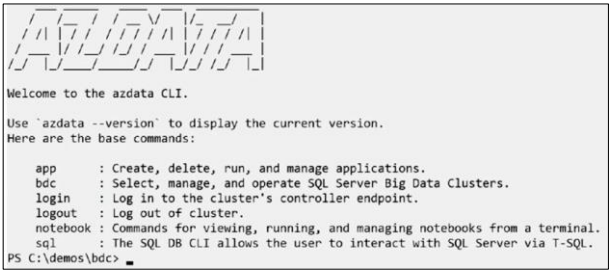

Рис. 5.1. Интерфейс командной строки azdata

Azure Data Studio (ADS) — кроссплатформенный инструмент с открытым исходным кодом, который можно использовать для выполнения запросов, развертывания, управления и навигации по данным в BDC. Хотя SQL Server Management Studio (SSMS) можно использовать для подключения к главному экземпляру SQL Server в BDC, ADS имеет функции и расширения, специально предназначенные для BDC, включая поддержку записных книжек (Notebooks).

Существуют следующие способы развертывания:

– одношаговый способ для развертывания AKS и BDC с использованием Python, сценарий которого можно найти на сайте Microsoft[1];

[1] *Используйте* скрипт Python для развертывания кластера больших данных SQL Server в службе Azure Kubernetes (AKS) / Microsoft. URL: https://docs.microsoft.com/en-us/sql/big-data-cluster/quickstart-big-data-cluster-deploy?view=sql- server-ver15 (дата обращения: 03.03.2023).

– одношаговый сценарий оболочки Bash для развертывания K8s и BDC в кластере K8s с использованием kubeadm[1];

– создание своего собственного кластера AKS или K8s с последующим развертыванием BDC с помощью инструмента azdata[2];

– использование Azure Data Studio для развертывания BDC вместе с новым кластером AKS в существующем кластере AKS или в существующем кластере K8s, который был развернут с помощью kubeadm.

На рис. 5.2 показано, каким образом выбрать вариант развертывания BDC в ADS.

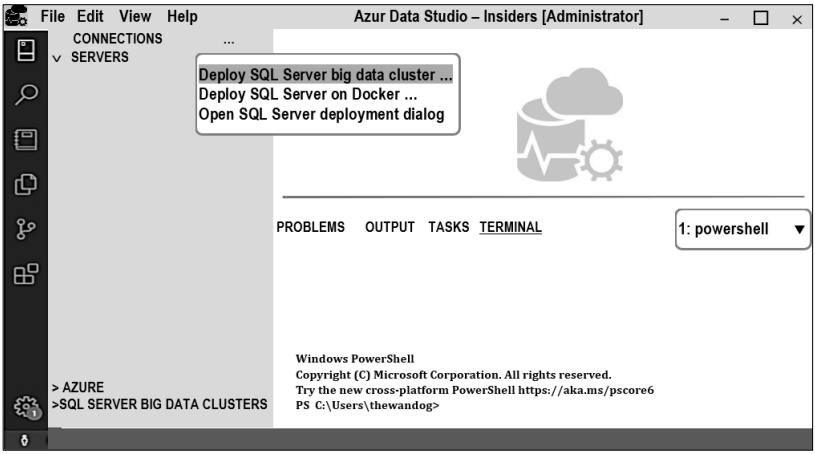

Рис. 5.2. Выбор варианта развертывания BDC в Azure Data Studio

На рис. 5.3 показан выбор метода развертывания.

Чтобы понять представление о развертывании BDC, развернем BDC на AKS и используем сценарий Python, предостав-

[1] *Развертывание* с помощью сценария bash в кластере kubeadm с одним узлом / Microsoft. URL: https://docs.microsoft.com/en-us/sql/big-data-cluster/deployment-script-single-node-kubeadm?view=sql-server-ver15 (дата обращения: 03.03.2023).

[2] *Как* развернуть кластеры больших данных SQL Server в Kubernetes / Microsoft. URL: https://docs.microsoft.com/en-us/sql/big-data-cluster/deployment-guidance (дата обращения: 03.03.2023).

ленный в одношаговом способе. Выберем настройки по умолчанию, за исключением того, что нужно было развернуть кластер AKS и BDC в области eastus2.

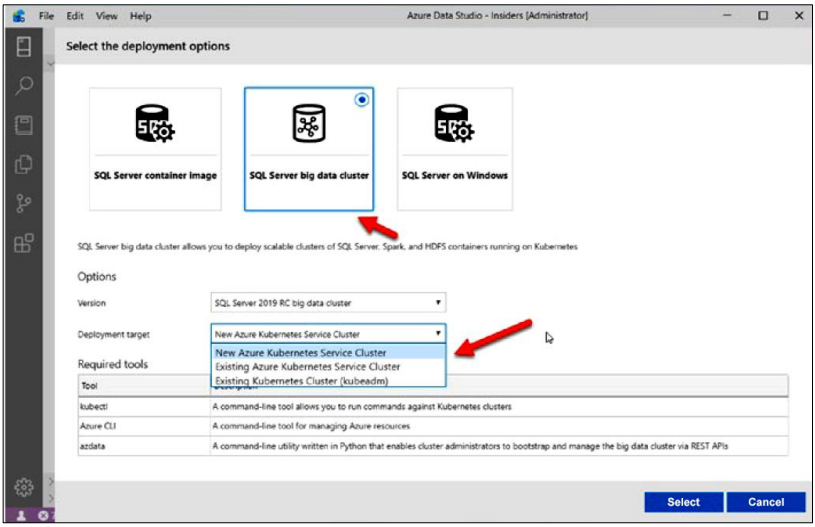

Рис. 5.3. Параметры развертывания для BDC в Azure Data Studio

Используемый сценарий Python по сути является «оберткой» для az и azdata. Он использует выбранные параметры (переменные или значения по умолчанию) для создания группы ресурсов Azure, кластера AKS и BDC. BDC создается с использованием конфигурации aks-dev-test. Это базовая конфигурация для BDC, которая хорошо подходит для сценария разработки или тестирования.

Для решения BDC нужно развернуть множество объектов pod и контейнеров, и этот процесс займет больше времени, если также будет развертываться кластер K8s.

При запуске сценария Python получим следующее:

```
Creating azure resource group: <rgname>
<json details for the resource group>
Creating AKS cluster: <aks cluster name>
<json for the AKS cluster>
Creating SQL Big Data cluster:mssql-cluster
```

```
custom\bdc.json created
custom\control.json created
The privacy statement can be viewed at:
https://go.microsoft.com/fwlink/?LinkId=853010

The license terms for SQL Server Big Data Cluster can be
viewed at:
https://go.microsoft.com/fwlink/?LinkId=2002534

Cluster deployment documentation can be viewed at:
https://aka.ms/bdc-deploy

NOTE: Cluster creation can take a significant amount of time
depending on configuration, network speed, and the number of
nodes in the cluster.

Starting cluster deployment.
Waiting for cluster controller to start.
```

Последнее сообщение «Waiting for cluster controller to start» (Ожидание запуска контроллера кластера) может повторяться несколько раз. Сначала в кластере K8s создается контроллер, а затем служба контроллера будет использоваться для развертывания оставшейся части BDC.

Далее получим следующее:

```
Cluster controller endpoint is available at <ip address>:<port>
Cluster control plane is ready.
```

А вскоре появятся следующие сообщения:

```
Data pool is ready.
Master pool is ready.
Compute pool is ready.
Storage pool is ready.
Cluster deployed successfully.
```

Рассмотрим следующие сценарии проверки работоспособности успешного развертывания AKS и BDC:

– выполним действия, описанные в документации, в которой сказано, как использовать kubectl для проверки кластера[1];

[1] *Используйте* скрипт Python для развертывания кластера больших данных SQL Server в службе Azure Kubernetes (AKS) / Microsoft. URL: https://docs.microsoft.com/en-us/sql/big-data-cluster/quickstart-big-data-cluster-deploy?view=sql-server-ver15 (дата обращения: 03.03.2023).

– войдем в кластер, используя azdata, найдем конечную точку контроллера, а затем попробуем подключиться к SQL Server, чтобы убедиться, что можно подключиться к нему[1].

Найдем конечную точку с названием **SQL Server Master Instance Front-End,** которой является IP-адрес и порт для подключения к SQL Server.

Следуя инструкциям на следующей странице документации, подключимся к SQL Server в BDC с помощью Azure Data Studio (ADS)[2].

Результат проверки возможности подключения к BDC в ADS показан на рис. 5.4. Необходимо проверить общее состояние BDC с помощью команды `azdata bdc status show`.

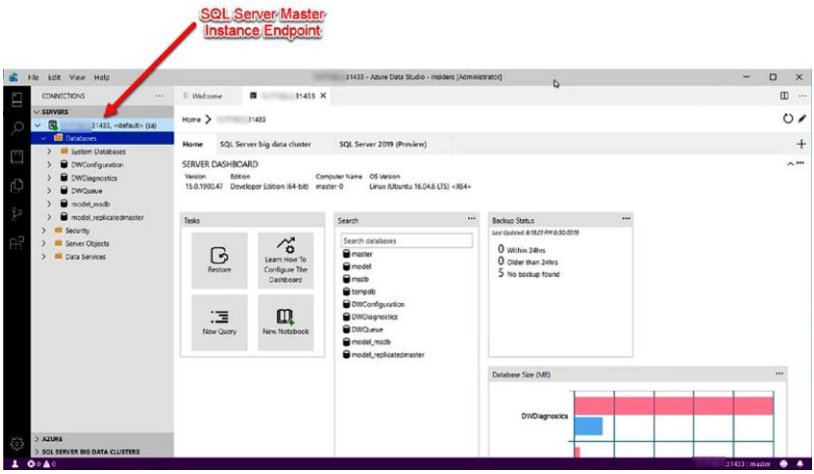

Рис. 5.4. Подключение к SQL Server в BDC после завершения развертывания

[1] *Кластеры* больших данных / Microsoft. URL: https://learn.microsoft.com/en-us/sql/big-data-cluster/?view=sql-server-ver15 (дата обращения: 03.03.2023).

[2] *Подключитесь* к кластеру больших данных SQL Server с помощью Azure Data Studio / Microsoft. URL: https://learn.microsoft.com/en-us/sql/big-data-cluster/connect-to-big-data-cluster?view=sql-server-ver15 (дата обращения: 03.03.2023).

Полученный для кластера BDC результат имеет следующую структуру:

```
Mssql-cluster: ready                                                    Health Status:
                                                                        healthy
================================================================================
Services: ready                                                         Health Status:
                                                                        healthy
--------------------------------------------------------------------------------
Servicename     State     Healthstatus     Details
sql             ready     healthy          -
hdfs            ready     healthy          -
spark           ready     healthy          -
control         ready     healthy          -
gateway         ready     healthy          -
app             ready     healthy          -
Sql Services: ready                                                     Health Status:
                                                                        healthy
--------------------------------------------------------------------------------
Resourcename    State     Healthstatus     Details
master          ready     healthy          StatefulSet master is healthy
compute-0       ready     healthy          StatefulSet compute-0 is healthy
data-0          ready     healthy          StatefulSet data-0 is healthy
storage-0       ready     healthy          StatefulSet storage-0 is healthy
Hdfs Services: ready                                                    Health Status:
                                                                        healthy
--------------------------------------------------------------------------------
Resourcename    State     Healthstatus     Details
nmnode-0        ready     healthy          StatefulSet nmnode-0 is healthy
storage-0       ready     healthy          StatefulSet storage-0 is healthy
sparkhead       ready     healthy          StatefulSet sparkhead is healthy
Spark Services: ready                                                   Health Status:
                                                                        healthy
--------------------------------------------------------------------------------
Resourcename    State     Healthstatus     Details
sparkhead       ready     healthy          StatefulSet sparkhead is healthy
storage-0       ready     healthy          StatefulSet storage-0 is healthy
Control Services: ready                                                 Health Status:
                                                                        healthy
--------------------------------------------------------------------------------
Resourcename    State     Healthstatus     Details
controldb       ready     healthy          -
control         ready     healthy          -
metricsdc       ready     healthy          DaemonSet metricsdc is healthy
metricsui       ready     healthy          ReplicaSet metricsui is healthy
```

```
metricsdb          ready     healthy      StatefulSet metricsdb is healthy
logsui             ready     healthy      ReplicaSet logsui is healthy
logsdb             ready     healthy      StatefulSet logsdb is healthy
mgmtproxy          ready     healthy      ReplicaSet mgmtproxy is healthy
Gateway Services: ready                                  Health Status:
                                                         healthy
-------------------------------------------------------------------------------------------------
Resourcename       State     Healthstatus Details
gateway            ready     healthy      StatefulSet gateway is healthy
App Services: ready                                      Health Status:
                                                         healthy
-------------------------------------------------------------------------------------------------
Resourcename       State     Healthstatus Details
appproxy           ready     healthy      ReplicaSet appproxy is healthy
```

Если возникли проблемы, следует обратиться к документации для устранения неполадок в кластере[1].

Выводы по главе 5

Рассматриваются большие данные как любые данные, которые невозможно обработать за приемлемое время с помощью имеющихся систем. Для SQL Server это означает, что в организации могут иметься данные, для хранения которых не подходит система управления реляционными базами данных, такая как SQL Server. Показано, что подобная ситуация может быть следствием многих разнообразных причин, в числе которых можно назвать объем, структуру, происхождение данных и сложность их преобразования в реляционные таблицы.

Рассмотрен кластер больших данных SQL Server как продукт, являющийся частью другого продукта, основные технологии SQL Server как центр доступа к данным в кластере. Определено, что большие данные — технологии больших данных, такие как HDFS и Spark. Большой эффект достигается при использовании кластера Kubernetes для развертывания и запуска различных контейнеров. Важными являются компоненты Polybase для SQL

[1] *Устранение* неполадок кластеров больших данных SQL Server Kubernetes / Microsoft. URL: https://learn.microsoft.com/en-us/sql/big-data-cluster/cluster-troubleshooting-commands?view=sql-server-ver15 (дата обращения 03.03.2023).

Server с BDC и распределенная файловая система Hadoop. Определена технология машинного обучения и развертывание кластеров больших данных с учетом выбора клиентского приложения и загрузка необходимого инструментария. Показаны возможности кроссплатформенного инструмента Azure Data Studio и выполнение проверки выполненного развертывания.

Вопросы для самоконтроля

1. Каковы основные технологии SQL Server как центра доступа к данным в кластере?
2. Назовите определения больших данных и технологии больших данных.
3. Что образует технологию развертывания и запуска различных контейнеров?
4. Каково назначение компонента Polybase для SQL Server с BDC?
5. Каково назначение файловой системы для Big Data?
6. Каковы цели машинного обучения?
7. Каково назначение кластеров больших данных и их развертывание для Big Data?
8. В чем заключаются критерии выбора клиентского приложения?
9. Какие задачи обеспечивает кроссплатформенный инструмент Azure Data Studio (ADS)?
10. Каковы этапы проверки выполненного развертывания Big Data Clusters?

Глава 6

Инновационная операционная система ArubaOS-CX

6.1. Особенности сетевой операционной системы ArubaOS-CX

Несмотря на быстрое развитие технологий и их влияние на бизнес, сетевая инфраструктура предприятий за последние два десятилетия не претерпела значительных инноваций в области дизайна и программного обеспечения.

Были предприняты несколько попыток, но с начала 2000-х гг. бизнес требовал от ИТ преимущественно решения задач по обеспечению взаимодействия между растущим числом цифровых подсистем бизнес-процесса и, как следствие, обеспечения передачи растущего объема сетевого трафика.

И действительно, на протяжении последних 20 лет скорости существенно выросли. Бизнес диктовал в том числе и функциональное развитие. Важность стабильной ИТ-инфраструктуры выражалась в более жестких требованиях к обеспечению отказоустойчивости на уровне коробки коммутатора или всей сети. Жадность и здравый смысл выражались в оптимизации использования ресурсов коммутаторов и каналов связи, включении базовых элементов информационной безопасности сети.

В то же время мало что поменялось с точки зрения эксплуатации сетевой инфраструктуры, для которой основным требованием было наличие интерфейсов взаимодействия с коммутаторами, механизмов оповещения и журналирования. Время как

будто замерло на 20 лет: традиционные и несменные ручные операции через CLI, сбор и офлайновый анализ syslog, оповещение только через SNMP и сложная интеграция со сторонними ИТ-системами из-за отсутствия принципа программируемости и программных интерфейсов.

Внешний мир ИТ, который основывается на сетевой инфраструктуре, сильно изменился: наступила эра мобильности, облачных технологий, Интернета вещей. Существенно развились web-технологии, подходы к программированию, архитектуры пользовательских приложений и взаимодействий между программными сервисами, подходы к хранению информации и т. д.

Изменения являются реакцией ИТ-отрасли на запросы бизнеса для обеспечения его гибкости и стабильного роста. Характерная для современной действительности цифровизация бизнес-процессов требует удобных инструментов для всех, кто так или иначе влияет на итоговые бизнес-показатели.

Инструментом офисного работника сейчас является специализированное программное обеспечение (приложение и платформа как технический комплекс) с применением современных технологий для реализации удобного пользовательского интерфейса, быстрого функционального развития и возможности масштабируемости.

Информационная сеть — это основной транспорт всех процессов, а администраторы — не последние люди в бизнесе, которые хотят работать с современными технологиями. Но необходимо, чтобы это не превратилось в самоцель, а дало как можно больше возможностей администраторам при эксплуатации или внедрении сетевой инфраструктуры, поэтому при выборе объекта последующего администрирования очень важно быть внимательным к следующим его характеристикам:

– наличию инструментов видимости происходящего и удобной диагностики для быстрого обнаружения неисправности и оперативного решения инцидентов и восстановления;

– наличию инструментов автоматизации для переключения ИТ-персонала от рутинных задач к специфическим. Например, поручить работу по анализу системных журналов инфраструктуры на предмет аномалий автоматизированному скриптовому агенту;

– наличию необходимой производительности, что является базовым и минимальным требованием, чтобы попасть практически в любой шорт-лист.

Особенность последнего пункта заключается в том, что быстрые порты с большими гигабитами, неблокируемые коммутационные фабрики на современных чипах и объемы памяти уже стали обыденностью. По первому пункту важно отметить, что во многом функциональность и инновационность практически любого ИТ-продукта составляет программное обеспечение.

Именно поэтому в продуктах фирмы **Aruba Networks** уделяют много внимания не только аппаратной платформе, но и сетевой операционной системе. С анонсом коммутаторов ядра и агрегации для сетей предприятий они представили новейшую ОС ArubaOS-CX, которая отвечает всем современным требованиям. ArubaOS-CX — это ОС для коммутаторов ядра и агрегации предприятий, которая автоматизирует и упрощает многие критические и сложные сетевые задачи, обеспечивает повышенную отказоустойчивость и позволяет проще справляться с непредсказуемыми событиями.

Рассмотрим ключевые особенности ArubaOS-CX.

Программируемость в своей основе — открытый API для автоматизации задач по работе с коммутатором и машинного сбора статистики. Использование Python для аналитики и мониторинга.

Функциональная расширяемость — модульная архитектура в стиле микро-сервисов позволяет быстро расширять функционал коммутатора, развивать платформу в зависимости от тенденций современного сетевого мира, добавлять уникальные инструменты.

Инновации — архитектура операционной системы, взаимодействие процессов посредством баз данных, наличие встроенной базы данных временных рядов, инструмент Network Analytics Engine, API — все это создает основу для инновационных подходов к управлению и работе с оборудованием сетевой инфраструктурой.

Безопасность — наличие модуля TPM для безопасного хранения цифрового сертификата, который используется в защите всех коммуникаций с коммутатором и при проверке цифровой подписи программных модулей ОС.

Отказоустойчивость — операционная система в стиле микросервисов обеспечивает большую степень отказоустойчивости и доступности, поскольку отдельные программные модули могут быть обновлены и перезапущены по отдельности.

Рассмотрим подробнее основные архитектурные особенности и инструменты операционной системы ArubaOS-CX.

Использование модульной архитектуры в стиле микросервисов и взаимодействие процессов посредством базы данных — одно из ключевых преимуществ ArubaOS-CX.

База данных — центральный компонент. Она обеспечивает взаимодействие между процессами, позволяет снизить зависимость процессов друг от друга и других компонентов.

Такой подход отличается от традиционной архитектуры сетевой ОС, при которой процессы непосредственно обмениваются сообщениями между собой, формируя сложную матрицу межпроцессового взаимодействия и зависимостей. В ArubaOS-CX база данных берет на себя исключительную роль с точки зрения общения между процессами. Все состояния и события, генерируемые процессами, записываются в базу данных, откуда они доступны для чтения другим процессам в ОС.

Более того, базой поддерживается модель взаимодействия, при которой процессы могут «подписаться» на мгновенное получение информации о событиях, произошедших в других конкретных процессах. При такой архитектуре работают следующие сценарии. Например, процессы ARP и маршрутизации во время своей работы обновляют только таблицы базы данных, а уже специализированный процесс получает агрегированную информацию из базы и программирует ASIC линейных плат.

Другой пример — процесс завис и перезапустился. Сохраненная в базе информация о его состоянии может быть использована при восстановлении процесса и (или) как источник последних актуальных данных для другого процесса, пока первый не восстановится.

Подобный подход создает хорошие условия для высокой надежности как отдельного элемента внутри ОС, так и всего коммутатора и, как следствие, сети.

Сценарий сегодняшнего дня: ИТ-департамент выявляет неисправности и устраняет неполадки постфактум, используя такие

инструменты, как CLI и SNMP, или привлекая сторонние системы мониторинга, аналитики и диагностирования.

При грамотном подходе процесс устранения неисправности подразделяется на этапы. С учетом характера или симптома проблемы делается предположение о том, что может выйти из строя и для подтверждения или опровержения предположения выполняется сбор информации. Как правило, это — выполняемая вручную диагностика через консоль или терминал SSH. На основе собранной информации вырабатывается и реализуется первичный план действий для устранения проблемы. Затем проверяется, решилась проблема или нет. Если проблема не решена, выдвигается следующее предположение.

Network Analytics Engine (NAE) — встроенный в ArubaOS-CX инструмент, который автоматизирует процесс сбора и последующего анализа информации, обеспечивает интеллектуальный мониторинг и ускоряет процесс поиска причины неисправности и решения инцидентов.

NAE выполняет непрерывный мониторинг определяемых администратором параметров, например: объем трафика, показатели «здоровья» компонентов коммутатора. NAE способен замерять показатели, хранить статистику в базе данных временных рядов и визуализировать их в виде графиков, коррелировать графики с событиями и действиями администратора на оборудовании, например, с изменениями в конфигурации, отслеживать исторические тенденции.

С помощью NAE администратор сможет отслеживать происходящие в сети события, выявлять слабые места в безопасности, узкие места производительности и превентивно их устранять. При обнаружении аномалии в серии собранных данных NAE может сгенерировать оповещение через традиционный Syslog или завести кейс с использованием API, выполнить действие локально на коммутаторе или опросить через API соседние коммутаторы.

Для подобной умной аналитики NAE использует агенты. Для всех, кто знаком с языком программирования Python и умеет его использовать, написание скрипта-агента для NAE будет простой задачей. Для некоторых это будет прекрасным драйвером начать изучать и применять программирование в эксплуатации

сетевой инфраструктуры. Остальные же могут использовать GUI для загрузки готовых скриптов из репозитория. Процесс загрузки скрипта напоминает процесс установки приложения из Google Play.

Virtual Switching Extension (VSX) — технология виртуализации для коммутаторов уровней распределения (ядра), работающих под управлением операционной системы ArubaOS-CX. В сетевой индустрии можно выделить два основных подхода в реализации технологий виртуализации для обеспечения отказоустойчивости на уровне ядра и агрегации.

Первый — создание логически единого со всех точек зрения виртуального коммутатора, второй — с применением технологий, которые виртуализируют Data Plane, не затрагивая Control Plane и Management Plane.

Независимо от подхода, целью является возможность создания распределенного логического канала через агрегацию физических портов, взятых с разных коммутаторов-участников виртуализации (стека). При реализации VSX используются преимущества и устраняются недостатки обоих подходов.

В отличие от стандартных механизмов виртуализации (VSF, VSS, IRF) или, как принято говорить, стекирования, VSX позволяет двум коммутаторам работать как единое устройство на втором уровне OSI-модели, при этом сохраняя независимость на L3. Сохраняется также и независимость управления, но не требуется настраивать каждый коммутатор отдельно: возможна синхронизация конфигурации. С помощью Virtual Switching Extension администратор может отказаться от устаревших протоколов резервирования с медленной сходимостью и сложными настройками (STP, VRRP), при этом сохраняя отказоустойчивость и получая возможность балансировать нагрузку и использовать оптимальные маршруты для трафика.

6.2. Коммутаторы на базе ArubaOS CX

ArubaOS-CX является основой для коммутаторов уровня ядра и агрегации Aruba 8400, Aruba 8325, Aruba 8320, которые характеризуются высокой производительностью, отказоустойчивостью, широким набором поддерживаемых сетевых протоколов.

Для всех коммутаторов ядра, агрегации и центров обработки данных на базе ОС ArubaOS-CX выделяются следующие ключевые особенности:

– обеспечивает автоматизацию и удобство использования с помощью REST API и скриптов Python;

– интеллектуальный мониторинг, видимость происходящего и быстрая диагностика с инструментом Aruba Network Analytics Engine;

– расширенный набор функций уровня 2/3 включает в себя BGP, OSPF, VRF и IPv6 и т. д.;

– технологии виртуализации коммутаторов (VSX) для повышенной отказоустойчивости и организации MLAG.

Коммутатор Aruba 8400 (рис. 6.1) представляет собой компактное 8U шасси с возможностью установки восьми интерфейсных модулей с портами 10GbE (SFP/SFP+), 40GbE (QSFP+) и 40GbE/100GbE (QSFP+/QSFP28). Высокопроизводительная фабрика обеспечивает производительность на слот до 1.2 Tbps с возможностью роста. Aruba 8400 с точки зрения архитектуры аппаратной платформы отвечает не только требованиям сетей предприятий, но и требованиям высокой доступности сетей операторского класса с резервированием (указывается количество подсистем питания и количество подсистем резерва).

Рис. 6.1. Коммутатор серии Aruba CX 8400

Коммутаторы серии Aruba 8325 (рис. 6.2) обеспечивают скорость передачи данных на скорости 1/10/25GbE (SFP/SFP+/SFP28) и 40/100 GbE (QSFP+/QSFP28) в компактном форм-факторе 1U. Коммутаторы поддерживают стек VXLAN/EVPN. Таким образом, они решают задачи не только на уровне корпоративного ядра и агрегации, но и на уровнях spine и leaf (ToR, EoR) в центре обработки данных. Традиционно для таких моделей обеспечивается резервирование по блокам питания и охлаждения.

Рис. 6.2. Коммутатор серии Aruba CX 8325

Коммутаторы серии Aruba 8320 (рис. 6.3) обеспечивают скорость передачи данных 1/10GbE (SFP/SFP + и 10GBASE-T) и 40GbE в компактном форм-факторе 1U. Вместе с модульным шасси Aruba 8400 и серией коммутаторов Aruba 8325 серия коммутаторов Aruba 8320 дополняет портфель коммутаторов Aruba для корпоративного ядра и агрегации и помогает обеспечить высокую производительность и большую продолжительность безотказной работы.

Рис. 6.3. Коммутатор серии Aruba CX 8320

Недавно созданы коммутаторы серии Aruba CX 6300 с фиксированным набором портов и серии модульных коммутаторов Aruba CX 6400. Они предназначены для уровней доступа, дистрибуции и ядра.

Выводы по главе 6

Определено, что инновационная операционная система ArubaOS-CX — это ОС для коммутаторов ядра и агрегации предприятий, которая автоматизирует и упрощает многие критические и сложные сетевые задачи, обеспечивает повышенную отказоустойчивость и позволяет проще справляться с непредсказуемыми событиями.

Рассмотрены различные коммутаторы на базе ArubaOS-CX.

Вопросы для самоконтроля

1. Каковы возможности операционной системы ArubaOS-CX?
2. В чем заключаются изменения эксплуатации сетевой инфраструктуры?
3. Каковы свойства и назначение информационной сети?
4. В чем заключается администрирование информационной сети?
5. Каковы ключевые особенности ArubaOS-CX?
6. Каково назначение коммутатора Aruba 8400?
7. Каково назначение коммутатора Aruba 8325?
8. Каково назначение коммутатора Aruba 8320?

Глава 7

Администрирование серверных систем

7.1. Управление программным обеспечением

Системы управления программным обеспечением служат для администрирования операционной системы Linux и других UNIX-подобных систем. Это набор программного обеспечения, позволяющего устанавливать, настраивать, обновлять и удалять программы и их компоненты.

Задачи управления программным обеспечением: реализация удобного механизма взаимодействия программиста и пользователя; управление внутренними ресурсами системы; разработка и введение стандартов, т. е. администрирование.

Установка программного обеспечения возможна следующими способами.

Первый способ. Новое программное обеспечение можно установить с помощью пакетов, которые хранятся в репозиториях (рис. 7.1), местах хранения файлов, свободно распространяющихся в сети. В репозиториях хранятся уже скомпилированные программы с метафайлами, в которых описаны основные параметры и свойства программы, а также текущая версия установочного пакета. Такой способ установки имеет ряд минусов: не все разработчики программного обеспечения могут себе позволить содержать репозиторий, для установки требуется Интернет, пакеты скомпилированы под конкретную архитектуру. Однако безусловным плюсом является удобство работы с пакетами программного обеспечения и их установкой.

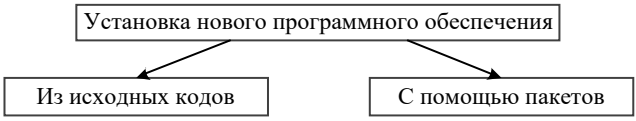

Рис. 7.1. Схема вариантов установки программного обеспечения

Второй способ. Довольно трудоемкий для начинающего пользователя, заключающийся в установке программ из исходных кодов, которые свободно распространяются в Интернете.

Рассмотрим **менеджеры пакетов.** Утилита под названием rpm (RPM Package Manager) разрабатывалась для Red Hat Linux, но позже была перенесена и в другие дистрибутивы Linux (рис. 7.2).

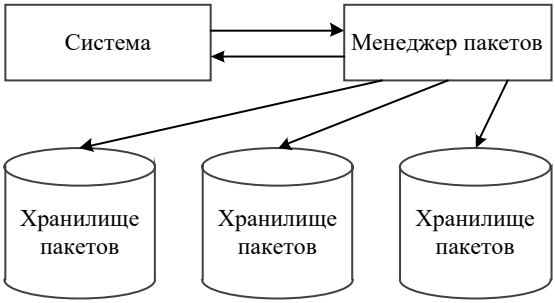

Рис. 7.2. Схема работы менеджера пакетов

Синтаксис утилиты: `rpm [режим опции] [пакет].rpm`
Режимы работы и основные опции:
`-q` — получение информации (запрос);
`-i` — установка;
`-U` — обновление;
`-V` — проверка пакетов;
`-e` — удаление;
`-v` — показать подробную информацию;
`-i` — получить информацию о пакете (`-qi`);
`-I` — список файлов пакета (`-qI`);
`-c` — список конфигурационных файлов пакета (`-qc`);
`-a` — список пакетов, установленных в системе;

`--force` — выполнять действие принудительно;

`--nodeps` — не проверять зависимости;

`--test` — проверить последствия удаления или установки пакета.

Yellowdog Ubdater, Modified (YUM) — консольный менеджер RPM-пакетов, представляющий собой оболочку для RPM, распространяется под лицензией GNU.

Основные команды `yum`:

`$ yum help` — список команд и опций;

`$ yum list` — список названий пакетов из репозиториев;

`$ yum list installed` — список установленных пакетов;

`$ yum repolist` — список подключенных репозиториев;

`$ yum install [пакет]` — установить пакет;

`$ yum remove [пакет]` — удалить пакет;

`$ yum update [пакет]` — обновить пакет;

`$ yum info [пакет]` — вывести информацию о пакете;

`$ yum search [строка]` — найти пакет по строке в описании.

В современных дистрибутивах, основанных на RedHat, применяется обновленный, более быстрый по сравнению с YUM менеджер пакетов DNF. Они имеют сходный синтаксис.

Advanced Packaging Tool (APT) — утилита для управления ПО в операционных системах Debian, а также системах, основанных на Debian, таких как Ubuntu и Mint. Утилита имеет широкое распространение, есть как консольные версии, так и большое число графических оболочек, таких как Synaptic и Adept.

Синтаксис утилиты: `apt [опции] [команда] [пакет]`.

Полезные опции для APT:

`-t` — версия релиза, для которой устанавливается пакет;

`-f` — выполнить операцию принудительно;

`-o` — строка конфигурации.

Основные команды APT:

`download` — скачать пакет без установки;

`update` — обновление информации о пакетах в репозиториях;

`upgrade` — обновление системы без удаления пакетов;

`search` — поиск пакетов в локальной базе;

`show` — просмотр информации о пакете;

`install` — установка пакета;

`remove` — удаление пакета;

`purge` — полное удаление пакета (вместе с конфигурационными файлами).

Рассмотрим и некоторые другие утилиты:

Aptitude — оболочка с графическим интерфейсом для **Advanced Packaging Tool**.

dselect — программа управления программными пакетами в операционной системе Debian. Является старейшим фронтендом к **dpkg**. На сегодняшний момент практически полностью вытеснена **Advanced Packaging Tool**.

dpkg — основное программное обеспечение системы управления пакетами Debian, а также систем со схожей архитектурой.

`dpkg -i [пакет].deb` — установка пакета .deb.

Во время установки программного обеспечения система управления распределяет файлы программы по следующим системным папкам:

/usr/bin — исполняемые файлы программ;

/usr/sbin — исполняемые файлы, запускаемые с правами администратора;

/usr/lib — библиотека программы;

/usr/share — остальные файлы программы;

/usr/local — ПО, собранное самостоятельно из исходных кодов.

С помощью команды `dpkg -s [название пакета]` можно получить список файлов и их местонахождение в системных папках.

Разберем, как устанавливать программы с открытым исходным кодом. Этот способ установки программ подходит для компьютеров, которые не имеют доступа в Интернет. Обычно программы с исходным кодом распространяются в виде упакованных архивов, которые перед установкой необходимо распаковать, используя, например, следующие команды:

```
tar -Jxvf [пакет].tar.xz
tar -zxvf [пакет].tar.gz
```

Дальше запускается файл конфигурации:

```
./configure
```

Скрипт командной оболочки **configure** будет пытаться определить правильные значения переменных, использующихся в процессе компиляции. Эти переменные необходимы для создания файлов **Makefile**, содержащих правила компиляции, сборки и установки, в каждом каталоге устанавливаемой программы. Также **configure** выполняет определение необходимых заголовочных файлов, содержащих системные зависимости.

Переход к установке:

```
make install
```

Удаление:

`make uninstall` — удаление установленных файлов (обратно make install);

`make clean` — удаление скомпилированных файлов в месте сборки (обычно — очистка каталога build).

Пример Makefile с пояснениями в виде комментариев (#):

 # объявление переменной

`-c` — собрать объектный файл.

`-Wall` — включить все предупреждения

```
CFLAGS=-c -Wall
```

 # цель по умолчанию (выполняется, если команда make вызвана без указания цели)

```
all: project
```

это — цель сборки. После двоеточия указывают, от чего она зависит.

Отсутствие зависимости позволяет не перестраивать весь проект при изменении одного файла с кодом.

```
project: 1.o 2.o 3.o
```

 # в следующей строке обязателен отступ (TAB). Это команды (компиляции)

```
	$(CC) 1.o 2.o 3.o -o project
1.o: 1.cpp
	$(CC) $(CFLAGS) 1.cpp
2.o: 2.cpp
	$(CC) $(CFLAGS) 2.cpp
3.o: 3.cpp
	$(CC) $(CFLAGS) 3.cpp
```

 # цель для очистки проекта (удаление скомпилированных файлов) clean:

```
rm -rf *.o project
```

Часто при компиляции из исходных кодов конфигуратор (**configure**) не может найти ту или иную библиотеку. Иногда название библиотеки может не совпадать с названием пакета в репозитории. В таком случае можно:

1) найти недостающую библиотеку

```
apt-cache search ключевое_слово
```

2) воспользоваться системами управления с графическим интерфейсом, например, **synaptic**, который выполнит поиск нужной библиотеки внутри пакетов.

Ошибки могут возникнуть при конфликте пакетов друг с другом, тогда для решения проблемы придется вручную удалять наименее важные пакеты и проверять целостность оставшихся.

Исполняемые программы в Linux делятся на два типа.

1. **Статически скомпонованные**. Они уже содержат в себе все необходимые библиотечные функции, так что не требуется предварительная установка компонентов, однако занимают больше места по сравнению с динамически скомпонованными.

2. **Динамически скомпонованные**. Занимают меньше места по сравнению со статически скомпонованными, но требуют установку дополнительных компонентов (библиотек).

Расположение библиотек в Linux. Системное программное обеспечение стандартно устанавливается в следующих расположениях:

– исполняемые файлы: **/bin**, **/sbin**, **/usr/bin**, **/usr/sbin**;

– библиотеки: **/lib**, **/usr/lib**, **/lib64**, **/usr/lib64**.

Как правило, в 64-разрядных ОС Linux поддерживаются и 32-, и 64-разрядные программы. Путь к библиотекам для 64-разрядных библиотек: **/lib64**, **/usr/lib64**, а к 32-разрядным **/lib**, **/usr/lib**.

Пользовательское программное обеспечение может быть установлено в системные расположения, если оно предоставляется в системе как пакет (возможна его установка через менеджер приложений) или каталог /opt (традиционный вариант, поскольку в прошлые десятилетия пользовательское ПО часто устанавливали с оптических дисков).

Библиотеки, в свою очередь, делятся на статические (static library) и динамические (shared library). Статические библиотеки используются на этапе сборки программы, а динамические во время ее выполнения.

Каждая разделяемая библиотека Linux имеет расширение so.X.Y.Z.

X — версия публичного интерфейса (API). Считается, что каждый раз, когда он меняется, программа теряет обратную совместимость (нельзя работать с новой версией, если программа использует старый интерфейс).

Y — ревизия. Обычно добавляет новые функциональные возможности к программе. При смене ревизии публичный интерфейс не меняется, т. е. программы, работающие со старой версией, смогут работать и с новой. Некоторые части интерфейса могут быть признаны устаревшими, но не могут быть удалены, так как это нарушит API.

Z — патч. Обновленная версия программы, исправляющая ошибки, улучшающая реализацию, работоспособность. Патч не меняет публичный интерфейс и не добавляет новые функции, поэтому программа с патчем или без него совместима не только обратно, но и прямо: старая версия может быть гарантированно использована вместо новой с точки зрения совместимости (как правило, старую версию использовать нежелательно, с точки зрений наличия ошибок, пониженной безопасности, меньшей производительности).

Рассмотрим **ключевые системные библиотеки:**

linux-vdso.so.1 — виртуальный динамический разделяемый объект — библиотека для быстрого использования системных вызовов;

/lib/ld-linux.so.2 и **/lib64/ld-linux-x86-64.so.2** — исполняемый файл, отвечающий за определение необходимых динамических библиотек для запущенного приложения и их связывание с приложением.

Рассмотрим **общие библиотеки**:

/etc/ld.so.cache — файл, указывающий, где искать динамические библиотеки по умолчанию (общие библиотеки);

/etc/ld.so.conf — конфигурационный файл для генерации /etc/ld.so.cache;

Idconfig — утилита для генерации /etc/ld.so.cache.

LD_LIBRARY_PATH — переменная для добавления путей к библиотекам (в случае совпадения имен приоритетом обладают библиотеки, подключенные через LD_LIBRARY_PATH).

Иногда ошибки могут возникать при запуске программ (отсутствие библиотек).

Не найдена разделяемая библиотека:

```
Error loading shared library libopenblas.so.3:
No such file or directory
```

Решение:

```
$ ldd «исполняемый файл, использующий разделяемые
библиотеки»
```

В списке библиотек ненайденные будут помечены как «Not Found».

```
$ sudo apt install apt-file
$ apt-file update
$ apt-file find [название ненайденной библиотеки]
$ sudo apt install [название пакета]
```

Централизованное управление программным обеспечением осуществляется через кроссплатформенную клиент-серверную программу Puppet. Она используется для администрирования в крупных компаниях, позволяет легко настроить сеть и управлять ей на основе различных операционных систем.

Системные журналы

В системах Linux существует пять основных стадий загрузки ОС.

1. **BIOS** — отвечает за базовый ввод и вывод данных с устройства на устройство. Загружает главную загрузочную запись (**Master Boot Record, MBR**). В BIOS всегда задан пользователем или по умолчанию порядок опроса дисков. Загрузка происходит с первого диска, на котором найдена действительная запись MBR.

2. **MBR** — основная загрузочная запись на диске (**/dev/hdX** или **/dev/sdX**). Вызывает загрузчик ОС (в Ubuntu Linux — Grub), имеет размер 512 байт.

Загрузочная запись состоит из кода загрузчика (446 байт), таблицы разделов и сигнатуры, указывающей на корректность загрузочной записи (всегда 55h AAh, «h» означает шестнадцате-

ричный код). Из-за ограниченного размера загрузочной записи таблица разделов может содержать не более четырех разделов по 16 байт. Каждый раздел имеет, помимо других атрибутов, свой код: 82h — Linux swap, 83h — Linux, 85h — Linux extended (расширенный). Вместо одного из разделов может присутствовать расширенный раздел, который указывает на расширенную загрузочную запись. Каждая расширенная загрузочная запись (EBR) содержит указатель на следующую EBR, таким образом, число разделов не ограничивается.

Если вместо BIOS используется более новый EFI, то вместо MBR используется таблица разделов GPT. Она состоит из 34 блоков по 512 байт, записанных симметрично с двух концов диска для дублирования и облегчения восстановления повреждений. Блоки имеют номер от 0 до 33, блок 0 соответствует MBR для совместимости.

3. **GRUB** (**Grand Unified Bootloader**) — загрузчик операционной системы под лицензией **GNU**. Позволяет выбирать загружаемую ОС (Windows, Linux и др.) и используемое ядро Linux среди установленных.

Возможности загрузчика Grub:

– поддерживает работу с файлами;

– поддерживает файловые системы: ext2, ext3, ReiserFS, XFS, JFS, FAT32, FAT16, NTFS, ISO и др.;

– загружает ядра GNU/Linux, GNU/HURD, FreeBSD, SUN Solaris;

– осуществляет загрузку Windows (передает управление другим загрузчикам);

– обладает встроенной командной оболочкой;

– поддерживает загрузчик EFI (начиная с версии 1.98);

– защита пунктов меню паролем;

– поддерживает загрузку через сеть по протоколу TFTP и BOOTP.

Администрирование Grub:

`/boot/grub/menu.lst` или `/boot/grub.conf` — последовательность команд, выполняемых Grub. Эти файлы генерируются на основе пользовательского файла **/etc/default/grub**;

`sudo update-grub` — обновить **/boot/grub/menu.lst** и **/boot/grub.conf** на основе **/etc/default/grub**;

<c> — перейти в оболочку Grub (во время загрузки);

<b> — продолжение загрузки.

Рассмотрим **команды Grub**:

`help` — помощь;

`geometry (hd0)` — определить структуру HDD;

`root (hd0, 1)` — раздел, с которого читается загрузочная запись (диск 0, раздел 1);

`setup (hd0)` — установить загрузчик в главную загрузочную запись;

`quit` — выйти из командной оболочки Grub;

`default` — образ загрузки по умолчанию;

`timeout` — задержка в секундах перед загрузкой образа по умолчанию;

`splashimage=(hd0,4)/boot/grub/splash.xpm.gz` — фоновое изображение при загрузке;

`title` — описание образа для загрузки;

`initrd` — имя образа начального электронного диска;

`chainloader (file)` — передать управление другому загрузчику (указать сектор).

Установка:

`/sbin/grub-install` — установить загрузчик Grub из командной оболочки bash.

4. **Kernel** — ядро операционной системы. Запускает программу `/sbin/init`.

5. **Init** — система инициализации в **System V** (ОС Linux, выпущенная в 1980-х гг. компанией AT&T. Инициализация служб по образцу System V на долгие годы становилась стандартом инициализации ОС Linux), которая просматривает файл **/etc/inittab** для определения уровней выполнения и запускает остальные процессы. Команда `init` позволяет переключить уровень выполнения `runlevel`:

`init 5`

`Runlevel` — уровень запуска и выполнения служб:

`0` — выключение системы;

`1` — однопользовательский режим (только root);

`2` — обычно эквивалентен уровню 3. На некоторых системах (RedHat) не поддерживает NFS;

`3` — многопользовательский режим;

`4` — уровень графического сеанса в некоторых дистрибутивах;

`5` — в большинстве дистрибутивов — уровень графического сеанса (в том числе, для Ubuntu);

`6` — перезагрузка системы.

`/sbin/runlevel` — узнать текущий уровень выполнения.

На конечном этапе загрузки происходит изменение runlevel от однопользовательской ОС (1) до многопользовательской ОС (3) с графическим интерфейсом (5). Серверные ОС остаются в режиме выполнения 3.

Уровни выполнения, журналирование с помощью syslog — это элементы концепции **System V** (**«System Five»**). Сейчас в большинстве дистрибутивов операционной системы Linux они частично или полностью заменены службой systemd, но обычно могут быть использованы и установлены параллельно.

На каждом уровне выполнения при переходе на него, а также при завершении работы на нем запускаются скрипты в каталогах **/etc/rc0.d** — **/etc/rc6.d** (это настроено в файле **/etc/inittab**), цифра соответствует уровню выполнения.

Для добавления в ОС собственной автозапускаемой программы **command1** используется команда

```
sudo update-rc.d command1 start 70 2 3 4 5 . stop 30 0 1 6
```

`70` и `30` — приоритет запуска (меньшее число означает более раннюю загрузку и/или завершение);

`2, 3, 4, 5` — уровни работы программы;

`0, 1, 6` — уровни, где программа не будет запущена.

Скрипт должен быть создан на основе файла **/etc/init.d/skeleton**.

Для всех уровней загрузки автозапуск можно настроить путем внесения строки в файл **/etc/rc.local**.

Служба syslog

Syslog — стандарт отправки и регистрации сообщений о происходящих в системе событиях, использующихся в компьютерных сетях, работающих по протоколу IP.

syslogd — демон службы syslog.

/etc/syslog.conf — файл конфигурации службы syslog.

Все службы и системные приложения Linux раньше относились к четко определенной группе. Принадлежность к группе определялась в соответствии с задачами, решаемыми приложением. Если приложение обрабатывало почту, оно относилось к группе mail, если оно отвечало за вопросы безопасности системы — к группе security и т. д. В файле конфигурации **/etc/syslog.conf** описываются действия (action) при получении сообщений с определенным приоритетом (priority), принимаемых от определенных групп (источников сообщений facility) (рис. 7.3).

Рис. 7.3. Схема конфигурации syslog

Можно использовать шаблон глобальной подстановки «*» (glob, wildcard), вместо конкретного названия указывая, что подходит любой приоритет сообщения, например, строка в файле конфигурации:

```
mail.err   /var/log/mail.err
```

Это означает прием сообщений группы «mail» с приоритетом «err» и запись их в файл **/var/log/mail.err**.

Отправить сообщение процесс может, например, использовав утилиту logger:

```
logger -p mail.err -t "MAIL DELIVERY FAILURE" 'Mail delivery
failed to the following recepients: user@gmail.com'
```

`-p` — селектор (источник и приоритет);

`-t` — тема (заголовок) сообщения.

Последним параметром утилите logger передается сам текст сообщения.

Источники сообщений:

/var/log — директория, основное место хранения лог-файлов (системных журналов);

/var/log/syslog или **/var/messages** — глобальный системный журнал, который содержит записи с момента запуска системы (запуск ядра, различных служб, обнаруженные устройства, интерфейсы и многое другое);

/var/log/auth.log или **/var/log/secure** — записи об авторизациях пользователей, включая неудачные попытки;

/var/log/dmesg — драйвера устройств;

/var/log/cron — сообщения служб cron и at;

/var/log/daemon — сообщения от демонов (служб);

/var/log/kern — сообщения ядра;

/var/log/lpr — сообщения службы печати;

/var/log/user — сообщения пользовательских программ;

/var/log/syslog — собственные сообщения службы syslog.

Приоритеты:

debug — отладочная информация;

info — информационное сообщение;

notice — основные события;

warning (warn) — предупреждение;

err (error) — ошибка;

crit — критическое событие;

alert — требуется немедленное вмешательство;

emerg (panic) — система неработоспособна.

Основные способы фильтрации сообщений:

`[источник].*` — все записи;

`[источник].=[уровень]` — только записи указанного уровня;

`[источник].![уровень]` — все записи не меньше указанного уровня;

`[источник].!=[уровень]` — все записи, кроме указанного уровня.

Ротация системных журналов

В каталоге **/var/log** находятся архивы системных логов, так как эта информация может быть довольно громоздкой. Имеется команда для ротации (**logrotate**) этих файлов, благодаря чему нет перемешивания новых и устаревших записей. Команда **logrotate** запускается автоматически, но есть возможность запустить ее вручную. Команда нумерует файлы, добавляя **".[число]"** в конец файла, а также может удалять, сжимать, создавать и посылать по почте файлы журналов, что определяется следующими идентификаторами:

`dmesg` — беглый обзор информации о последней загрузке;

`tail` — последние записи в журнал;

`more` — постраничный просмотр;

`less` — просмотр и поиск информации;

`logger` — помещение в журнал своих сообщений.

Ротация журналов настраивается в общем файле **/etc/logrotate** и специфичных файлах для отдельных служб внутри директории **/etc/logrotate.d**. Доступны следующие опции:

`weekly, dayly, monthly` — периодичность ротации;

`rotate <n>` — количество хранимых старых журналов;

`create 0666 root` — с какими правами создавать новый журнал;

`compress` — сжимать файлы журналов;

`notifempty` — запрет ротации пустых журналов;

`include <file>` — включить файл или множество файлов;

`mail` — посылать журналы по e-mail;

`size` — максимальный размер журнала.

Пример ежедневной ротации лог-файла **/var/log/filefire/main.log** для некоторой псевдослужбы «filefire»:

```
/var/log/filefire/main.log {
  daily
  rotate 3
  postrotate
    filefire reload > /dev/null
  endscript
  compress
}
```

Запуск служб вручную (System V):

`/sbin/service network` или `/etc/inid.d/network start` — запуск службы network вручную;

`/sbin/chkconfig -list` — получить информацию о процессе начальной загрузки;

`/sbin/chkconfig --level 2345 network on` — включить исполнение network на уровнях 2, 3, 4, 5;

`/sbin/chkconfig --add <служба>` — добавить службу;

`/sbin/chkconfig --del <служба>` — удалить службу;

`/sbin/chkconfig [level levels] on | off | reset` — включить, выключить, перезапустить;

`/sbin/runlevel или who -r` — определить текущий уровень выполнения;

`/sbin/init 1` — перейти на уровень выполнения 1.

Завершение работы и перезагрузка ОС:

`/sbin/init 0` или `halt` — немедленное завершение работы ОС;

`/sbin/init 6` или `reboot` — немедленная перезагрузка ОС;

`halt -p` или `poweroff` — завершение работы с отключением питания;

`/sbin/shutdown [опции] [время] [сообщение]` — команда выключения, перезагрузки, остановки, завершения сеанса локальных или удаленных компьютеров.

Опции `shutdown`**:**

`-c` — отменить начавшуюся остановку системы;

`-a` — создать файл **/fastboot** и не проверять файловую систему при загрузке;

`-F` — создать файл **/forcefsk** — проверить файловую систему при загрузке;

`-h` — остановка системы;

`-r` — перезагрузка системы;

`-R` — послать пользователям сообщение, но не перезагружать систему.

Примеры:

`/sbin/shutdown -r 16:00 'Reboot at 16:00!'` — перезагрузка в 16:00;

`/sbin/shutdown -h now` — остановка прямо сейчас;
`/sbin/shutdown -h +10` — остановка через 10 минут.

Работа с systemd

Подсистема systemd имеет два основных преимущества по сравнению с System-V: параллельный запуск служб при инициализации ОС и запуск служб по требованию.

В systemd единицей взаимодействия и настройки является юнит. Базовые виды юнитов:

service — управляет демонами;

socket — конфигурационный файл сокета;

device — конфигурационный файл с правилом udev для дерева устройств;

mount — отвечает за монтирование файловой системы;

automount — отвечает за автоматическое монтирование файловой системы.

Запуск служб вручную (systemd):

`systemctl start <unit>` — запустить юнит;
`systemctl stop <unit>` — остановить юнит;
`systemctl restart <unit>` — перезагрузить юнит;
`systemctl reload <unit>` — перезагрузить конфигурацию юнита;
`systemctl status <unit>` — проверка статуса юнита;
`systemctl enable <unit>` — сделать юнит активным (можно запускать);
`systemctl disable <unit>` — сделать юнит неактивным (нельзя запускать).

Для отладки работы юнитов можно просмотреть только информацию об юнитах с проблемами:

```
systemctl --failed
```

Общий журнал расположен в **/var/log/journal**. Существуют следующие команды:

`journalctl -b` — вывести журналы, записанные с начала работы системы;
`journalctl /usr/lib/systemd/system` — все сообщения утилиты system;
`journalctl _PID=ЧИСЛО` — вывести журнал по PID процесса (используется при отладке работы юнитов).

Создание юнитов

Системные юниты находятся в директории **/usr/lib/systemd/system**, а пользовательские юниты — в директории **/etc/systemd/system**.

Для юнитов создаются файлы конфигурации **.service**. Юнит имеет обязательные секции **[Unit]** и **[Install]**, а также **[Сервис]**. При отсутствии файла юнита, но наличии файла System V с тем же именем без суффикса **.service**, юнит будет быстро создан. Также для сервиса **Test** файлы с суффиксом **.conf** из каталога **Test.service.d** будут прочтены после разбора основного файла конфигурации юнита.

Пример файла юнита **test1**:

```
[Unit]
Description=Test
[Service]
ExecStart=/usr/sbin/test1
[Install]
WantedBy=multi-user.target
```

Важные опции раздела **[Unit]**:

`Required=` — настраивает зависимости в виде жестких требований. Если какой-либо юнит, на который опирается текущий, не был запущен, то текущий юнит не будет запускаться;

`Wants=` — слабое требование. Юнит попробует стартовать, если какая-либо зависимость не была запущена;

`Before =, After =` — настройка зависимостей между юнитами.

Важные опции раздела **[Install]**:

`RequiredBy` — обратная (какие юниты зависят от текущего) жесткая зависимость;

`WantedBy` — обратная мягкая зависимость;

`sudo apt install syslog-ng` — обеспечивает включение syslog в systemd наряду с journalctl.

В **systemd** для использования **/etc/rc.local** должен быть включен сервис **rc-local.service**:

```
systemctl status rc-local.service
```

7.2. Сетевые службы Linux

Сетевая служба (сервис, демон) — специальный процесс (или взаимосвязанный ряд процессов) операционной системы, отвечающий за работу с локальными или удаленными соединениями заданного типа.

Сетевая служба создает **сокет** (пара IP-адреса и номера порта) и привязывает его к порту в операционной системе. Говорят, что в таком случае служба выполняет прослушивание порта, т. е. ожидает запросов на установку соединения через этот порт.

Порт — целое число, имеющее номер от 0 до 65535. Номера портов от 1 до 1023 (общеизвестные порты) зарезервированы организацией IANA (Internet Assigned Numbers Authority) под стандартные службы. Порты с номерами от 1024 до 49151 также должны проходить регистрацию в организации. Это обеспечивает то, что никакие две службы не будут пытаться прослушать один порт. Порты с большими номерами являются динамическими.

Рассмотрим некоторые общеизвестные порты и протоколы, по которым осуществляется работа через эти порты:

– 20 — данные, передаваемые через протокол FTP;

– 21 — управляющая информация, передаваемая через протокол FTP;

– 22 — протокол удаленного шифрованного соединения SSH;

– 23 — удаленное соединение Telnet;

– 25 — почтовый протокол SMTP (отправка писем);

– 53 — протокол DNS;

– 80 — протокол HTTP;

– 110 — почтовый протокол POP3 (прием писем с сервера);

– 143 — почтовый протокол IMAP (прием и отправка заголовков писем);

– 443 — протокол HTTPS;

– 465 — шифрованный протокол SMTP;

– 993 — шифрованный протокол IMAP;

– 995 — шифрованный протокол POP3.

В файле **/etc/services** можно найти полный список сетевых служб: имя службы, номер порта, используемый протокол, псевдонимы службы.

`netstat -an` — получить список активных соединений.

Опции netstat:

`-a` — выводить список портов;

`-n` — выводить информацию в виде чисел;

`-t` — только TCP-порты;

`-u` — только UDP-порты.

В выводе команды можно видеть следующие состояния соединения:

`LISTEN (LISTENING)` — идет прослушивания порта, создан сокет;

`ESTABLISHED` — установленные соединения.

Создание удаленного подключения telnet:

```
telnet имя_или_адрес_компьютера
```

В UNIX было популярно использовать утилиты **rsh/rlogin** для входа на удаленный компьютер. В файле **/etc/hosts.equiv** — доверенные компьютеры, вход с которых осуществляется для заданных пользователей без пароля при входе через **rsh/rlogin**. Но эти утилиты не используют шифрование (как и telnet) и в настоящий момент их категорически не следует применять.

Связь с удаленным компьютером через зашифрованное соединение. Пакет OpenSSH:

– sshd — сервер службы SSH, прослушивает порт 22 (TCP);

– ssh — клиент службы SSH, позволяет начать сеанс связи с удаленным компьютером;

– scp — программа для удаленного копирования;

/etc/ssh/sshd_config — конфигурационный файл настройки сервера;

/etc/ssh/ssh_config — настройка клиентов ssh и scp.

ssh имя_пользователя@имя_или_адрес_компьютера — начать работу с удаленным компьютером. Разрешение имен в IP-адреса производится с использованием файла **/etc/hosts**.

Копирование файлов через ssh:

```
scp пользователь@компьютер:<откуда-копировать>
<куда-копировать>
```

Пример:

```
scp 1.txt user@10.0.0.100:/home/user/shared_docs/
```

Традиционно аутентифицируются либо по паролю (менее безопасно), либо через связку публичного и приватного ключей.

Для аутентификации используются протоколы шифрования RSA (в примере) или DSA:

`ssh-keygen -t rsa` — создание пары ключей (открытый и закрытый) асимметричного шифрования. Публичный (открытый) ключ **~/.ssh/id_rsa.pub** передается на все компьютеры, с которыми должна осуществляться связь, его содержимое копируется как строка в файл авторизованных (которые разрешены для авторизации) ключей:

```
cat id_rsa.pub >> ~/.ssh/authorized_keys
```

Этот файл находится в директории **.ssh** в домашней папке пользователя, под которым осуществляется вход на удаленном компьютере. На файл **authorized_keys**, должны быть выставлены права 600:

```
chmod 600 ~/.ssh/authorized_keys
```

На саму директорию **.ssh** — права 700:

```
chmod 700 ~/.ssh
```

Владельцем директории **.ssh** и файлов в ней должен быть тот пользователь, под которым осуществляется вход.

Приватная часть ключа (закрытый ключ) **~/.ssh/id_rsa** строго никому никогда не передается. Это конфиденциальная информация, которую вместо пароля используют для входа на удаленные машины.

При подозрении на то, что приватный ключ скомпрометирован, следует немедленно создать новую пару ключей и удалить все старые публичные ключи на удаленных машинах, чтобы третьи лица, завладевшие утерянным приватным ключом, не смогли получит к ним доступ. Аналогично следует поступать и при компрометации пароля пользователя (установить новый пароль).

~/.ssh/authorized_keys — публичные ключи для беспарольного доступа.

~/.ssh/known_hosts — известные хосты, с которых подключение осуществляется без дополнительного подтверждения.

Протокол передачи файлов FTP:

`ftp <адрес_сервера>` — синтаксис команды;

`sftp <адрес_сервера>` — ftp через зашифрованное соединение (через ssh). Следует всегда использовать эту команду вместо незашифрованоного ftp.

Команды интерактивного режима `ftp/sftp`:

`ftp> help` — вызов справки;

`ftp>?` — аналог help;

`ftp> ls` — вывод списка файлов и директорий; `ls` выполняет команды для локального, а не удаленного компьютера;

`ftp> cd 123` — перейти в каталог 123;

`ftp> cdup` — аналог cd .. в Linux;

`ftp> ascii` — передача файлов в текстовом режиме ASCII;

`ftp> binary` — передача файлов в двоичном режиме (нельзя передавать двоичные данные в текстовом режиме, так как это приводит к потере данных);

`ftp> get 1.jpg` — загрузить файл **1.jpg** с удаленного компьютера (без подтверждения перезаписи);

`ftp> put 1.jpg` — загрузить файл **1.jpg** на удаленный компьютер (без подтверждения перезаписи);

`ftp> mget *.txt` — загрузить все текстовые файлы с удаленного компьютера;

`ftp> mput *.txt` — загрузить все текстовые файлы на удаленный компьютер;

`ftp> delete 1.txt` — удалить файл 1.jpg с удаленного компьютера;

`ftp> mdelete *.txt` — удалить все текстовые файлы с удаленного компьютера.

Копирование файла в неинтерактивном режиме (аналог scp):

```
sftp user@10.0.0.100:docs/1.txt ~/
```

Утилита wget:

`wget <адрес_или_шаблон>` — загрузка файлов с удаленного адреса.

Опции:

`-c` — возобновление загрузки в случае разрыва соединения;

`-i <file>` — взять адреса загружаемых объектов из файла;

`-r` — рекурсивная загрузка каталогов;

`-l <num>` — ограничить рекурсию до уровня num;

`-x` — создавать необходимую структуру каталогов;

`--passiveftp` — использовать пассивный режим ftp.

Пример использования:

`wget -c http://releases.ubuntu.com/16.04.4/ubuntu-16.04.4-desktop-amd64.iso` — загрузить образ диска Ubuntu 16.04 (система с пятилетней поддержкой до апреля 2021 г.).

NFS (Network File System) — универсальный, не зависящий от ОС протокол сетевого доступа к файловым системам. Для обмена данными между клиентом и сервером используется протокол RPC. Ресурсы NFS можно монтировать при помощи команды mount.

/etc/exports — файл настройки NFS.

Возможное содержимое файла:

`/home/user/dir 10.1.12.0/24 (ro,async,all_squash,insecure)` — предоставить каталог в режиме «только для чтения» для узлов сети 10.1.12.0/24;

`sync/async` — синхронный/асинхронный режим доступа — определяет, возможен ли ответ сервера до окончания записи на диск изменений, async может привести к потере данных при обрыве соединения;

`all_squash` — опция означает, что подключения будут выполняться от имени анонимного пользователя;

`sudo apt install nfs-kernel-server nfs-common` — установка поддержки NFS в Ubuntu;

`showmount -e` — вывести список ресурсов, предоставленных системе;

`/etc/init.d/nfs-kernel-server restart` — перезапуск службы NFS (Ubuntu);

`exportfs -a` — считать заново настройки /etc/exports;

`mount -t nfs 10.5.122.221:/home/user/mnt/uhome` — монтирование ресурса NFS.

Возможно записать в /etc/fstab информацию для удобства монтирования NFS-ресурсов через нажатие кнопки на панели дисковых устройств в Nautilus:

`10.5.122.221:/home/user /mnt/uhome nfs user,rw,noauto 0 0`

Служба печати CUPS:

`sudo apt install cups` — установка CUPS через менеджер пакетов apt.

Служба обслуживания очереди заданий на печать имеет название **cupsd**. Она должна быть запущена для работы CUPS. Конфигурационные файлы CUPS находятся в директории **/etc/cups**:

– **cupsd.conf** — основной файл настроек сервера;

– **printers.conf** — описания и настройка принтеров;

– **classes.conf** — описания классов (групп принтеров);

– **client.conf** — индивидуальные настройки для клиентов (формат файлов схож с форматом сервера Apache). В этом файле должна быть строка с указанием адреса сервера печати:

`ServerName server.example.com:port`

Важно перед изменением создать резервные копии конфигурационных файлов.

`sudo /etc/init.d/cups restart` — перезапуск службы cupsd после изменений конфигурации.

В основном файле конфигурации должны присутствовать следующие строки, которыми устанавливается прослушивание порта по умолчанию на сервере 10.0.10.129, разрешается удаленный доступ к средствам печати (Allow @LOCAL):

```
Listen 127.0.0.1:631
Listen 10.0.10.129:631
<Location />
Order allow,deny
Allow @LOCAL
</Location>
```

Веб-интерфейс для конфигурирования доступен по адресу **http://localhost:631**. Также для конфигурирования существует утилита командной строки `lpadmin`, например:

`Ipadmin -p laser -o job-page-limit=10` — установить ограничение в 10 страниц;

`lpadmin -p laser -u allow:stud1` — разрешить печать пользователю stud1;

`lpadmin -p laser -o Resolution=600dpi` — изменить разрешение печати;

`lpoptions -l` — получить текущие настройки.

Печать с помощью CUPS документа **test.txt** осуществляется командой

`lp test.txt`

Опции `lp`:

`-d` — имя принтера;

`-h` — имя узла сети;

`-i` — идентификатор задания;

`-n` — количество копий;

`-q` — приоритет задания;

`-u` — имя пользователя;

`-H` — дополнительные опции задания;

`-P` — список страниц для печати.

`lp -d HP1 -n 3 1.txt` — печать трех копий файла **1.txt**.

`lpstat -a` — состояние очереди печати;

`lp -i HP-10 -H immediate` — перемещение задания в очереди на первую позицию (-H immediate);

`cancel` — снять задание с печати.

Пакет SAMBA позволяет Linux-компьютеру работать в сети Windows, он предназначен для общего использования ресурсов: файлов и каталогов, а также устройств.

`sudo apt install samba` — установка SAMBA в Ubuntu.

`/etc/samba/smb.conf` — основной файл настройки.

Для печати важны следующие настройки **smb.conf**:

```
workgroup = COMP
...
security = user
...
[printers]
browsable = yes
guest ok = yes
```

Для Windows-компьютеров следует добавить опцию:

`use client driver = yes`

Применение изменений выполняется командами:

```
sudo restart smbd
sudo restart nmbd
```

Пример части основного файла конфигурации SAMBA:

```
[global]
netbios name = IKT-201
```

```
workgroup = COMP
security = user
guest account = nobody
[homes]
comment = Home directories
read only = no
[PUB]
comment = public share
path = /home/samba/pub
read only = no
guest ok = yes
[STUD]
path = /home/samba/student
browseable = no
valid users = student
```

Важные настройки в файле конфигурации:

– `global` — раздел общих настроек;

– `printers` — описывает методы подключения к принтерам;

– `homes` — настраивает доступ к домашним каталогам пользователей;

– `PUB` — настройки для публичного доступа;

– `STUD` — настройки для пользователя student;

– `netbios name` — имя компьютера в сети NetBIOS;

– `workgroup` — название рабочей группы NetBIOS;

– `security` — идентификация пользователей через использование домена;

– `share` — права доступа проверяются при подключении к ресурсу, но после подключения к компьютеру, предоставляющему ресурс;

– `user` — права доступа при подключении к компьютеру, предоставляющему ресурсы;

– `domain` — компьютер является членом домена NT, аутентификация производится на контроллере домена (PDC);

– `server` — задает имя иного сервера аутентификации;

– `guest account` — имя анонимного пользователя;

– `comment` — необязательный комментарий;

– `path` — путь к разделяемому ресурсу, или к очереди печати принтера;

– `read only` — доступ только для чтения;

– `guest ok` — разрешение гостевого доступа;

– `browseable` — разрешение отображать ресурс в сетевом окружении.

Настройка SAMBA может быть осуществлена как редактированием файла **smb.conf**, так и с помощью утилит, например, **SWAT**.

Подключение через SAMBA:

`testparm -s | less` — проверка правильности конфигурации smb.conf;

`nmblookup <имя компьютера>` — проверить разрешение NetBIOS-имен службой nmbd;

`nmblookup -M ++-` — определить, какой компьютер сети является мастер-браузером (компьютером, обслуживающим базу данных NetBIOS-имен компьютеров с именем по умолчанию __MSBROWSE__);

`smbclient -L <имя-компьютера>` — получить список smb-ресурсов на сервере;

`smbclient //<имя-компьютера>/<имя-ресурса>` — подключение к SAMBA-ресурсу (из ОС Linux);

`smbclient //IKT-201/pub` — аналог предыдущей команды;

`net view \\IKT-201` — подключение к SAMBA-ресурсу (из ОС Windows)

`/sbin/useradd -M -d /home/samba/stud1 stud1` — регистрация SAMBA-пользователя;

`smbpasswd -a stud1` — установка пароля;

`smbclient //IKT-201/STUD -U stud1` — указание пользователя при доступе;

`/etc/samba/smbpasswd` — база данных учетных записей SAMBA.

Монтирование файловых ресурсов SAMBA. Команда `mount.cifs (smbmount)` — монтирование файловых ресурсов протокола SMB/CIFS (реализована как демон, контролирующий события, происходящие во время монтирования и использования файловых ресурсов).

`smbmount //IKT-201/pub/mnt/winshare -o username=nobody, password=' '`

Разберем синтаксис команды:

`//IKT-201` — имя компьютера;

`pub` — имя ресурса;

`/mnt/winshare` — точка монтирования.

Команда `smbstatus` выводит список текущих подключений SAMBA.

7.3. Сетевые средства Linux

Адресация IPv4 (стандарт RFC 791, 1981 г.). IP-адрес — адрес компьютера в сети, состоящий из 32 бит (версия 4 стандарта). Их записывают в виде четырех десятичных чисел от 0 до 255 (по 8 бит), разделенных точкой, A.B.C.D, где A,B,C,D ∈ [0,255]. Пример: 192.168.10.10.

В IP-адресе можно выделить две части, разделенные точкой: **[адрес сети].[адрес узла]**. Длина адреса сети разнится для сетей разных классов. Сети класса A имеют первые 8 бит, начинающиеся с 0. Это соответствует A ∈ [0,127], всего 128 сетей (рис. 7.4). При этом 24 бита произвольны и соответствуют адресу узла (хоста, компьютера). Каждая сеть, соответственно, включает (2^{24}–2) узлов в сети (16 777 214). Число «2» отнимается, поскольку каждая сеть всегда включает широковещательный адрес (последний в сети, 127.255.255.255) и нулевой адрес, который означает саму сеть и используется для маршрутизации.

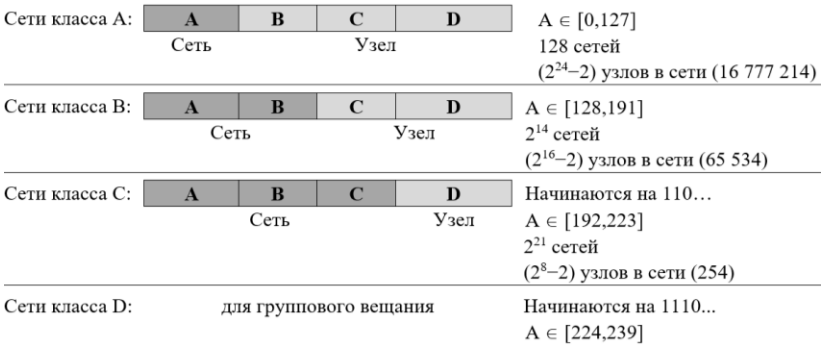

Рис. 7.4. Схема адресации IPv4

Маска подсети определяет то, какая часть адреса является адресом сети. Там, где биты относятся к адресу сети (а не узла), в маске содержатся единицы (табл. 2).

Таблица 2

Маска подсети

	В десятичном виде	В двоичном виде
IP-адрес	192.168.111.253	1100 0000 1010 1000 0110 1111 1111 1101
Маска	255.255.255.0	1111 1111 1111 1111 1111 1111 0000 0000
Адрес сети	192.168.111.0	1100 0000 1010 1000 0110 1111 0000 0000

Маски для сетей различных классов:

– класс A: маска 255.0.0.0;

– класс B: маска 255.255.0.0;

– класс C: маска 255.255.255.0;

– класс D: маска 255.255.255.255.

Зарезервированные адреса. Адрес 127.0.0.1 сети класса A 127.0.0.0 — интерфейс «петля». Блоки адресов для локальных сетей:

– класс A: 10.0.0.0−10.255.255.255, одна сеть;

– класс B: 172.16.0.0−172.31.255.255, 16 сетей;

– класс C: 192.168.0.0−192.168.255.255, 256 сетей.

Адресация IPv6 (RFC 2373). Возможности IPv6:

– расширенное адресное пространство (128 бит);

– нет понятия класса сети;

– упрощенный формат заголовка IP-пакета;

– встроенная поддержка IPSec — создание виртуальных частных сетей с шифрованными каналами;

– поддержка IPMobile — для мобильных пользователей.

Настройка сетевого интерфейса **Ethernet**:

`/sbin/ifconfig` — посмотреть текущие настройки сети;

`lsmod` — убедиться, что загружен модуль ядра, **/etc/ modprobe.conf**, связанный с сетью;

`eth0, eth1, eth2...` — имена сетевых интерфейсов;

`ifconfig eth0 [ip-адрес]` — просмотр базовой конфигурации устройства (интерфейса);

`ifconfig -a` — информация обо всех интерфейсах;

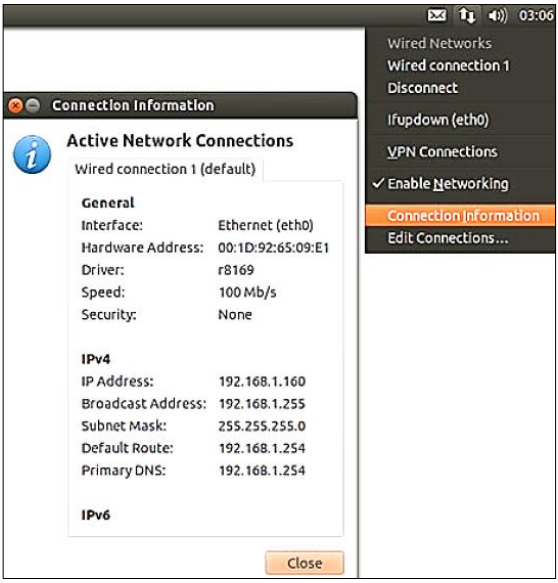

Рис. 7.5. Информация о сетевых соединениях в Ubuntu

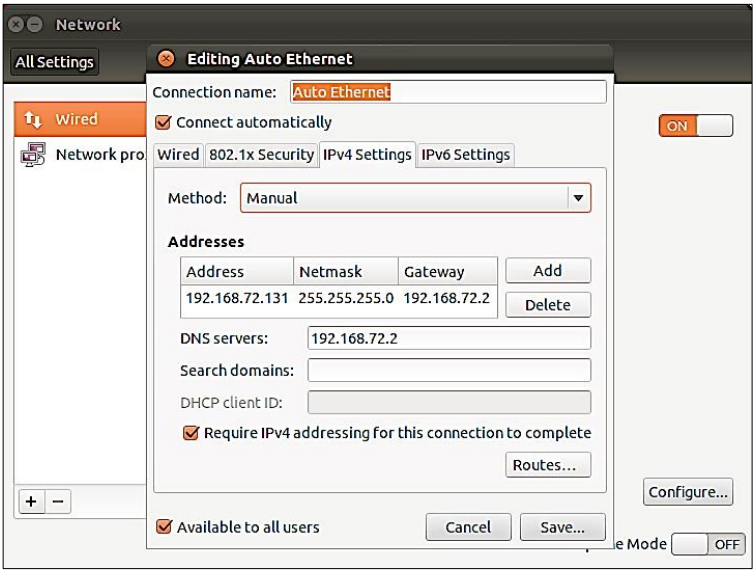

Рис 7.6. Управление сетями Network в Ubuntu

`netstat -i` — информация о трафике (пакетах) через все интерфейсы;

`ping [ip-адрес]` — проверка работоспособности сетевого интерфейса.

Настройка сетевого интерфейса в Ubuntu осуществляется через специальные программы, вызываемые либо нажатием на значок в панели значков в верхнем правом углу экрана (рис. 7.5), либо в верхнем меню **Параметры системы** — **Сеть** (рис. 7.6).

Рассмотрим **настройку маршрутизатора по умолчанию.**

Маршрутизатор (роутер) — устройство, передающее IP-пакеты между различными сетями. Обычно при настройке подключения указывают маршрутизатор по умолчанию (default gateway).

В случае локальной сети или соединения точка-точка маршрутизацию настраивать не требуется. Для функционирования маршрутизации ядро должно поддерживать обмен пакетами между интерфейсами (forwarding).

Рассмотрим команды:

– `netstat rn` — получить текущую таблицу маршрутизации (содержит маршруты направления IP-пакетов в различные сети);

– `arp an` — посмотреть кеш протокола ARP (преобразование IP-адресов в MAC-адреса). Для использования arp без указания пути директория **/sbin** должна быть в переменной **$PATH**;

– `route add default gw [ip-адрес]` — назначить маршрутизатор по умолчанию;

– `traceroute [ip-адрес]` или `[имя_домена]` — утилита для построения маршрута между текущим компьютером и целевым (покажет все узлы, через которые будет проходить пакет). Позволяет отследить процесс прохождения пакетов через маршрутизаторы. Существует аналог — `tracepath`; `tracert` — аналог команды traceroute для Windows.

Просмотр и смена сетевого имени:

– `hostname` — получить сетевое имя компьютера;

– `/etc/sysconfig/network` — настройка имени компьютера в Red-Hat-подобных системах.

Настройка имени компьютера в Ubuntu (13.04 и выше):

1) изменить имя хоста в файлах **/etc/hosts** и **/etc/hostname**;

2) `systemctl restart systemd-logind.service`

Другой способ:

```
hostnamectl set-hostname new_host_name
```

Настройка разрешения имен. Разрешение имен — установление соответствия между парой «хост, домен» и IP-адресом хоста.

Способы разрешения имен:

– файл **/etc/hosts** — для небольших сетей;

– использование службы DNS (Domain Name System);

– использование служб NIS, NIS+, LDAP — для корпоративных сетей.

Важные файлы для конфигурации разрешения имен:

– **/etc/resolv.conf** — настройка системы разрешения имен. В файле встречаются термины `nameserver` — сервер DNS, `domain` — имя домена, `search` — список доменов, которые нужно подставлять при поиске;

– **/etc/host.conf** — тонкая настройка системы разрешения имен;

– **/etc/nsswitch.conf** — информация о доменах и связанных с ними базах данных.

Поиск и устранение проблем с сетью. Основные причины проблем в работе сети:

– неисправности в сетевых кабелях и сетевом оборудовании;

– отсутствует драйвер для сетевой карты или установлен не тот драйвер;

– неверно настроены IP-адреса узлов сети;

– засорен кеш ARP;

– неверно указан маршрутизатор по умолчанию;

– неверно настроена система разрешения имен;

– избыточная блокировка фильтром IP-пакетов (сетевым экраном).

Устранение проблем:

– `/sbin/lspci` — проверить работоспособность сетевой карты;

– `/sbin/modinfo` — получение информации о модуле ядра;

– `/sbin/lsmod` — список загруженных модулей;

– `/sbin/ifconfig` и `netstat rn` — проверить, правильно ли настроен IP-адрес и маршрутизатор по умолчанию.

Алгоритм проверки работоспособности сети при помощи команды ping.

1. Проверить работоспособность сетевой подсистемы в ядре:
```
ping 127.0.0.1
```
2. Проверить работоспособность сетевого адаптера:
```
ping [IP-адрес, присвоенный адаптеру]
```
3. Проверить работоспособность локальной сети:
```
ping [IP-адрес локального компьютера]
```
4. Проверить прохождение пакетов к внешнему сетевому адаптеру маршрутизатора по умолчанию:
```
ping [IP-адрес маршрутизатора]
```
5. Проверить прохождение пакетов по любому внешнему IP-адресу:
```
ping [IP-адрес удаленного компьютера]
```
6. Проверить работу подсистемы разрешения имен:
```
ping [имя домена]
```

Утилита netstat — информация о соединениях, и процессах, их открывших.

`netstat -tulpn | grep :80` — вывести информацию об открытых портах и отфильтровать только информацию о порте 80.

Nmap — утилита для исследования состояния компьютеров в сети (проверка открытых портов и т. п.). Применяется для проверки безопасности и корректности настройки сети.

`nmap -sV -p 22,80,110,143,8080 192.168.0-10.1-255` — исследование состояния портов 22,80,110,143,8080 с попыткой определения открывшего порт приложения для ряда хостов от 192.168.0.1 до 192.168.10.255.

`nmap -sP 192.168.0.0/16 -oG file.txt` — пинг-сканирование всей локальной сети класса B для ряда хостов от 192.168.0.1 до 192.168.255.255 с выводом результатов в файл **file.txt** в удобном для программы grep формате.

Проверка кеша ARP. Ethernet работает с MAC-адресами, а не IP-адресами, поэтому для установки соединения компьютер выполняет широковещательный запрос (broadcast) по протоколу ARP. Целевой компьютер получает запрос и посылает ответный, содержащий MAC-адрес. Кеш ARP позволяет не перезапрашивать IP-адрес заново в течение некоторого времени (не превышающего, обычно, несколько часов).

`/sbin/arp a` — вывести содержимое кеша ARP (должно быть правильное сопоставление MAC-адресов и IP-адресов);

`/sbin/arp d` — удаление неправильных записей в кеше;

`/sbin/arp s` — установить запись вручную.

DNS (Domain Name System) — компьютерная система для получения информации о доменах. Основное употребление — перевод доменного имени в IP-адрес.

Для сайта **disk.yandex.ru** уровни доменов: III — disk, II — yandex, I — ru (доменная зона).

host — утилита для тестирования клиента;

dig — утилита для тестирования DNS-сервера;

nslookup — позволяет тестировать и сервер DNS, и клиента — подсистему разрешения имен.

Утилита IPTraf — мониторинг сетевого трафика.

/var/log/iptraf — директория для хранения лог-файлов по умолчанию.

Сетевой экран (брандмауэр, firewall) — программа, осуществляющая фильтрацию сетевого трафика на основе заданного набора правил. Популярный межсетевой экран, встроенный в ядро Linux с версии 2.4, — netfilter. Утилита конфигурирования правил в netfilter называется iptables.

Основы конфигурирования с помощью iptables. Обмен трафиком между компьютерами происходит пакетами (информация разбивается на части опредленного размера, упаковывается и пересылается). Пакеты пропускаются через цепочки, т. е., списки правил.

Каждый пакет проходит цепочку, начиная от первого правила. Как только найдено соответствие правилу, осуществляется переход (-j = --jump) к указанному действию (обычно — REJECT, ACCEPT, DROP).

Существует пять основных типов цепочек:

1) PREROUTING — предварительная обработка входящих пакетов;

2) INPUT — цепочка входящих пакетов, предназначенных для самой истемы;

3) FORWARD — для перенаправляемых пакетов ();

4) OUTPUT — цепочка исходящих пакетов от локальных процессов;

5) POSTROUTING — пост-обработка исходящих пакетов.

Переадресуемые пакеты проходят три цепочки: PREROUTING — FORWARD — POSTROUTING, а не одну.

Стандартные действия, доступные во всех цепочках: ACCEPT (разрешить прохождение пакета), DROP (отклонить), QUEUE (передать на анализ внешней программе) и RETURN (вернуть в предыдущую цепочку).

Цепочки организованы в таблицы.

filter — основная таблица, используемая по умолчанию, если название таблицы не указано. Содержит цепочки INPUT, FORWARD и OUTPUT.

`iptables-save > /etc/network/iptables.rules` — сохранить изменения iptables в файл.

Следует в файле **/etc/network/interfaces** для интерфейса устройства, к которому применяется политика фильтрации пакетов, указать строку

```
pre-up iptables-restore < /etc/network/iptables.rules
```

В версии Ubuntu 16.04 появился специальный пакет, упрощающий сохранение конфигурации iptables:

```
sudo apt install iptables-persistent
```

Сохранение и перезагрузка (применение) конфигурации iptables могут быть произведены командами:

```
sudo /etc/init.d/iptables-persistent save
sudo /etc/init.d/iptables-persistent reload
```

Примеры использования iptables:

`iptables -L -n -v --line-numbers` — просмотреть таблицы;

`iptables -I INPUT 3 -s 205.205.205.205 -j DROP` — вставить в цепочку INPUT правило под номером 3, блокирую-

щее (-j DROP) прием пакетов от источника (-s = --source) 205.205.205.205;

`iptables -A INPUT --source 127.0.0.1 --jump ACCEPT` — пропустить все пакеты от 127.0.0.1;

`iptables -A INPUT --jump FURTHER_ANALYSIS` — отправить оставшиеся пакеты на анализ в цепочку FURTHER_ANALYSIS;

`iptables -A OUTPUT -o lo -j ACCEPT` — разрешить трафик для интерфейса lo (внутренний трафик интерфейса локальная петля, lo — стандартное название интерфейса);

`iptables -A INPUT --i eth0 -p tcp --dport 9000 -j LOG --log-prefix "Port 9000 incoming blocked!"` — позволяет отправить информацию о пакетах, пришедших на порт 9000 через устройство eth0 в /var/log/messages;

`iptables -A INPUT -i eth0 -p tcp --dport 9000 -j DROP` — блокировка приема пакетов, пришедших на порт 9000 через устройство eth0;

`host -t a имя_домена` — найти IP-адрес для домена;

`whois IP-адрес | grep CIDR` — найти диапазон IP-адресов, используемых доменом и поддоменами.

На основе информации о диапазоне адресов, можно заблокировать выход на определенный домен и его поддомены.

`iptables -A OUTPUT -o eth0 -d 205.205.205.0/24 -j DROP` — запретить исходящие соединения в заданную подсеть.

`iptables -A INPUT -p icmp --icmp-type echo-request -j DROP` — замаскировать компьютер, скрыв его от ping-запросов.

`iptables -A INPUT -m conntrack --ctstate ESTABLISHED, RELATED -j ACCEPT` — разрешить входящий трафик, если соединение с удаленным копьютером было инициализировано первично локальным компьютером.

`iptables -A INPUT -p tcp -s 205.205.205.0/24 --dport 22 -m conntrack --ctstate NEW,ESTABLISHED -j ACCEPT` — разрешить соединения по порту 22 (стандартно используется службой ssh) только с компьютеров подсети 205.205.205.0/24 (если политика по умолчанию для входящих пакетов — DROP).

Механизм NAT (Network Address Translation) — IP-адрес локального компьютера подменяется сетевым экраном или маршрутизатором подменяется на имеющийся внешний IP-адрес (например, маршрутизатора) для обмена пакетами с удаленным компьютером, для каждого соединения используется отдельный порт. Пакеты, поступившие от удаленного пакета по этому порту, переадресуются обратно инициировавшему сеанс связи локальному компьютеру.

Существуют антивирусные программы для Linux. Популярный антивирус — ClamAV. Управляется из командной строки.

Установка:

```
sudo apt install clamav clamav-daemon или sudo yum install clamav
```

Обновление базы данных вирусов:

```
sudo freshclam
```

Сканирование без очистки с оповещением при нахождении вирусов:

```
sudo clamscan -r /home/user --bell
```

Сканирование с автоматическим удалением инфицированных файлов:

```
sudo clamscan --infected --remove -r /home/user
```

Выводы по главе 7

В настоящее время с открытым кодом используют преимущественно семейство операционных систем Linux, работающих на основе одноименного ядра. Нет одной операционной системы Linux, как, например, Windows или MacOS. Рассмотрены множества дистрибутивов (набор файлов, необходимых для установки ПО), выполняющих конкретные задачи. Основное внимание уделено свободно распространяемым операционным системам семейства Linux, которые популярны в научной среде и в среде информационных технологий, бесплатны и имеют открытый исходный код.

Вопросы для самоконтроля

1. Что понимается под управлением программным обеспечением?
2. Где располагаются установленные программы в файловой системе?
3. Где расположены библиотеки в Linux?
4. Каково назначение ключевых системных библиотек?
5. Каково назначение системных журналов? Опишите процесс загрузки и уровни выполнения.
6. Каковы задачи администрирования Grub?
7. Каково назначение файлов конфигурации syslog?
8. Каково назначение сетевых служб Linux?
9. Каково назначение сетевых средств Linux?

Заключение

Было рассмотрено современное определение операционной системы как комплекса управляющих и обрабатывающих программ, которые, с одной стороны, выступают интерфейсом между устройствами вычислительной системы и прикладными программами, а с другой стороны, предназначены для управления устройствами, вычислительными процессами, эффективного распределения вычислительных ресурсов между вычислительными процессами и организации надежных вычислений. Это определение применимо к большинству современных операционных систем общего назначения.

Определено, что администрирование операционной системы — управление ее компонентами — важное и ответственное дело. Интерес процесса администрирования в том, что один человек, он же администратор, не покидая своего рабочего места, может влиять на работу целой сети, в которую может входить от нескольких до многих тысяч компьютеров.

Показано, что в настоящее время операционная система предназначена не только для настольных компьютеров или смартфонов, но и для реализации сквозных технологий национальной программы цифровой экономики РФ, прежде всего Big Data и машинного обучения технологий искусственного интеллекта.

Управление операционной системой для локального компьютера определяется при установке и дальнейшем изменении параметров в интересах пользователя и выполняется самим пользователем.

В учебном пособии рассмотрены SQL Server 2022 — инновационная версия популярной системы управления базами данных, вопросы производительности и безопасности, использование контейнеров и технологии Kubernetes, работа с кластерами больших данных и средства машинного обучения.

Изучение новых функций SQL Server 2022 позволит студентам расширить свои навыки в области управления и извлечения информации из больших данных, машинного обучения ML.NET и технологий искусственного интеллекта.

Для сетевых систем рассмотрены возможности инновационной операционной системы ArubaOS-CX — операционной системы для коммутаторов ядра и агрегации предприятий, которая автоматизирует и упрощает многие критические и сложные сетевые задачи, обеспечивает повышенную отказоустойчивость и позволяет проще справляться с непредсказуемыми событиями.

Рекомендуемая литература

1. *Андреев А. Е., Кузнецов М. А., Абдрахманов Д. Л.* Администрирование операционных систем: учеб. пособие. Волгоград: ВолгГТУ, 2021. 100 с.

2. *Бен-Ган И., Сарка Д., Талмейдж Р.* Microsoft SQL Server 2012. Создание запросов. Учебный курс Microsoft: пер. с англ. М.: Русская редакция, 2014. 720 с.

3. *Береснев А. Л.* Администрирование GNU/Linux с нуля. 2-е изд., перераб. и доп. СПб.: БХВ-Петербург, 2010. 576 с.

4. *Жуматий С. А., Стефанов К. С.* Суперкомпьютеры: администрирование. ЛитРес: Самиздат, 2018. 610 с.

5. *Кенин А. М., Колисниченко Д. Н.* Самоучитель системного администратора. 5-е изд., перераб. и доп. СПб.: БХВ-Петербург, 2019. 608 с.

6. *Левицкий Н. Д.* Справочник системного администратора. Полное руководство по управлению Windows-сетью. СПб.: Наука и техника, 2020. 464 с.

7. *Митричев И. И.* Администрирование операционных систем. Конспект лекций: учеб. пособие. М.: РХТУ имени Д. И. Менделеева, 2019. 156 с.

8. *Силен Д., Мейсман А., Али М.* Основы Data Science и Big Data. Python и наука о данных. СПб.: Питер, 2018. 336 с.

9. *Станек У.* Справочник администратора. Internet Information Services (1 IS) 7.0.: пер. с англ. М.: Русская Редакция; СПб.: БХВ-Петербург, 2013. 528 с.

10. *Таненбаум Э., Бос Х.* Современные операционные системы. 4-е изд. СПб.: Питер, 2015. 1120 с.

11. *Уорд Б.* Инновации SQL Server 2019. Использование технологий больших данных и машинного обучения / пер. с англ. Н. Б. Желновой. М.: ДМК Пресс, 2020. 408 с.

12. *Microsoft* Windows Server 2012. Полное руководство / Р. Моримото, М. Ноэл, Г. Ярдени и др.: пер. с англ. М.: ИД Вильямс, 2013. 1456 с.

Информация об авторах

Часовских Виктор Петрович — профессор кафедры шахматного искусства и компьютерной математики УрГЭУ, доктор технических наук, профессор
e-mail: u2007u@ya.ru

Лабунец Валерий Григорьевич — профессор кафедры шахматного искусства и компьютерной математики УрГЭУ, доктор технических наук, профессор

Стариков Евгений Николаевич — заведующий кафедрой шахматного искусства и компьютерной математики УрГЭУ, кандидат экономических наук, доцент
e-mail: starikov_en@usue.ru

Акчурина Галия Абдулазисовна — старший преподаватель кафедры шахматного искусства и компьютерной математики УрГЭУ
e-mail: mstrk@yandex.ru

Кох Елена Викторовна — доцент кафедры шахматного искусства и компьютерной математики УрГЭУ, кандидат сельскохозяйственных наук, доцент
e-mail: elenakox@mail.ru

Оглавление

Учебное издание

Часовских Виктор Петрович,
Лабунец Валерий Григорьевич,
Стариков Евгений Николаевич
и др.

Администрирование операционных систем

Учебное пособие

Редактор и корректор *П. А. Давыдова*
Компьютерная верстка *И. В. Засухиной*

Поз. 38. Подписано в печать 28.12.2023.
Формат 60 × 84 1/16. Бумага офсетная. Печать плоская.
Уч.-изд. л. 5,2. Усл. печ. л. 7,7. Печ. л. 8,3. Заказ 65. Тираж 24 экз.
Издательство Уральского государственного экономического университета
620144, г. Екатеринбург, ул. 8 Марта / Народной Воли, 62/45

Отпечатано с готового оригинал-макета в подразделении оперативной полиграфии
Уральского государственного экономического университета

УРАЛЬСКИЙ ГОСУДАРСТВЕННЫЙ
ЭКОНОМИЧЕСКИЙ УНИВЕРСИТЕТ