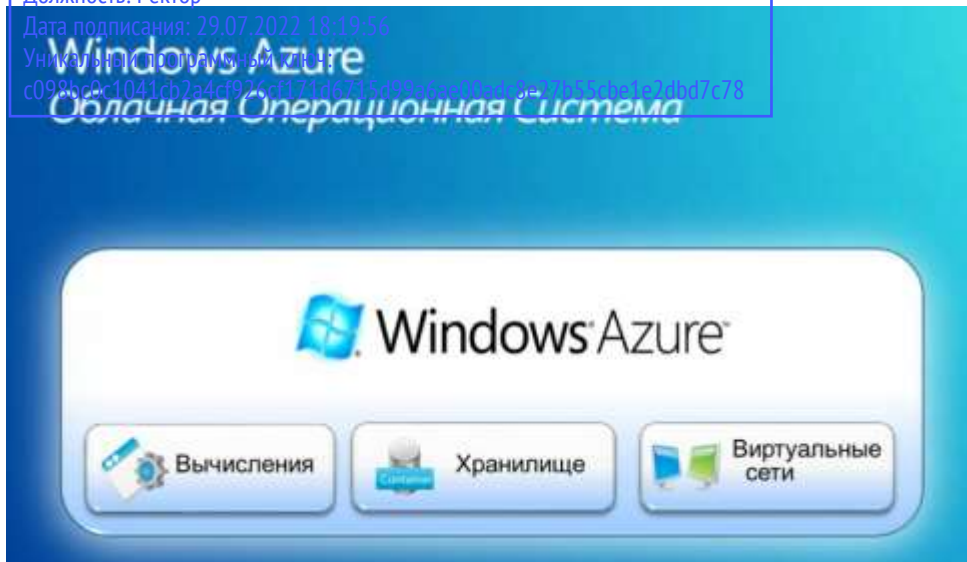


Документ подписан простой электронной подписью
Информация о владельце:
ФИО: Макаренко Елена Николаевна
Должность: Ректор

Дата подписания: 29.07.2022 18:19:56

Уникальный программный ключ:
с098bc9c1041cb2a4cf926cf171d6715d99a6ae00a1dfe71955cbe1e7dbd7c78



Windows Azure: Хранилище

Масштабируемое хранилище в облаке <ul style="list-style-type: none">• 100 TB на аккаунт• Автоматически изменяемое в соответствии с различными вариантами использования данных	Доступное через RESTful Web services <ul style="list-style-type: none">• Доступ из Windows Azure Приложений• Доступ из произвольного места в Internet• Поддержка .NET Client Library	Различные типы хранилища <ul style="list-style-type: none">• Tables• Blobs• Queues• Drives
---	---	--

СИСТЕМЫ АНАЛИТИКИ БОЛЬШИХ ДАННЫХ

ПРАКТИЧЕСКИЕ РАБОТЫ

I. ЦЕЛИ И ЗАДАЧИ ОСВОЕНИЯ ДИСЦИПЛИНЫ

Цели освоения дисциплины:

формирование у обучающихся способности осуществлять руководство проектами по созданию, поддержке и использованию комплексных систем на основе аналитики больших данных с применением новых методов и алгоритмов машинного обучения.

Задачи освоения дисциплины:

- формирование у обучающихся знаний о методологии и принципах руководства проектами по созданию, поддержке и использованию комплексных систем на основе аналитики больших данных со стороны заказчика;
- развитие у обучающихся умения применять современные инструментальные средства и системы программирования для разработки новых методов и моделей машинного обучения;
- развитие у обучающихся умения решать задачи по руководству коллективной проектной деятельностью для создания, поддержки и использования комплексных систем на основе аналитики больших данных со стороны заказчика.

II. СОДЕРЖАНИЕ И СТРУКТУРА ДИСЦИПЛИНЫ

Трудоёмкость дисциплины составляет 5 зачётных единиц, 180 часов.

в том числе 1 зачётная единица, 36 часов на экзамен.

Форма промежуточной аттестации: экзамен.

№ п/ п	Темы дисциплины	Семестр	Виды учебной работы и их трудоёмкость, часы (в том числе с использованием онлайн-курсов)				Наименования оценочных средств
			Контактная работа			Самостоятельная работа	
			Лекции и	Практические занятия	Лабораторные занятия		
Модуль 1. Основы построения и работы с системами аналитики больших данных							
1	Архитектура систем аналитики больших данных.	3	2	-	-	6	Практическая работа №1
2	Хранение больших данных в облаке. 1. Хранилища общего назначения. Форматы хранения данных. Облачное хранилище Microsoft Azure Storage. Облачное хранилище AWS. 2. Реляционные базы данных. Azure SQL. AWS RDS. 3. Нереляционные базы данных. Сервисы нереляционных баз данных от Azure и AWS. 4. Корпоративные хранилища данных (DWH). Azure SQL DWH. AWS RedShift. 5. Хранилища данных типа Data Lake. Azure Data Lake Store. AWS Data Lake Solutions.	3	4	12	-	28	Практическая работа №1

II. СОДЕРЖАНИЕ И СТРУКТУРА ДИСЦИПЛИНЫ

Модуль 2. Разработка систем аналитики больших данных							
3	<p>Доставка больших данных в облако.</p> <p>1. Прямая загрузка данных. Доставка данных в облачное хранилище общего назначения. Доставка данных в реляционные БД и хранилища. Доставка данных в нереляционные базы данных. Доставка данных в HDFS-совместимые хранилища.</p> <p>2. Прямая загрузка потоковых данных. Azure Event Hub. AWS Kinesis Data Streams. Облачные сервисы развертывания кластерных систем.</p> <p>3. Облачные сервисы копирования и трансформации данных. Azure Data Factory. AWS Data Pipeline. AWS Glue.</p>	3	6	12	-	28	Практическая работа №2
4	<p>Аналитика больших данных в облаке.</p> <p>1. Интерактивный анализ данных. Анализ реляционных данных. Azure Data Lake Analytics. AWS Athena. Apache Spark. Встроенные редакторы запросов сервиса CosmosDB</p> <p>2. Потоковый анализ данных. Общие сведения. Azure Stream Analytics. Amazon Kinesis Analytics. Apache Storm.</p> <p>3. Пакетный анализ данных. Hadoop. Apache Pig. Apache Hive.</p>	3	6	12	-	28	Практическая работа №3
	Промежуточная аттестация (для дисциплин с экзаменом)	3	-	-	-	36	Экзаменационные вопросы и билеты
	Итого часов	3	18	36	-	126	-

Практическая работа №1

Развертывание облачного хранилища общего назначения

Облачное хранилище Microsoft Azure Storage состоит из четырех сервисов: BLOB Storage, Queue Storage, Table Storage и File Storage. Непосредственно хранение информации осуществляется в сервисах BLOB, Table и File Storage. Queue Storage — это облачный сервис обмена сообщениями и синхронизации распределенных приложений. Сервис Table Storage — база данных NoSQL типа «ключ — значение».

1. Разверните облачное хранилище Microsoft Azure Storage. Получите доступ к панели мониторинга и управления. Организуйте работу с хранилищем BLOB на веб-портале.

В отличие от Azure у AWS хранилища не объединяются логически в Storage Account, а представляют собой набор отдельных сервисов. Хранение двоичных объектов обеспечивается сервисом AWS S3 – Amazon Simple Storage Service. Экземпляр сервиса AWS S3 может быть логически разбит на бакеты, которые будут определять видимость объектов в них, шифрование, жизненный цикл и т.д.

2. Разверните сервис AWS S2. Создайте бакеты и настройте права доступа к ним. Разверните сервис облачного сетевого хранилища Amazon Elastic File. Разверните сервис AWS EBS (Elastic Block Storage) для создания, размещения, шифрования, подключения к виртуальным машинам, создания бэкапов в виде мгновенных снимков (snapshots).

Сравните возможности и ограничения облачных хранилищ общего назначения Azure BlobStorage и AWS S3 для решения прикладных задач.

Практическая работа №2

Развертывание облачных сервисов копирования и трансформации данных

Сервисы трансформации и копирования играют важную роль в построении информационных систем анализа больших данных. Требования к сервисам хранения и анализа противоречивы и не могут быть удовлетворены в рамках одного хранилища, особенно когда данных много. Чтобы удовлетворить эти требования, необходимо использовать хранилища разных типов и сервис, который связывает их информационно.

Сервис AWS Glue представляет собой удобное средство каталогизации, трансформации и копирования данных, что позволяет одновременно создавать каталоги данных и использовать их для ETL.

Задание:

1. На веб-портале Azure создайте и сконфигурируйте сервис [Azure Data Factory](#). Предложите решение задачи копирования данных из таблицы Azure Table Storage в реляционную БД Azure SQL.
2. Разверните сервис [AWS Data Pipeline](#) для автоматизации копирования и трансформации данных.
3. Разверните бессерверный сервис [AWS Glue](#) для категоризации данных в каталог данных и выполнения задач ETL. Предложите решение задачи копирования в AWS Redshift данных в виде файла JSON, хранящегося в AWS S3

Практическая работа №3

Развертывание облачных сервисов для потокового анализа данных

Потоковый анализ состоит в анализе потока в виде последовательности сообщений, то есть в применении алгоритма анализа к каждому сообщению потока. Этот анализ может состоять в выделении из всего потока сообщений, соответствующих определенным признакам, например, сообщений об ошибках, которые должны быть направлены в отдельное хранилище или сервис, занимающийся обработкой ошибок. Пример прикладной задачи – анализ банковских транзакций и определение в этом потоке потенциально подозрительных (выдача большой суммы в нетипичной для плательщика стране или выдача мелких сумм в разных банкоматах в течение малого интервала времени).

1. Разверните облачный сервис Azure Stream Analytics для потоковой обработки Microsoft-сообщений. Создайте экземпляр потокового задания на веб-портале. Обеспечьте интеграцию с сервисом Azure ML: для этого обучите модель с помощью сервиса Azure ML, а затем, после ее размещения в качестве REST-сервиса, получите ее адрес и используйте для интеграции с Azure Stream Analytics.
2. Разверните сервис потокового анализа данных AWS Kinesis Analytics. Подключите входной источник. Определите схему сообщений, поступающих на вход сервиса. Сконфигурируйте приложение.

Сравните возможности и ограничения облачных сервисов для потокового анализа данных Azure Stream Analytics и AWS Kinesis Analytics для решения прикладных задач.

Windows Azure

Облачная Операционная Система

 Windows Azure™



Вычисления



Хранилище



Виртуальные
сети

Windows Azure: Хранилище

Масштабируемое хранилище в облаке

- 100 TB на аккаунт
- Автоматически изменяемое в соответствии с различными вариантами запросов на обработку или использование данных

Доступное через RESTful Web services

- Доступ из Windows Azure Приложений
- Доступ из произвольного места в internet
- Поддержка .NET Client Library

Различные типы хранилища

- Tables
- Blobs
- Queues
- Drives

СИСТЕМЫ АНАЛИТИКИ БОЛЬШИХ ДАННЫХ

МЕТОДИЧЕСКИЕ УКАЗАНИЯ К
ВЫПОЛНЕНИЮ ПРАКТИЧЕСКИХ РАБОТ

Практическая работа №1

Облачное хранилище Microsoft Azure Storage

Облачное хранилище

Microsoft Azure Storage

Рассмотрим, как реализовано облачное хранилище, на примере Microsoft Azure Storage.

Оно состоит из четырех сервисов:

BLOB Storage, Queue Storage, Table Storage и File Storage (рис. 4.4).

Непосредственно хранение информации осуществляется в сервисах BLOB, Table и File Storage.

Queue Storage — это облачный сервис обмена сообщениями и синхронизации распределенных приложений. Сервис Table Storage — база данных NoSQL типа «ключ — значение».

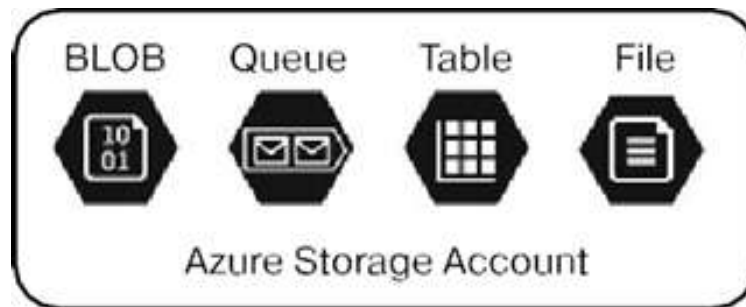


Рис. 4.4. Виды сервисов Azure Storage Account

Облачное хранилище Microsoft Azure Storage

Рассмотрим подробнее, как создавать *Azure Storage Account* и управлять с веб-портала.

Прежде всего необходимо нажать ссылку добавления новых ресурсов Azure, расположенную в левом верхнем углу, — [+ New](#).

Затем в открывшемся окне (рис. 4.5) выбрать последовательно *Storage* — *Storage account*. После нажатия ссылки *Storage account* откроется форма настройки *Storage Account* (рис. 4.6).

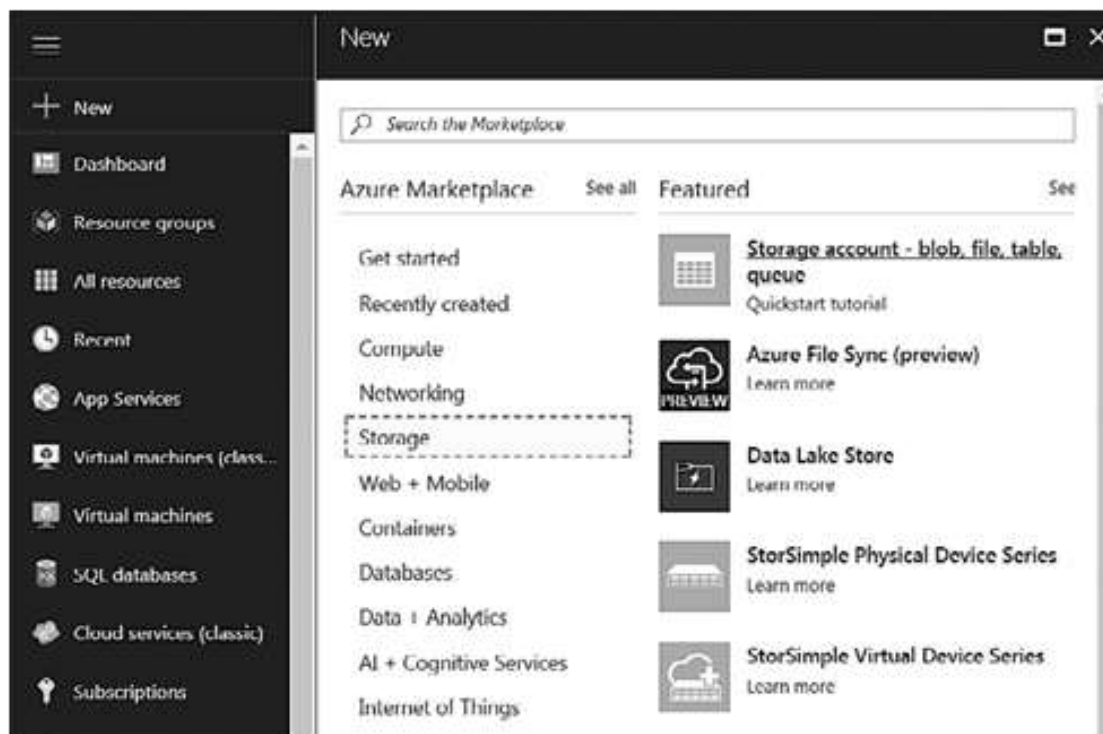


Рис. 4.5. Добавление нового Storage account через веб-портал

Облачное хранилище Microsoft Azure Storage

После нажатия ссылки Storage account откроется форма настройки Storage Account (рис. 4.6).

The image shows two side-by-side screenshots of the 'Create storage account' form in the Azure portal. The left screenshot shows the form with a scrollbar on the right side, indicating it is scrollable. The right screenshot shows the same form but with the scrollbar moved to the bottom, revealing more options. Both forms contain the following fields and options:

- Name:** pokermainstorage (with a dropdown arrow)
- Deployment model:** Resource manager (selected), Classic
- Account kind:** Storage (general purpose v1) (with a dropdown arrow)
- Performance:** Standard (selected), Premium
- Replication:** Locally-redundant storage (LRS) (with a dropdown arrow)
- Secure transfer required:** Disabled (selected), Enabled
- Subscription:** Pay-As-You-Go (with a dropdown arrow)
- Resource group:** Create new (radio button), Use existing (radio button, selected), PockerRumExample (with a dropdown arrow)
- Location:** Central US (with a dropdown arrow)
- Virtual networks:** Configure virtual networks (with a dropdown arrow), Disabled (selected), Enabled
- Pin to dashboard:**
- Create:** Button
- Automation options:** Link

Рис. 4.6. Форма настройки Storage Account (показана со сдвигом ползунка)

Облачное хранилище

Microsoft Azure Storage

После нажатия ссылки Storage account откроется форма настройки Storage Account (рис. 4.6).

[В ней доступны следующие конфигурации:](#)

- Имя аккаунта (поле Name) будет частью URL аккаунта <Name>.core.windows.net, а потому правила наименования ресурсов точно такие же, как и в случае именования ресурсов, доступных по URL.
- Тип модели развертывания ресурсов (deployment model) — выбираем Resource Manager, чтобы иметь возможность добавить аккаунт к общей ресурсной группе PockerRunExample.
- Тип Storage (Account kind) — Storage (general purpose v1). Помимо этого, доступны типы Storage (general purpose v2) и BLOB.
- Выбираем уровень производительности (Performance) — Standard. В зависимости от выбранного уровня производительность операций чтения-записи будет отличаться. Наиболее высоким уровнем будет обладать комбинация Account kind = BLOB, Performance = Premium. Для premium-уровня в качестве физических устройств хранения выступают SSD-диски.
- В качестве режима репликации (Replication) выбираем Locally-Redundant storage (LRS). Эта опция означает, что информация, хранящаяся в Storage Account, физически реплицируется три раза, но в пределах одного дата-центра.

Помимо LRS, доступны различные режимы географической репликации в разных дата-центрах в пределах одной зоны (ZRS) или среди нескольких различных зон (GRS и ReadOnly-GRS — репликация по различным географическим зонам с репликой, доступной только для чтения). Все эти режимы различаются по стоимости и уровню надежности и доступности, что позволяет реализовать облачные хранилища, отвечающие различным наборам требований.

После создания Storage Account доступна следующая панель мониторинга и управления (рис. 4.7).

Облачное хранилище Microsoft Azure Storage

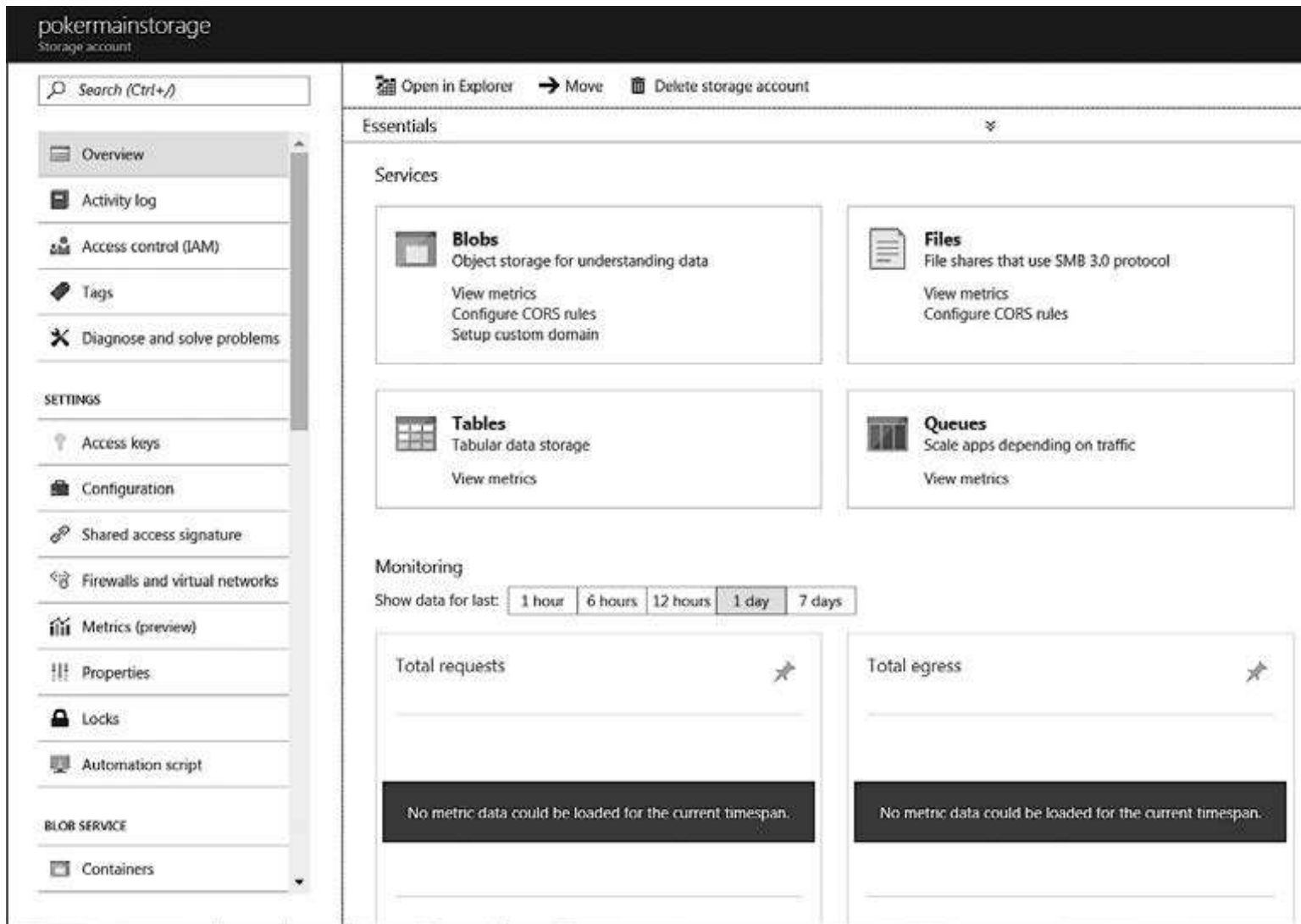


Рис. 4.7. Общая панель мониторинга и управления Storage Account

Облачное хранилище

Microsoft Azure Storage

На этой панели доступны все четыре сервиса, входящие в состав Storage Account: Blob, File, Table и Queue.

Кратко рассмотрим возможности конфигурирования Storage Account в целом. Прежде всего, доступен File Explorer — бесплатная программа от Microsoft, позволяющая просматривать содержимое всех сервисов всех аккаунтов хранения.

Ее внешний вид (рис. 4.8). На рисунке можно увидеть результат выборки программы из хранилища Table Storage (таблица events), которая содержит тестовые данные телеметрии метеостанции, полученные путем приема сообщений через сервисы концентратора EventHub и потоковой аналитики Stream Analytics Job.

Чтобы обеспечить доступ к аккаунту извне, необходимо знать ключи доступа и URL, которые доступны на вкладке Access Keys (рис. 4.9).

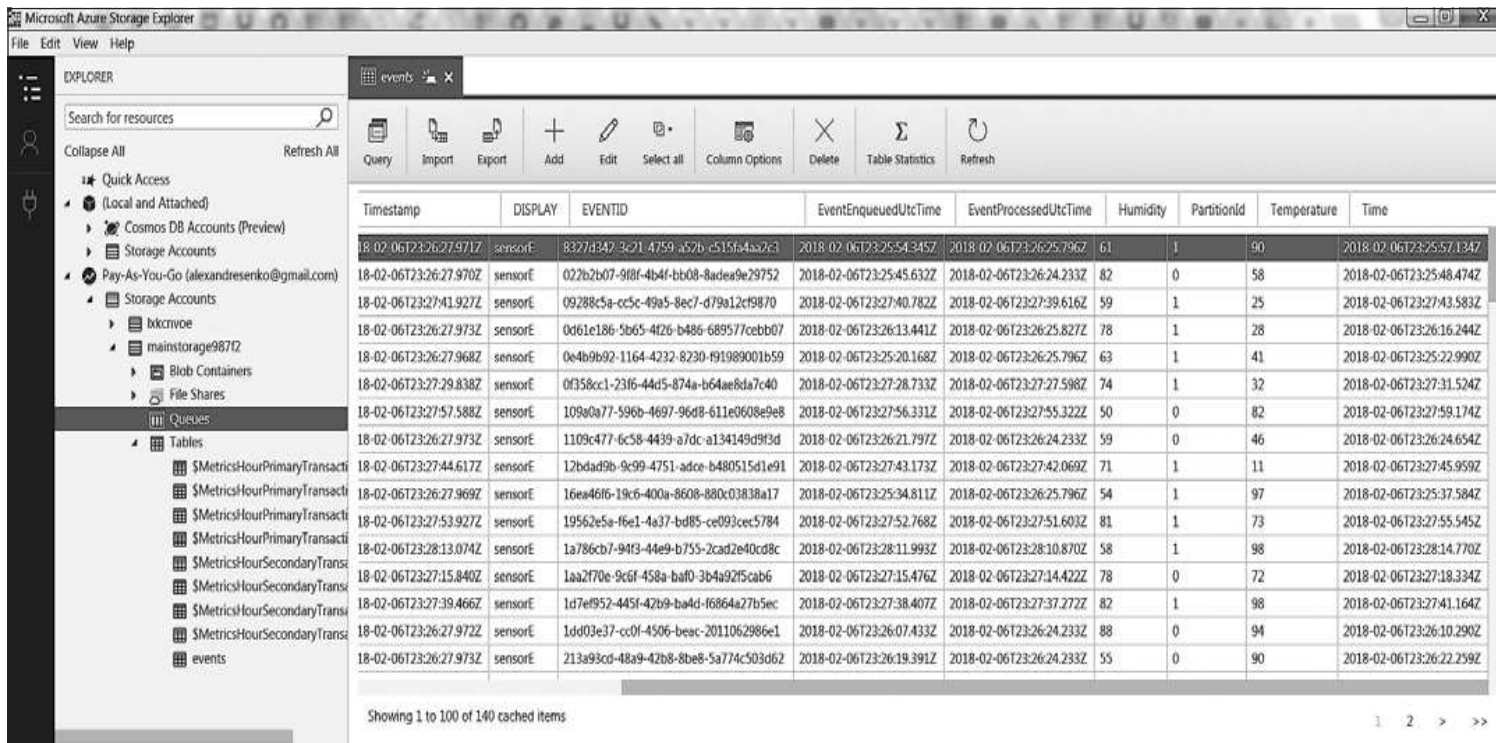


Рис. 4.8. Внешний вид программы File Explorer от Microsoft



Рис. 4.9. Ключи доступа и строки подключения к Storage Account

Облачное хранилище

Microsoft Azure Storage

Две пары ключей или строк подключения нужны для обеспечения «бесшовного» обновления ключей. Для этого первоначально используется ключ key1, затем клиенты переключаются на ключ key2, а ключ key1 обновляется. После этого клиенты переключаются на новый ключ key1, а ключ key2 обновляется.

Страница конфигурирования аккаунта в целом показана на рис. 4.10. Она позволяет менять тип аккаунта (Account kind), уровень производительности (Performance). К подобным манипуляциям следует относиться внимательно, поскольку каждый уровень имеет различную стоимость, а также переключение между ними может занять время, если в аккаунте много данных.

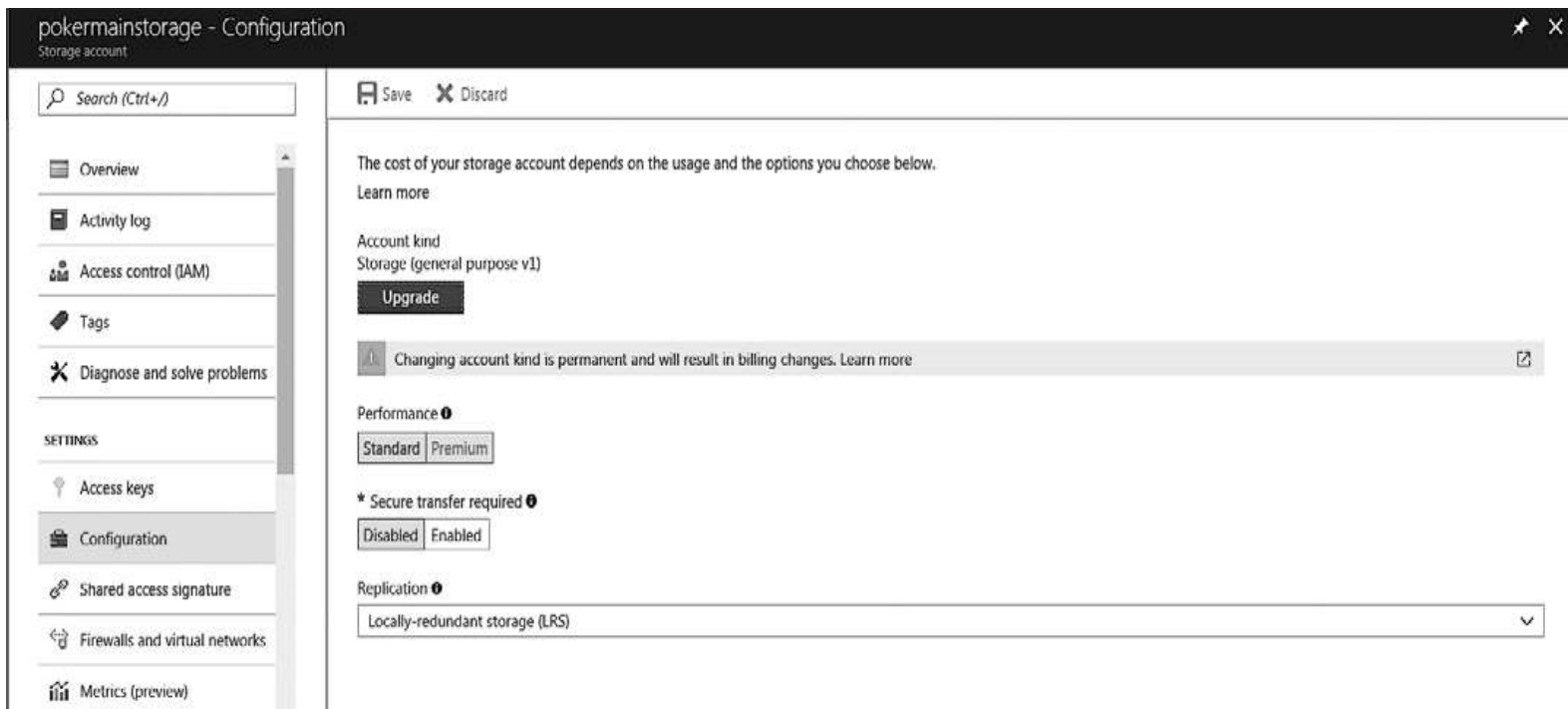


Рис. 4.10. Вкладка конфигурирования типа аккаунта, уровня производительности и репликации

Сервисы хранения

Microsoft Azure Storage

Следующий важный конфигурируемый параметр — Shared access signature (SAS) (рис. 4.11). Концепция SAS состоит в том, что к объектам, расположенным в Storage Account, можно предоставить прямой доступ для скачивания — с помощью URL, который содержит ряд ограничивающих параметров, а именно: время жизни ссылки и ограничения IP-адресов, с которых доступен ресурс. Эти параметры подписываются ключом аккаунта, и данная подпись добавляется в конец URL, по которому данный ресурс может быть доступен.

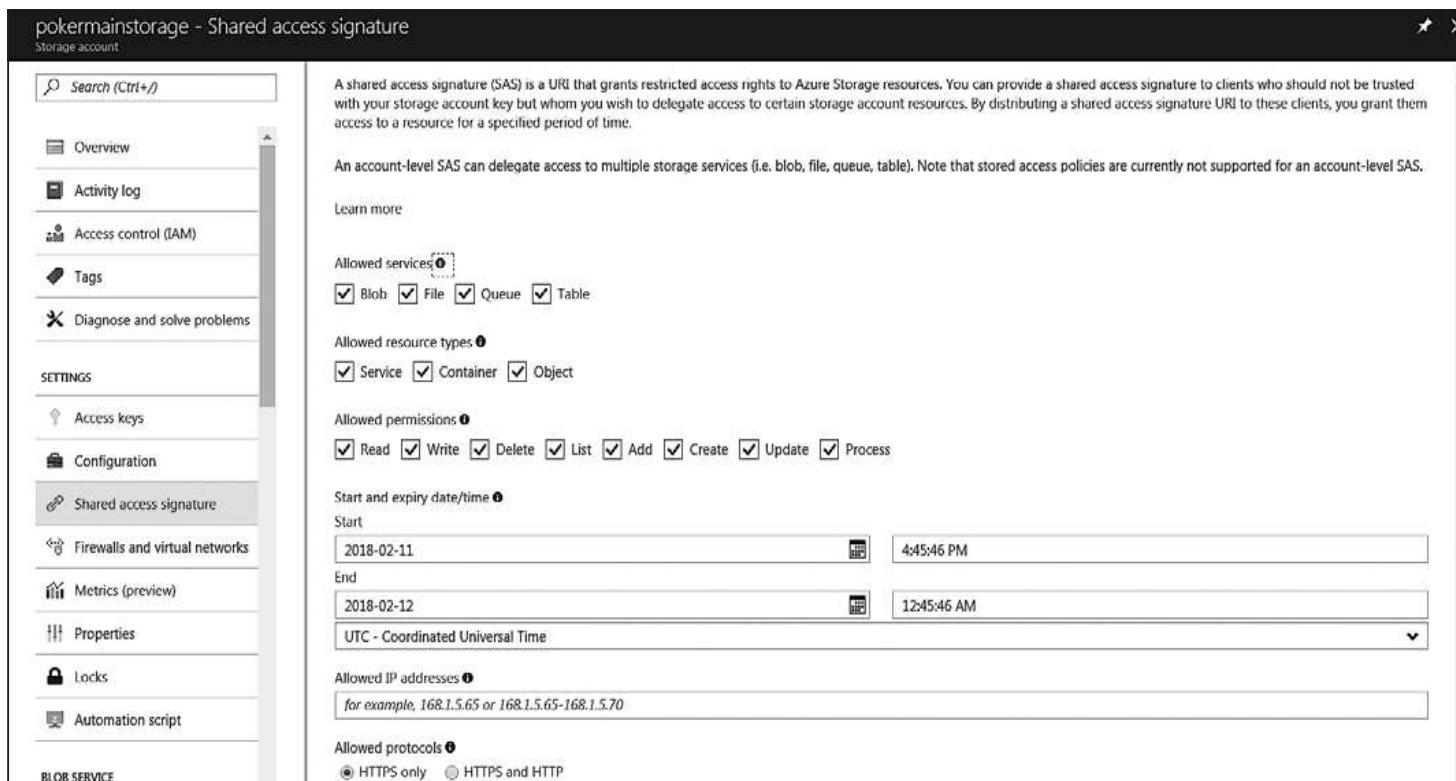


Рис. 4.11. Вкладка настройки параметров Shared Access Signature

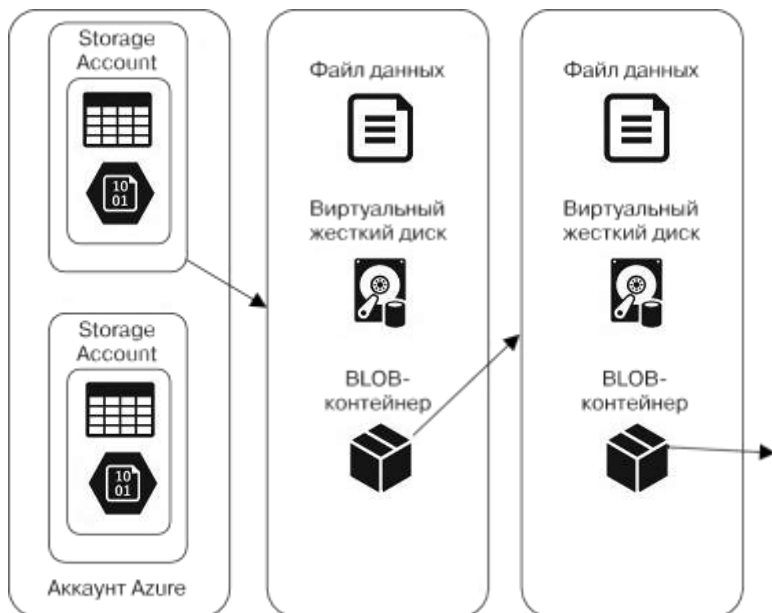
Сервисы хранения

Microsoft Azure Storage

Теперь подробнее познакомимся с отдельными сервисами хранения файлов Storage Account.

Сервис Azure BLOB Storage предназначен для хранения различных файлов, поэтому и называется хранилищем больших двоичных объектов (Binary Large Objects Storage, BLOB). Двоичные объекты могут храниться в нем как непосредственно, так и будучи размещенными в контейнерах (не путайте с Docker-контейнерами, речь идет о контейнерах BLOB Storage) или на виртуальных дисках, тоже расположенных в BLOB Storage Account.

Чтобы прояснить ситуацию, рассмотрим рис. 4.12.



Аккаунт Azure может содержать один или несколько Storage-аккаунтов. Каждый такой аккаунт способен непосредственно хранить файлы, виртуальные жесткие диски и контейнеры BLOB. Последние, в свою очередь, тоже могут включать файлы, виртуальные жесткие диски и другие контейнеры. Таким образом, с помощью контейнеров BLOB реализована иерархическая структура организации файлов.

Рис. 4.12. Структура вложенности объектов в хранилище BLOB

Сервисы хранения

Microsoft Azure Storage

В **BLOB Storage** могут храниться любые файлы: текстовые, двоичные и др., но для разных типов хранимых объектов требуется разный тип BLOB.

Всего есть три типа BLOB: **Page BLOB**, **Block BLOB** и **Append BLOB**.

Page BLOB — бинарный объект со страничной организацией памяти. Этот тип используется только для размещения виртуальных жестких дисков виртуальных машин.

Block BLOB — BLOB с блочной организацией памяти, служащий для хранения всех видов файлов (кроме VHD), включая контейнеры BLOB. Это основной тип хранения файлов в BLOB Storage обычных файлов.

Append Block — BLOB с блочной организацией памяти, представляющий собой текстовый файл, размещенный в Azure Storage и допускающий добавление новой записи в конец файла. В остальных типах BLOB файлы не редактируемые, то есть, чтобы отредактировать файл, его необходимо скачать, открыть, отредактировать и загрузить обратно. Понятно, что эти действия сопряжены с большими трудностями при работе с файлами логов. В то же время Append Block как раз оптимизирован для сценариев прямой записи логов.

Все объекты, расположенные в BLOB Storage, могут быть доступны через веб-портал, с помощью SDK, команд расширения командной оболочки, а также по прямой ссылке.

Сервисы хранения

Microsoft Azure Storage

Доступ к Storage-аккаунту с помощью веб-портала позволяет создавать файлы, контейнеры, просматривать список файлов, добавлять, удалять и загружать их, менять области видимости файлов (они могут быть общедоступными по ссылке, закрытыми для всех, кроме сервисов Azure).

Интерфейс веб-портала удобен для работы с небольшим количеством файлов. Добавлять в облачное хранилище через веб-портал можно файлы не слишком большого размера. А вот файлы, загружаемые из облака, могут быть любого размера, допустимого в хранилище. Кроме того, для файлов можно открыть общий доступ, и они станут доступными для скачивания по ссылке (эта ситуация отражена на рис. 4.13).

Доступ может быть открыт для файлов как в корневом каталоге, так и в контейнерах внутри хранилища.

Способы доступа к файлам в облачном хранилище BLOB

- Для программного доступа облачный аккаунт содержит REST API, который, в свою очередь, через SDK предоставляет гораздо большие возможности: синхронную и асинхронную загрузку и выгрузку, удаление, создание, добавление в конец Append BLOB и пр.
- Кроме того, через SDK можно создать временную ссылку на файл, то есть ссылку, становящуюся нерабочей через определенный промежуток времени.

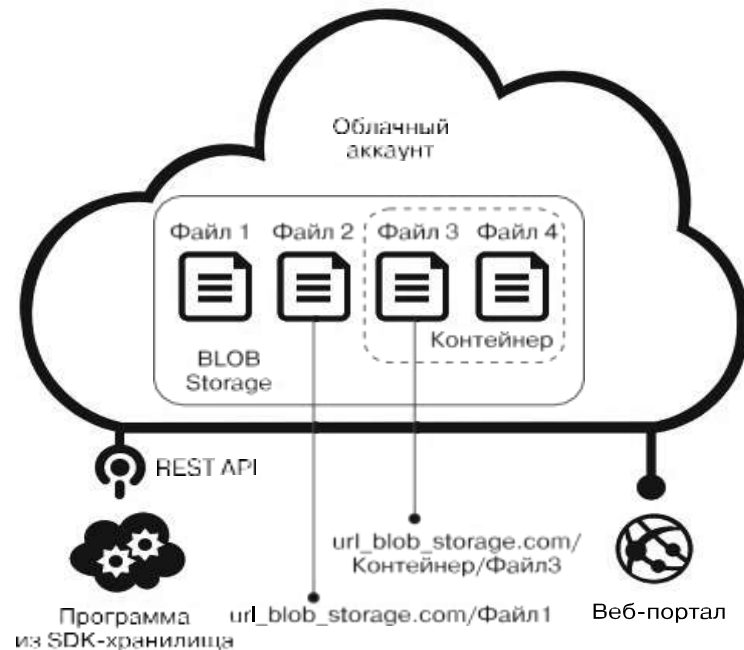


Рис. 4.13. Способы доступа к файлам в облачном хранилище BLOB

Сервисы хранения

Microsoft Azure Storage

Рассмотрим подробнее, как работать с хранилищем BLOB с помощью веб-портала.

- Чтобы перейти к хранилищу BLOB, необходимо нажать ссылку Blobs на общей панели аккаунта (см. рис. 4.7).
- В результате откроется вкладка, показанная на рис. 4.14.

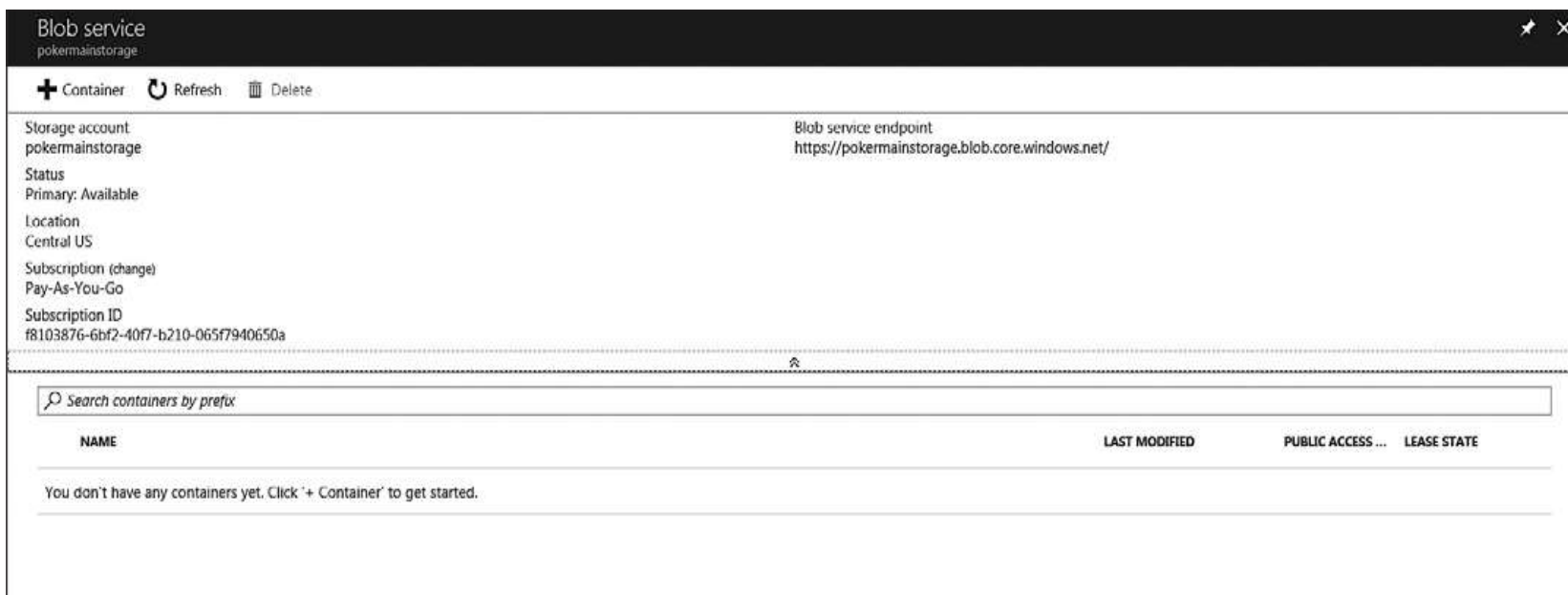


Рис. 4.14. Общая панель сервиса BLOB Storage

Сервисы хранения

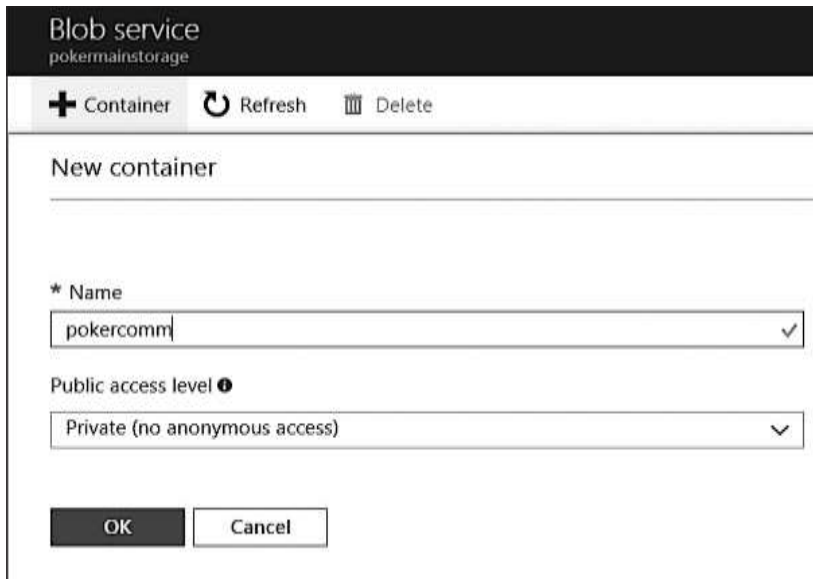
Microsoft Azure Storage

Далее требуется добавить контейнеры.

Для этого необходимо нажать ссылку **+ Container** (рис. 4.15).

В данной форме указано **имя (свойство Name)** — **pokercomm**; **уровень доступа (Public access level)** — **Private (no anonymous access)**. Этот тип доступа подразумевает доступ не по прямой ссылке, а только с помощью ключей аккаунта с использованием SDK.

Другие варианты типа доступа: **Blob (anonymous read access for blobs only)** — разрешает анонимный доступ только к файлам (BLOB); **Container (anonymous read access for containers and blobs)** — разрешен анонимный доступ к контейнерам и файлам. Вкладка созданного контейнера выглядит следующим образом (рис. 4.16).



Blob service
pokermainstorage

+ Container Refresh Delete

New container

* Name
pokercomm ✓

Public access level ⓘ
Private (no anonymous access) ▾

OK Cancel



Рис. 4.16. Вкладка созданного контейнера

Рис. 4.15. Форма создания нового контейнера

Сервисы хранения

Microsoft Azure Storage

- Загрузить файл в контейнер можно с помощью ссылки Upload. Свойства контейнера доступны по ссылке Container properties и показаны на рис. 4.17. Они включают в себя имя, адрес, статус, количество и суммарный размер BLOB-объектов, статус.
- Для контейнера доступна настройка политики доступа через вкладку Access policy (рис. 4.18). Эта настройка позволяет организовать различный уровень доступа к различным контейнерам и объектам BLOB.
- Помимо упомянутых настроек, для BLOB доступен ряд других (рис. 4.19).

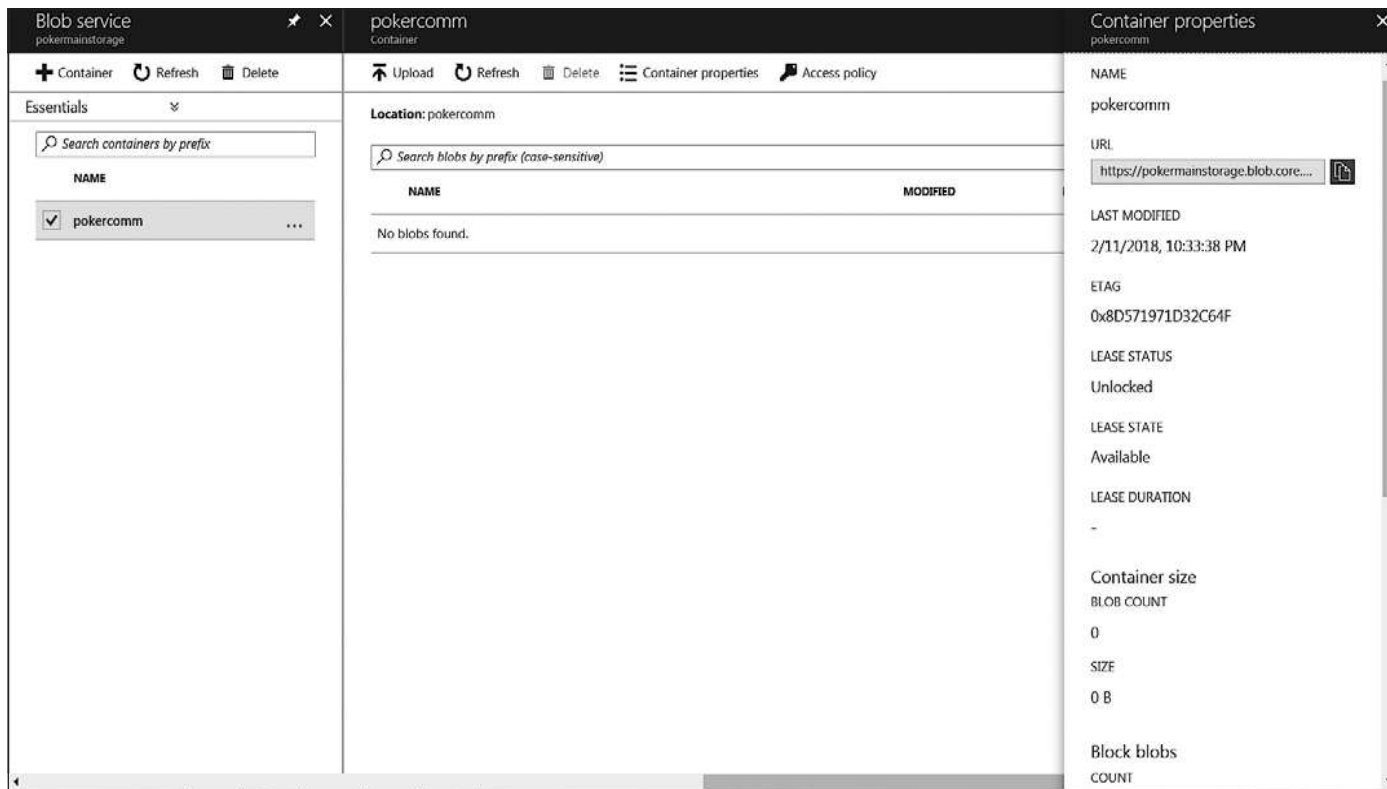


Рис. 4.17. Вкладка свойств контейнера

Сервисы хранения

Microsoft Azure Storage

- Загрузить файл в контейнер можно с помощью ссылки Upload. Свойства контейнера доступны по ссылке Container properties и показаны на рис. 4.17. Они включают в себя имя, адрес, статус, количество и суммарный размер BLOB-объектов, статус.
- Для контейнера доступна настройка политики доступа через вкладку Access policy (рис. 4.18). Эта настройка позволяет организовать различный уровень доступа к различным контейнерам и объектам BLOB.
- Помимо упомянутых настроек, для BLOB доступен ряд других (рис. 4.19).

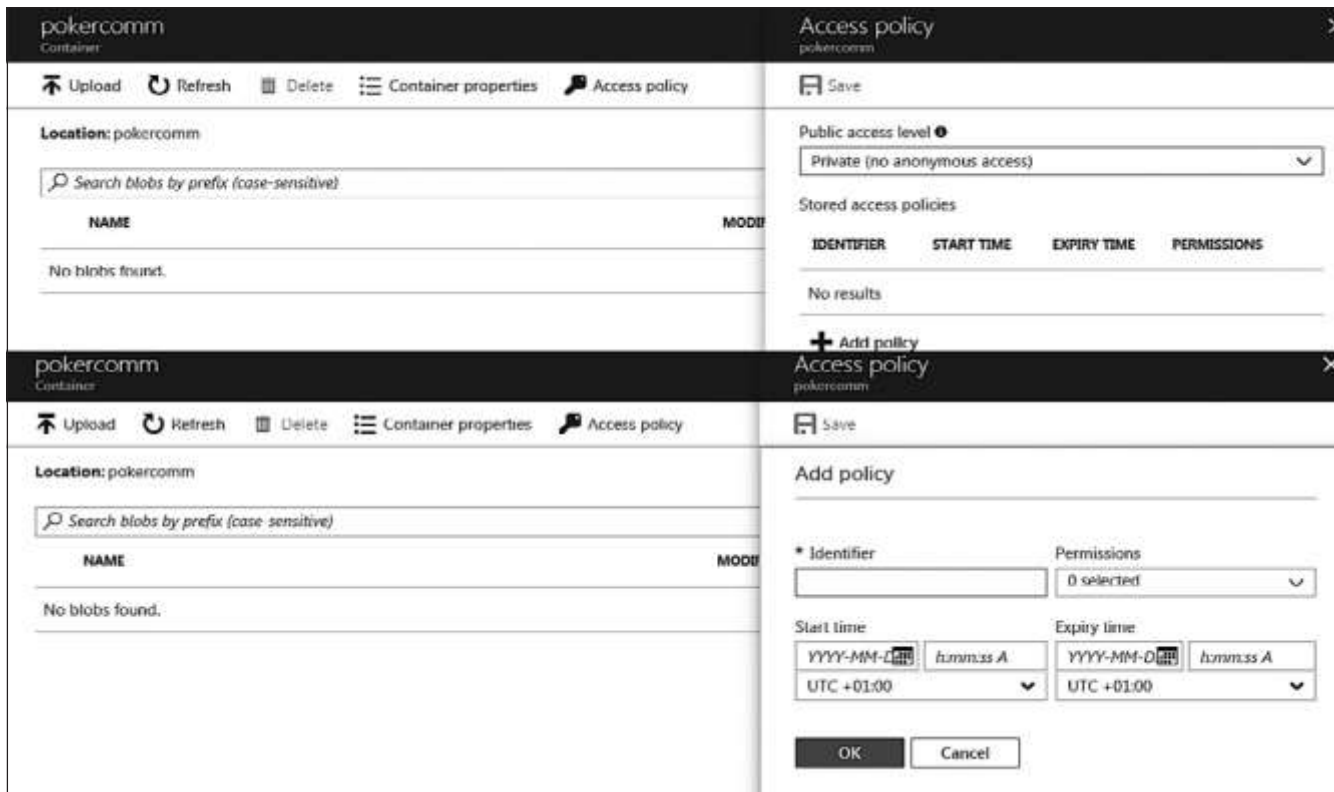


Рис. 4.18. Вкладка настройки политики доступа

Сервисы хранения

Microsoft Azure Storage

- Загрузить файл в контейнер можно с помощью ссылки Upload. Свойства контейнера доступны по ссылке Container properties и показаны на рис. 4.17. Они включают в себя имя, адрес, статус, количество и суммарный размер BLOB-объектов, статус.
- Для контейнера доступна настройка политики доступа через вкладку Access policy (рис. 4.18). Эта настройка позволяет организовать различный уровень доступа к различным контейнерам и объектам BLOB.
- Помимо упомянутых настроек, для BLOB доступен ряд других (рис. 4.19).

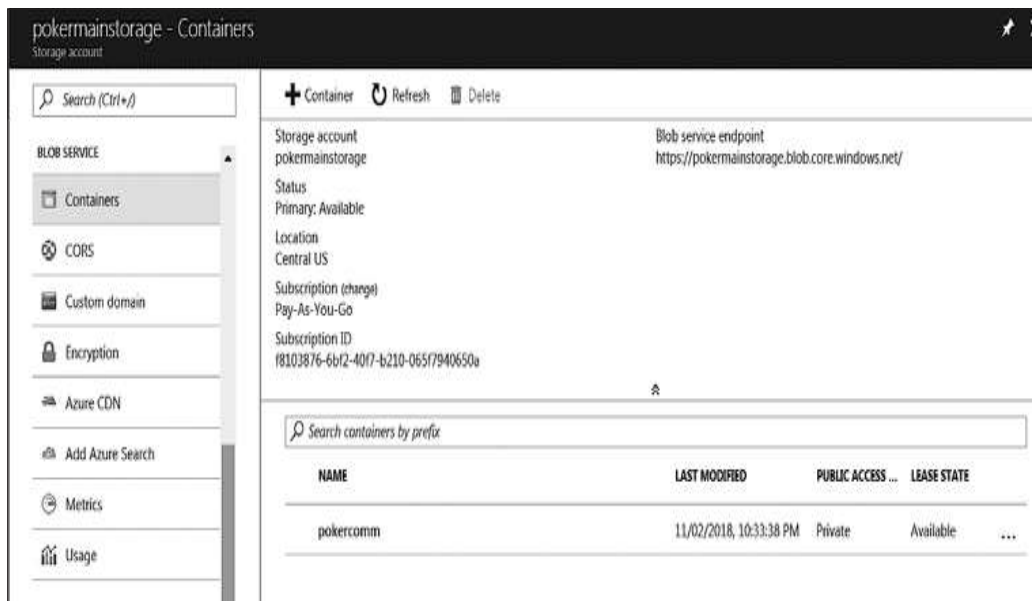


Рис. 4.19. Различные настраиваемые сервисы Blob Storage

Сервисы хранения

Microsoft Azure Storage

Самые важные параметры:

- CORS (cross-origin resource sharing — совместное использование ресурсов между разными источниками) — свойство, обеспечивающее доступ к ресурсам BLOB из другого домена;
- Custom domain — конфигурирование DNS-записей CNAME в целях указания домена пользователя, в дополнение к домену аккаунту Azure Storage;
- Encryption — шифрование объектов в хранилище;
- Azure CDN — конфигурирование Azure Content Delivery Network, которая служит для хранения часто используемого контента из хранилища BLOB с анонимным доступом.

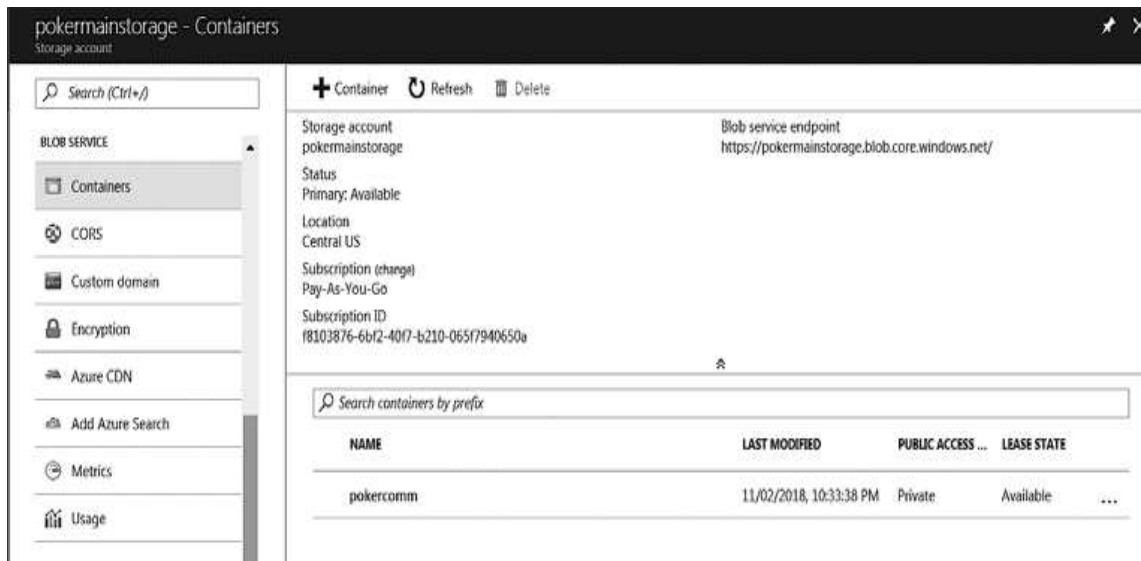


Рис. 4.19. Различные настраиваемые сервисы Blob Storage

Сервисы хранения

Microsoft Azure Storage

Доступ к объектам в BLOB Storage возможен по прямой ссылке URL или с помощью интерфейса REST API (напрямую либо через SDK). Данный способ хранения обеспечивает самый быстрый доступ (минимальное время загрузки/выгрузки), но требует применения специальных программных клиентов, взаимодействующих с этими API.

Последнее условие может помешать существующим приложениям большого масштаба мигрировать в облако. Кроме того, в ряде случаев нужно создать облачное хранилище, которое должно быть доступно из виртуальной машины без всяких «самописных» программных клиентов.

Для этого в хранилище BLOB можно разместить виртуальный жесткий диск (или набор дисков для RAID-массива) и примонтировать его к виртуальной машине.

- **Преимущество такого решения** состоит в том, что данные из виртуальной машины могут быть доступны точно таким же образом, как и с физического диска, подключенного к ней.
- **Недостаток такого способа** заключается в его низкой масштабируемости (верхний предел размера ограничен на уровне, определяемом операционной системой виртуальной машины и типом ее устройства ввода-вывода), дороговизне, а также в недоступности файлов извне по прямой ссылке. Скорость доступа к файлам тоже определяется конфигурацией топологии соединения дисков в массив RAID и в ряде ситуаций существенно ниже, чем в случае BLOB.

Чтобы преодолеть эти ограничения и одновременно обеспечить работу с файлами стандартными средствами операционных систем и программных библиотек ввода-вывода, следует использовать облачное файловое хранилище Azure File Storage (рис. 4.20).

Способы доступа к облачному файловому хранилищу Azure File Storage

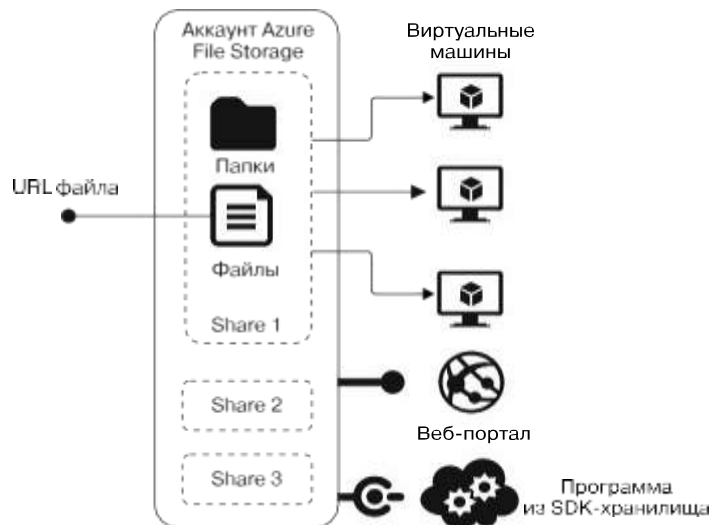


Рис. 4.20. Способы доступа к облачному файловому хранилищу Azure File Storage

Сервис Azure File Storage составляет часть Azure Storage Account.

Каждый Storage Account может включать в себя одну или несколько шар (share).

Чтобы создать новую шару, необходимо с общей панели (см. рис. 4.7) добавить вкладку, нажав + **File share**.

Рис. 4.21. Форма добавления новой файловой шары

В появившейся форме (рис. 4.21) указывается имя шары (Name) — в нашем примере это pokerfileshare — и ее размер (Quota), в данном случае 10 Гбайт. Размер каждой шары (квота) может изменяться (с помощью веб-портала, SDK-управления облачными ресурсами или Azure CLI) и способна достигать 5 Тбайт.

Способы доступа к облачному файловому хранилищу Azure File Storage

- Ключевым отличием шары от контейнера является то, что для нее устанавливается квота — верхний предел размера.
- Эта шаря представляет собой корневой каталог, который доступен по протоколу SMB 3.0 и может быть примонтирован к виртуальным машинам в том же аккаунте Azure и в том же регионе, что и Azure File Storage.
- Наиболее важными опциями конфигурирования является возможность подключения (Connect) к виртуальной машине и создание мгновенного снимка (View snapshot) (рис. 4.22).

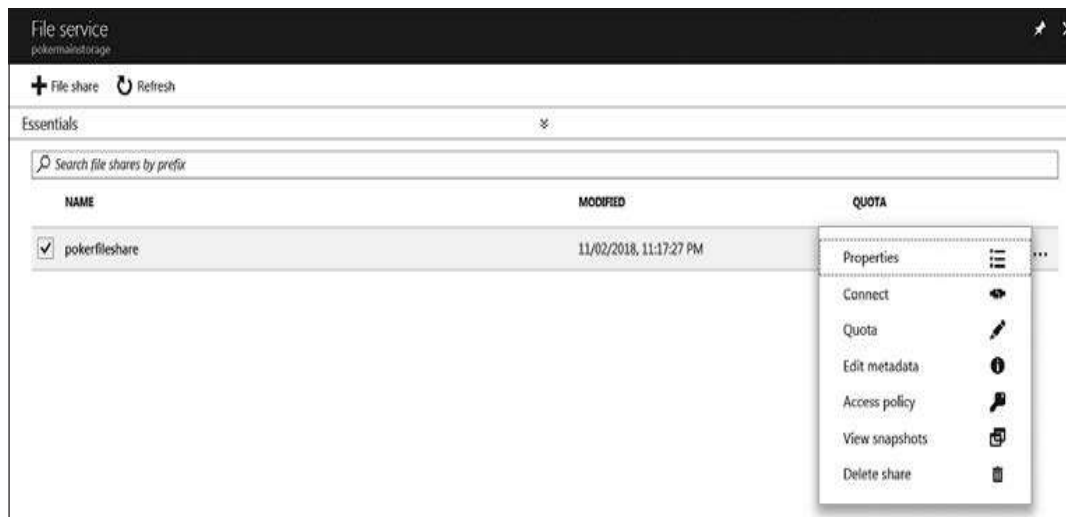


Рис. 4.22. Доступные опции конфигурирования файловой шары

Способы доступа к облачному файловому хранилищу Azure File Storage

Чтобы подключить шару File Storage к виртуальной машине, в оболочке виртуальной машины нужно выполнить команду монтирования сетевого диска. Эта команда может быть получена напрямую на веб-портале после щелчка на ссылке Connect (рис. 4.23).

Достоинства файлового хранилища представлены ниже.

- Возможность прямой миграции файловой системы из локального хранилища в облачное. Будет полностью сохранена иерархия этой системы (вероятные ограничения на символьную длину пути и глубину вложенности каталогов см. в документации).
- Простота взаимодействия из всех программных продуктов, реализующих стандартные интерфейсы ввода-вывода. Не требуется никаких модификаций кода программ, они могут быть устаревшими и все равно будут работать с Azure File Storage, поскольку это хранилище монтируется к основной файловой системе виртуальной машины на уровне операционной системы.
- Возможен просмотр списка файлов с помощью стандартных средств Windows или Linux.

Недостатки файлового хранилища:

- ограниченный размер файловой шары (5 Тбайт) и одного файла (1 Тбайт);
- более высокая по сравнению с BLOB Storage цена за гигабайт;
- меньшая по сравнению с BLOB Storage производительность операций чтения-записи.

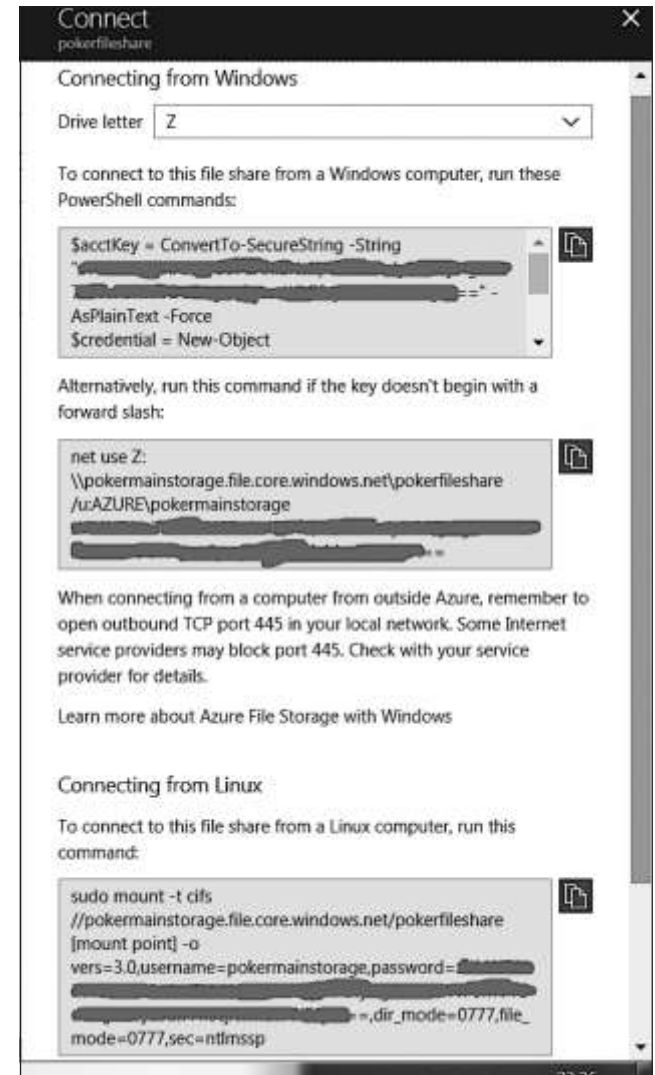
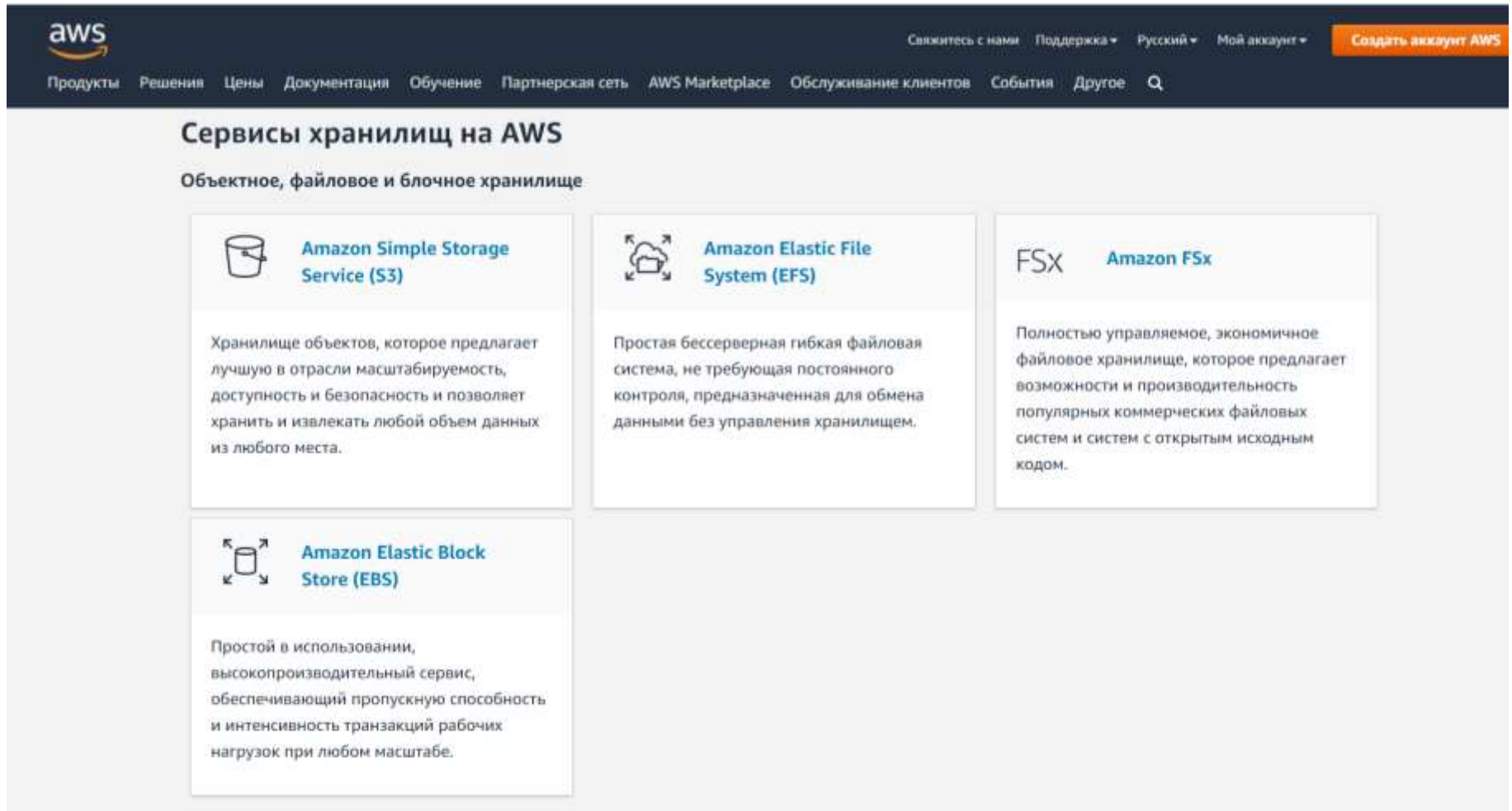


Рис. 4.23. Вкладка Connect Azure File Storage

Классы хранилища Amazon



The screenshot shows the AWS website header with the logo and navigation links. Below the header, the main heading is "Сервисы хранилищ на AWS" (Storage services on AWS), followed by the subtitle "Объектное, файловое и блочное хранилище" (Object, file, and block storage). Three service cards are displayed:

- Amazon Simple Storage Service (S3)**: Described as an object storage service offering the best scalability, availability, and security in the industry, allowing storage and retrieval of any amount of data from anywhere.
- Amazon Elastic File System (EFS)**: Described as a simple, serverless, flexible file system that does not require constant control, designed for data exchange without managing the storage.
- Amazon FSx**: Described as a fully managed, economical file storage service that offers the performance and productivity of popular commercial file systems and open-source systems.
- Amazon Elastic Block Store (EBS)**: Described as a simple to use, high-performance service that ensures high throughput and intensive transaction workloads at any scale.

Рассмотрим облачные хранилища общего назначения, предоставляемые облачным провайдером **Amazon Web Service**. В отличие от Azure у AWS хранилища не объединяются логически в Storage Account, а представляют собой набор отдельных сервисов. Хранение двоичных объектов обеспечивается сервисом AWS S3 — Amazon Simple Storage Service.

Amazon Simple Storage Service

Хранение двоичных объектов обеспечивается сервисом **AWS S3 — Amazon Simple Storage Service**. Экземпляр сервиса AWS S3 может быть логически разбит на бакеты (buckets, дословный перевод — «ведра»), которые будут определять видимость объектов в них, шифрование, жизненный цикл и пр.

Каждый объект (будь то бакет или хранящийся в нем файл) содержит ассоциированный с ним класс хранения (storage class).

Ниже перечислены существующие классы хранения.

- Стандартный (STANDARD) — класс, задаваемый по умолчанию. Предназначен для случая, когда частота доступа к файлам высокая, в связи с чем требуется высокая производительность операций чтения и записи.
- Стандартный с нечастым доступом (STANDARD_IA, Standard Infrequent Access) — предназначен для долговременного хранения файлов, доступ к которым будет производиться нечасто (например, бэкапы), но тем не менее высокая производительность операций чтения-записи все еще важна. Оба класса — STANDARD и STANDARD_IA — обеспечивают доступ к файлу в режиме, близком к реальному времени, то есть с минимальной задержкой между запросом и получением данных.
- Замороженный, или, точнее, ледник (GLACIER), — предназначен для файлов большого размера, доступ к которым будет запрашиваться очень редко (архивы, бэкапы и др.). Принципиальное его отличие от двух предыдущих классов хранения — невозможность прямого доступа к файлам, которые перед выгрузкой должны быть сначала восстановлены в другие классы.
- С уменьшенной избыточностью (REDUCED_REDUNDANCY) — предназначен для хранения некритических файлов с той же скоростью доступа, что и у STANDARD, но за счет уменьшенной избыточности допускается потеря 0,01 % объектов.

Amazon Simple Storage Service

Принципиальное различие между всеми классами — их стоимость. Очевидно, STANDARD обладает наивысшей стоимостью, а GLACIER и REDUCED_REDUNDANCY — наименьшей. В ряде сценариев перемещение файлов из одного класса в другой может быть широко используемой операцией.

Например, это происходит на фотохостинге, размещающем картинки, доступные для скачивания по ссылке. Широко известен интернет-феномен мемов, когда контент внезапно становится очень популярным и в течение нескольких месяцев количество его скачиваний может резко возрасти, а затем значительно упасть. Если у этого фотохостинга большой объем хранимого контента, то становится актуальной проблема оптимизации стоимости хранения. Решить ее призван специальный встроенный механизм AWS S3: управление жизненным циклом объектов (object lifecycle management).

Данный сервис автоматизирует перемещение объектов между классами хранения по цепочке STANDARD - STANDARD_IA - GLACIER, что избавляет пользователя от необходимости строить специальные сервисы, ответственные за это перемещение.

Следующий сценарий применения этого сервиса — жизненный цикл бэкапа. Это необходимо для оптимизации системы хранения бэкапа с точки зрения стоимости/доступности.

Рассмотрим, как создавать бакеты и манипулировать объектами в них. Начальная страница сервиса AWS S3 выглядит следующим образом (рис. 4.24).

Начальная страница сервиса AWS S3

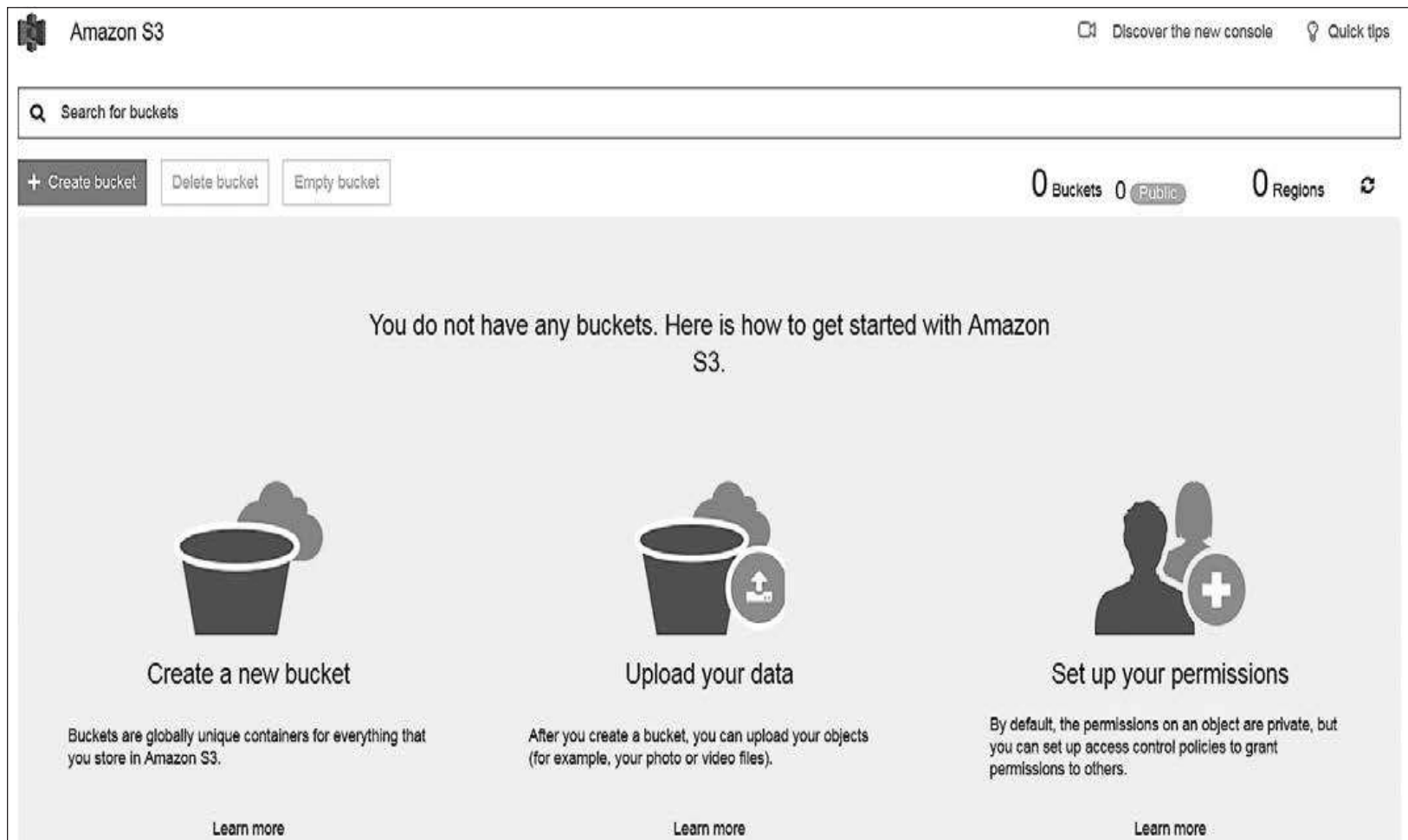


Рис. 4.24. Стартовая страница сервиса AWS S3

Начальная страница сервиса AWS S3

- Чтобы создать бакет, надо щелкнуть на ссылке + Create bucket в левом верхнем углу.
- В результате откроется следующая форма (правая половинка доступна после успешного заполнения всех полей первой формы — вкладки Name and region (Имя и регион)) (рис. 4.25).

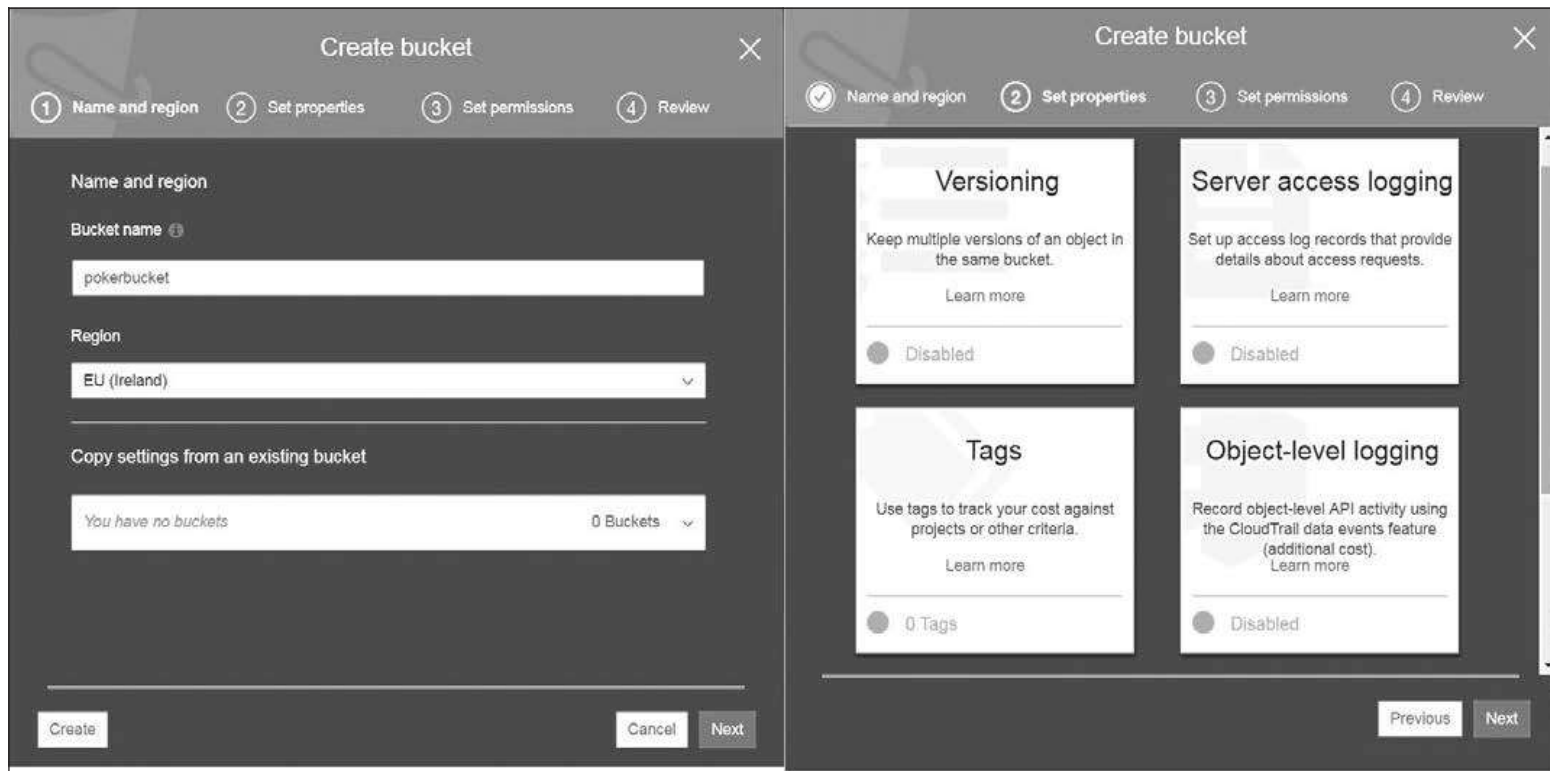


Рис. 4.25. Последовательные шаги создания бакета: вкладки Name and region (Имя и регион) и Set properties (Настройка свойств)

Дальнейшие шаги создания бакета

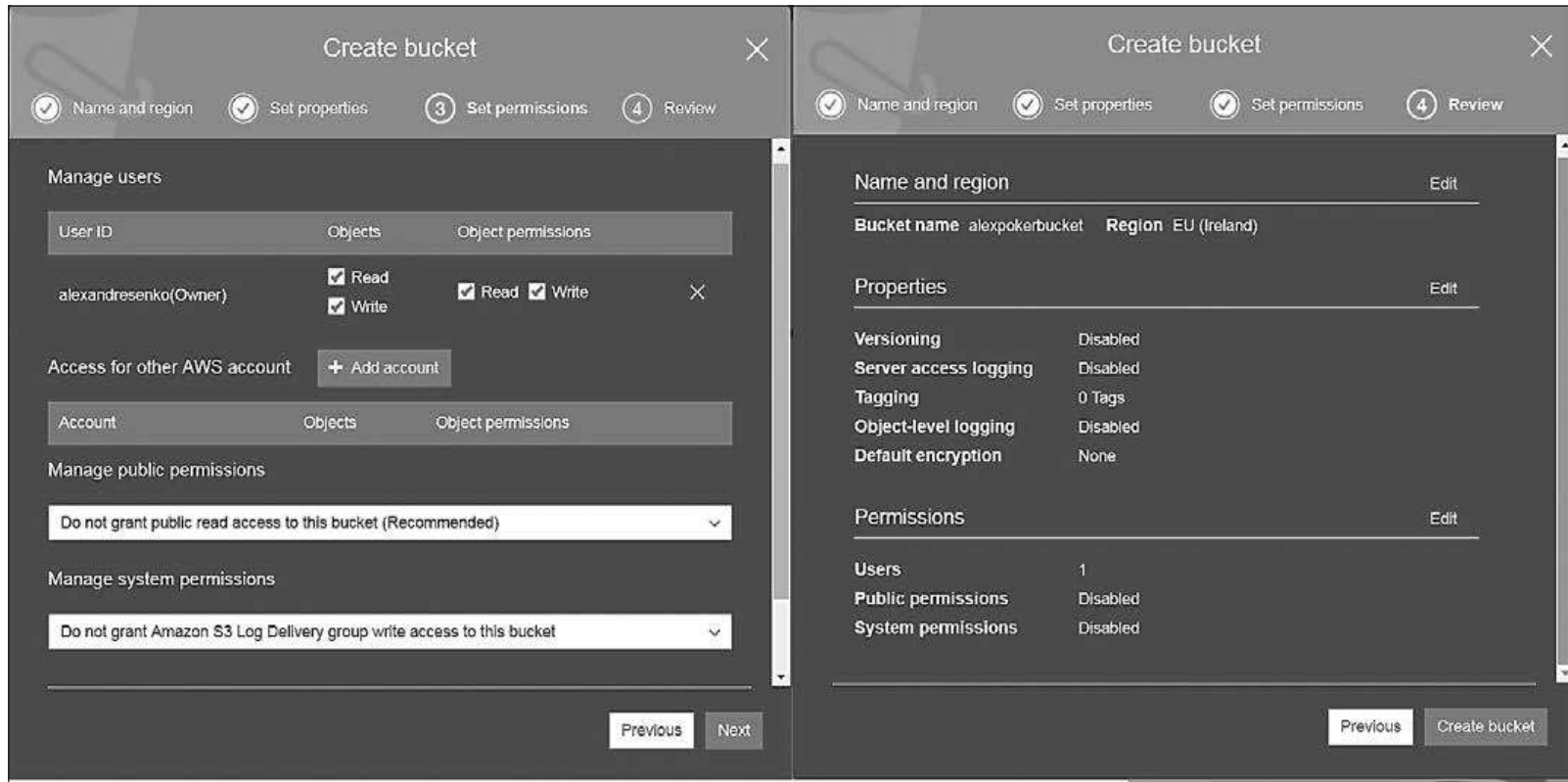


Рис. 4.26. Дальнейшие шаги создания бакета: Set permissions (Настройка прав доступа) и Review (Финальный обзор)

Дальнейшие шаги создания бакета

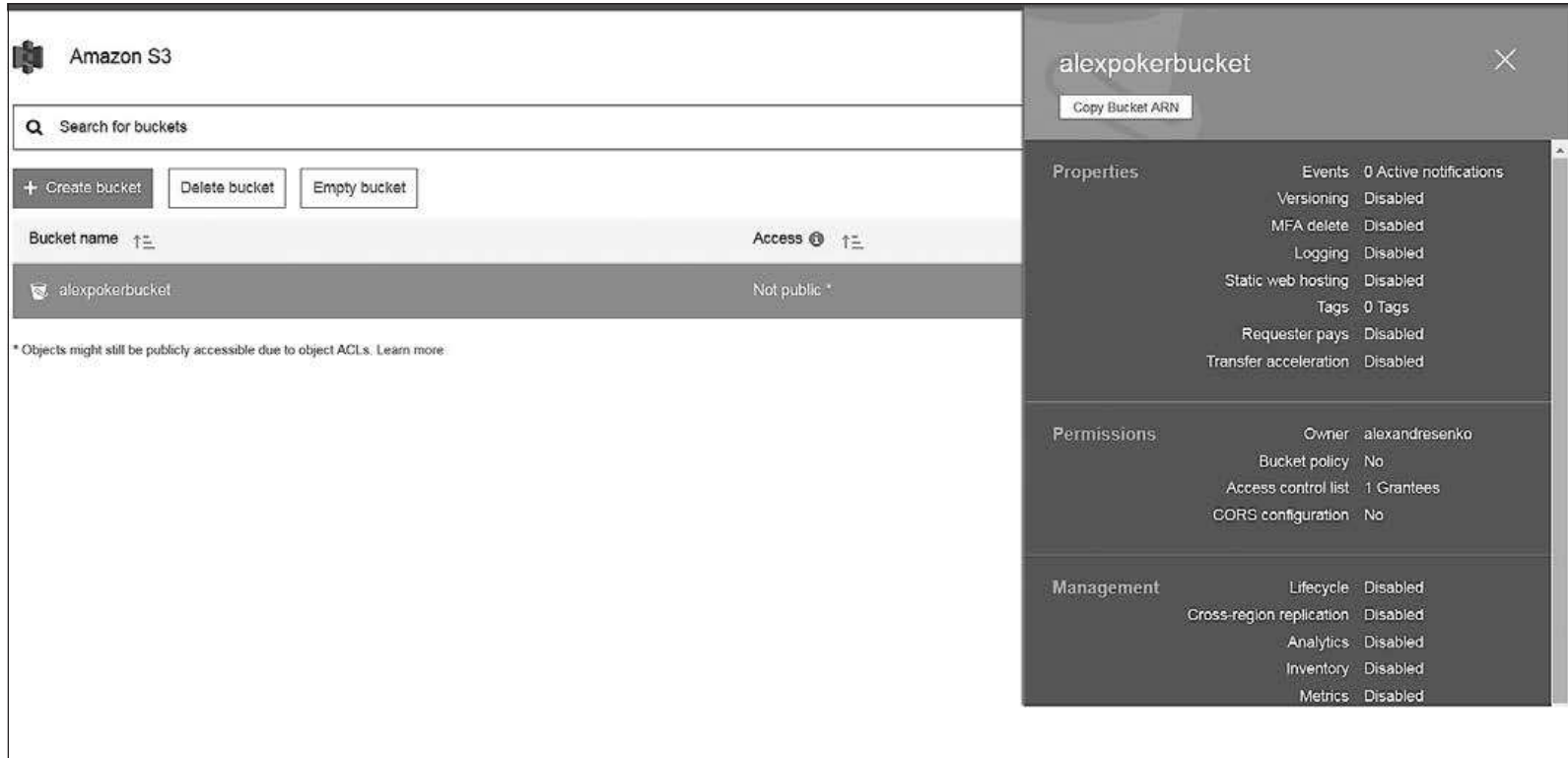


Рис. 4.27. Созданный бакет с раскрытой формой свойств

Дальнейшие шаги создания бакета

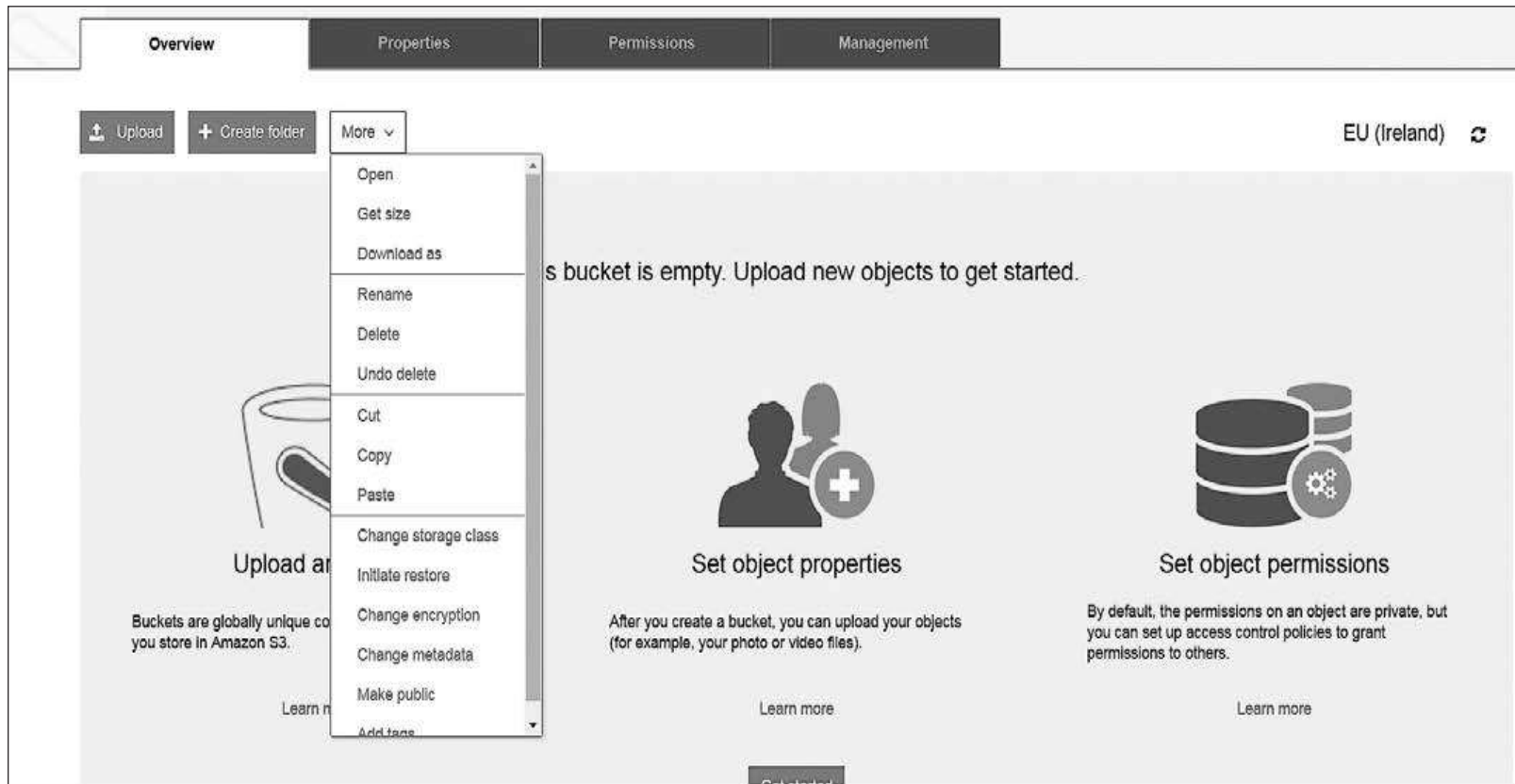


Рис. 4.28. Вкладка бакета, позволяющая управлять объектами

Хранилище файлов, предоставляемое AWS — Amazon Elastic File Service (EFS).

Следующее хранилище файлов, предоставляемое AWS, — Amazon Elastic File Service (EFS). Этот сервис является облачной реализацией сетевого хранилища с поддержкой протокола NFSv4. Он изначально предназначен для доступа с виртуальных машин AWS EC2. Файлы и метаданные хранятся в виде нескольких копий в разных зонах доступности (availability zone) в пределах одного региона.

Масштабирование хранилища происходит вплоть до петабайтных размеров.

Поскольку этот сервис тесно связан с AWS EC2, то для его конфигурирования требуется настраивать сетевые IaaS-сервисы — VPC, подсети, availability zone, сетевые группы безопасности (рис. 4.29).

Хранилище файлов, предоставляемое AWS — Amazon Elastic File Service (EFS).

Create file system

Step 1: Configure file system access

Step 2: Configure optional settings

Step 3: Review and create

Configure file system access

An Amazon EFS file system is accessed by EC2 instances running inside one of your VPCs. Instances connect to a file system by using a network interface called a mount target. Each mount target has an IP address, which we assign automatically or you can specify.

VPC

Create mount targets

Instances connect to a file system by using mount targets you create. We recommend creating a mount target in each of your VPC's Availability Zones so that EC2 instances across your VPC can access the file system.

	Availability Zone	Subnet	IP address	Security groups
<input checked="" type="checkbox"/>	us-east-2a	<input type="text" value="subnet-4e0cbb26 (default)"/>	Automatic	<input type="text" value="sg-464fe72d - default"/>
<input checked="" type="checkbox"/>	us-east-2b	<input type="text" value="subnet-3d48a647 (default)"/>	Automatic	<input type="text" value="sg-464fe72d - default"/>
<input checked="" type="checkbox"/>	us-east-2c	<input type="text" value="subnet-5b0d2c16 (default)"/>	Automatic	<input type="text" value="sg-464fe72d - default"/>

Cancel **Next Step**

Рис. 4.29. Начальная страница конфигурирования сервиса AWS EFS

Хранилище файлов, предоставляемое AWS — Amazon Elastic File Service (EFS).

На следующей вкладке указываем теги, уровень производительности (General purpose или Max I/O) и шифрование (Enable encryption) (рис. 4.30).

Далее выполняем обзор параметров создаваемого сервиса и в итоге имеем готовый сервис, который можно подключать к виртуальной машине AWS EC2 (рис. 4.31).

Чтобы получить инструкции по подключению файловой системы к виртуальной машине, следует нажать на ссылку [Amazon EC2 mount instructions](#) или [AWS Direct Connect mount instructions](#).

Хранилище файлов, предоставляемое AWS — Amazon Elastic File Service (EFS).

Create file system

Step 1: Configure file system access

Step 2: Configure optional settings

Step 3: Review and create

Configure optional settings

Add tags

You can add tags to describe your file system. A tag consists of a case-sensitive key-value pair. (For example, you can define a tag with key-value pair with key = Corporate Department and value = Sales and Marketing.) At a minimum, we recommend a tag with key = Name.

Key	Value	Remove
<input type="text" value="Name"/>	<input type="text" value="GeneralPokerStore"/>	<input type="button" value="✕"/>
<input type="text" value="Add New Key"/>	<input type="text"/>	

Choose performance mode

We recommend **General Purpose** performance mode for most file systems. **Max I/O** performance mode is optimized for applications where tens, hundreds, or thousands of EC2 instances are accessing the file system — it scales to higher levels of aggregate throughput and operations per second with a tradeoff of slightly higher latencies for file operations.

General Purpose (default)

Max I/O

Enable encryption

If you enable encryption for your file system, all data on your file system will be encrypted at rest. You can select a KMS key from your account to protect your file system, or you can provide the ARN of a key from a different account. Encryption can only be enabled during file system creation. [Learn more](#)

Enable encryption

Рис. 4.30. Дальнейшее конфигурирование AWS EFS

Хранилище файлов, предоставляемое AWS — Amazon Elastic File Service (EFS).

File systems

Create file system Actions

Name	File system ID	Metered size	Number of mount targets	Creation date
GeneralPokerStore	fs-43aa553a	6.0 KiB	3	2018-02-13T09:45:57Z

Other details

Owner ID: 759462564594
Life cycle state: Available
Performance mode: General Purpose
Encrypted: No

Tags

Name: GeneralPokerStore

File system access

DNS name: fs-43aa553a.efs.us-east-2.amazonaws.com

Amazon EC2 mount instructions
AWS Direct Connect mount instructions

Mount targets

VPC	Availability Zone	Subnet	IP address	Mount target ID	Network interface ID	Security groups	Life cycle state
vpc-2d2f8545 (default)	us-east-2a	subnet-4e0cbb26 (default)	172.31.4.176	fsmt-c85aa4b1	eni-e92065b7	sg-464fe72d - default	Available
	us-east-2c	subnet-5b0d2c16 (default)	172.31.42.64	fsmt-ca5aa4b3	eni-6ae8ed41	sg-464fe72d - default	Available
	us-east-2b	subnet-3d48a647 (default)	172.31.30.226	fsmt-cb5aa4b2	eni-5627f502	sg-464fe72d - default	Available

Рис. 4.31. Созданная файловая система AWS EFS, готовая к монтированию

Хранилище файлов, предоставляемое AWS — Amazon Elastic File Service (EFS).

- ❖ Интересная возможность AWS EFS — функция EFS FileSync, позволяющая синхронизировать файлы в EFS и файлы в локальной файловой системе как EC2, так и физического компьютера вне AWS. В последнем случае нужно установить специальный агент на тот компьютер, к которому будут подключаться диски, требующие синхронизации.
- ❖ В отличие от Azure, где для хранения дисков виртуальных машин используется обычный аккаунт BLOB Storage, в котором хранятся BLOB-объекты с блочной организацией, в Amazon применяется специальный сервис — AWS EBS (Elastic Block Storage).
- ❖ Этот сервис отвечает за создание, размещение, шифрование, подключение к виртуальным машинам, создание бэкапов в виде мгновенных снимков (snapshots). Чаще всего пользователь не будет взаимодействовать с ним напрямую, этот сервис доступен в «связке» с другими: AWS EC2, AWS ECS и пр.
- ❖ То есть при создании виртуальной машины тип и количество виртуальных дисков, их точки монтирования или метки указывается прямо в момент создания.
- ❖ Однако нужно помнить, что бывают сценарии, когда эти виртуальные диски не удаляются при удалении виртуальной машины.

Хранилище файлов, предоставляемое AWS — AWS EBS (Elastic Block Storage)

Диски EBS можно разделить на две большие группы: HDD (Hard Disk Drive) (классический магнитный диск) и SSD (Solid State Drive) (твердотельный диск). Обе группы имеют следующие разновидности.

- *Холодный HDD (Cold HDD)* — самый дешевый тип диска EBS. Сценарии его использования включают хранение больших объемов данных, которые редко запрашиваются извне в случаях, когда стоимость хранения имеет значение. Применение этого типа сопряжено с рядом ограничений: загрузочный (boot) диск его не поддерживает. Кроме того, существуют разного рода ограничения для использования холодного HDD вместе с ECS-оптимизированными виртуальными машинами.
- *Горячий HDD (throughput oriented HDD, дословно «HDD с оптимизированной производительностью»)* — магнитный диск малой стоимости, оптимизированный для сценариев хранения больших объемов данных, частота запросов которых высока и для которых требуется высокая производительность при минимальной цене. В документации AWS EBS указаны следующие сценарии применения этого типа: потоковые чтение-запись с высоким и постоянным уровнем производительности при минимальной цене; хранение и оперирование большими данными, в том числе складирование данных, построение DataWarehouse и обработка логов. Так же как и Cold HDD, горячий HDD не может быть загрузочным (boot).
- *SSD общего назначения (general purpose SSD)* — тип твердотельного диска, подходящий для широкого класса сценариев, включая диск для операционной системы. Его производительность, выраженная в величинах IOPS (input output per second — операции ввода/вывода в секунду), в два раза превышает таковую у горячего HDD. Этот тип диска рекомендуется для большинства случаев использования.
- *SSD с выделенной полосой пропускания (provisioned IOPS SSD)* — тип твердотельного диска, обладающий наибольшей производительностью чтения-записи (и как результат, наибольшей стоимостью), предназначенный для построения критически важных компонентов с жесткими требованиями к задержкам и производительности. В качестве типовых сценариев применения этого типа можно указать высоконагруженные диски баз данных (включая MongoDB, Cassandra, Microsoft SQL Server, MySQL, PostgreSQL, Oracle и др.).

Практическая работа №2

Развертывание облачных сервисов копирования и трансформации данных

Сервисы трансформации и копирования играют важную роль в построении информационных систем анализа больших данных. Требования к сервисам хранения и анализа противоречивы и не могут быть удовлетворены в рамках одного хранилища, особенно когда данных много. Чтобы удовлетворить эти требования, необходимо использовать хранилища разных типов и сервис, который связывает их информационно.

Сервис [AWS Glue](#) представляет собой удобное средство каталогизации, трансформации и копирования данных, что позволяет одновременно создавать каталоги данных и использовать их для ETL.

Задание:

- 1. На веб-портале Azure создайте и сконфигурируйте сервис Azure Data Factory. Предложите решение задачи копирования данных из таблицы Azure Table Storage в реляционную БД Azure SQL.**
2. Разверните сервис AWS Data Pipeline для автоматизации копирования и трансформации данных.
3. Разверните бессерверный сервис AWS Glue для категоризации данных в каталог данных и выполнения задач ETL. Предложите решение задачи копирования в AWS Redshift данных в виде файла JSON, хранящегося в AWS S3

Развертывание облачных сервисов копирования и трансформации данных

Итак, существуют следующие стратегии поступления больших данных в облачные информационные системы.

- *Прямое копирование файлов.* В зависимости от конечной цели копирования и объемов данных выбирается свой тип облачного хранилища. Как правило, облачное хранилище общего назначения отвечает большинству критериев. Когда необходимо обеспечить доступность файлов с виртуальных машин, используется хранение файлов в файловой системе. Но, как правило, конечная цель прямого копирования файлов — анализ информации в файлах. Для этого требуется переместить их в специализированное хранилище, допускающее такой анализ. Для AWS подобным хранилищем может быть напрямую хранилище общего назначения AWS S3, для Azure — специализированное хранилище Azure Data Lake Store. По сути, описанный подход реализует концепцию Data Lake.
- *Прямое копирование информации.* Информацию можно хранить в базах данных и копировать в однотипные сервисы БД, например из баз данных SQL в сервисы РБД или реляционные хранилища данных. То же самое относится к нереляционным базам данных.
- *Трансформация данных.* Информация из внешних источников преобразуется в другой формат. Существуют различные варианты трансформации, один из которых — последовательное преобразование данных из одного формата в другой. В простейшем случае выполняется одна подобная трансформация, например из CSV-файла в таблицу SQL. В другой ситуации возможна цепочка преобразований. Так, в части I был описан пример системы мониторинга потребления электроэнергии, в которой сырые данные телеметрии сначала хранятся в табличной базе данных, а после перемещаются в реляционное хранилище.

Прямая загрузка данных

- У каждого облачного сервиса хранения данных есть конечная точка REST API и программная SDK, которая служит для упрощения доступа к этой точке. Кроме того, каждый сервис поддерживает специфический протокол обмена данными. SDK и конечная точка позволяют загружать данные напрямую в этот сервис. Такой способ позволяет перемещать большие объемы данных, необходимых для последующего пакетного и интерактивного анализа, а также тренировки моделей машинного обучения. Каждый сервис хранения данных имеет специфичный способ добавления данных, и мы рассмотрим эти способы применительно ко всем сервисам, описанным в главе 2.
- Поскольку использование SDK не слишком отличается в AWS и Azure, ограничимся рассмотрением Azure SDK на примере консольных приложений .Net Core 2.0.
- Оба облачных аккаунта, AWS и Azure, имеют в своем составе очень мощные сервисы копирования и трансформации данных: AWS Glue и Azure Data Factory.
- С их помощью очень удобно перемещать информацию между отдельными сервисами внутри облачного провайдера и в ряде случаев между внешними ресурсами и облаком.

Доставка данных в облачное хранилище общего назначения

В самом общем виде облачные хранилища делятся на хранилища двоичных объектов (BLOB) и хранилища файлов на основе протоколов сетевых файловых систем (File Storage). В обоих случаях данные можно добавлять в хранилище следующими способами:

- с помощью веб-портала облачного аккаунта, что можно использовать только для добавления малого количества файлов небольшого объема;
- благодаря конечным точкам REST API, представленным в виде программных SDK, которые могут быть применены в программных продуктах.

Для файлового хранилища доступен еще один способ — монтировать его в качестве сетевого к виртуальной машине и работать с ним уже из нее.

Итак, рассмотрим использование [Azure Storage SDK](#) на примере приложения .Net Core.

- Сначала следует установить пакеты NuGet (рис. 9.1).
- Код для загрузки одиночного файла с именем "format.json" в BLOB-контейнер под именем "pokerattachments" выглядит так (рис. 9.2).

Azure Storage SDK

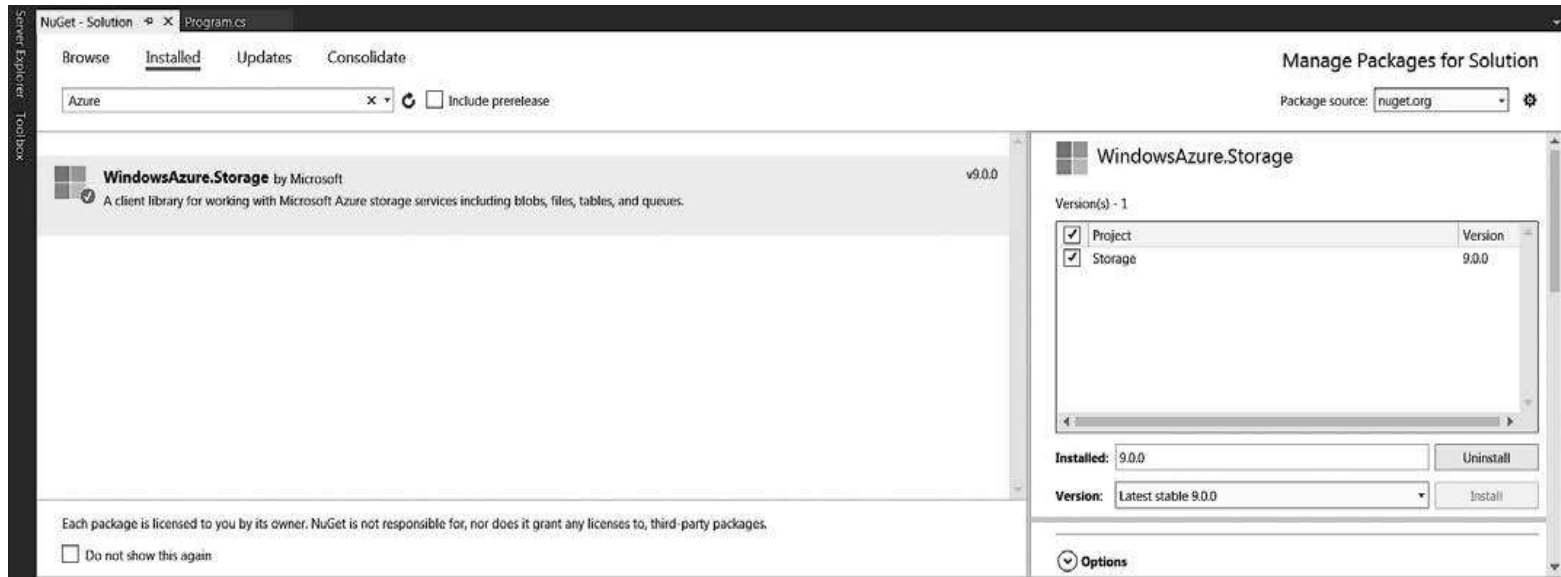


Рис. 9.1. NuGet-пакеты, необходимые для работы с Azure Storage SDK

```
private static async Task MainAsync()
{
    string storageConnectionString = "";

    CloudStorageAccount storageAccount = CloudStorageAccount.Parse(storageConnectionString);

    CloudBlobClient cloudBlobClient = storageAccount.CreateCloudBlobClient();
    var cloudBlobContainer = cloudBlobClient.GetContainerReference("pokerattachments");
    await cloudBlobContainer.CreateIfNotExistsAsync();

    BlobContainerPermissions permissions = new BlobContainerPermissions
    {
        PublicAccess = BlobContainerPublicAccessType.Blob
    };
    await cloudBlobContainer.SetPermissionsAsync(permissions);

    string localPath = @"C:\Users\alexander\Projects";
    string localFileName = "format.json";
    string sourceFile = Path.Combine(localPath, localFileName);

    CloudBlockBlob cloudBlockBlob = cloudBlobContainer.GetBlockBlobReference(localFileName);
    await cloudBlockBlob.UploadFromFileAsync(sourceFile);
}
```

Рис. 9.2. Пример использования Azure Storage SDK для загрузки файлов

Доставка данных в реляционные БД и хранилища

Синхронизировать данные между реляционными и нереляционными базами данных можно с помощью любого стороннего приложения, допускающего синхронизацию/миграцию данных между базами.

Довольно часто в среде Azure в качестве такового используется набор утилит RedGate. Кроме того, можно применять сервис **Azure Data Factory** и относительно новый сервис — **Azure Database Migration Service** (сейчас он находится в стадии Public Preview).

Ниже представлены несколько замечаний, которые позволят увеличить производительность операции добавления данных:

- имеет смысл на время копирования максимально поднять ценовой уровень экземпляра сервиса Azure SQL или взять максимальный размер экземпляра AWS RDS;
- в базе данных — приемнике на время копирования следует отключить опцию аудита и триггеры на добавление/обновление;
- после добавления данных в таблицу стоит обновить индексы и пересчитать статистику.

Прямая загрузка потоковых данных

К облачным сервисам очередей относятся Azure Storage Queue, Azure Service Bus Queue и AWS SQS. К сервисам, размещаемым на виртуальных машинах, можно отнести RabbitMQ, ZeroMQ, MSMQ, IBM MQ и др.

Различные сервисы очередей гарантируют различные виды доставки сообщений:

- как минимум однократную доставку сообщения;
- строго однократную доставку;
- доставку сообщений с сохранением порядка;
- доставку сообщений без сохранения порядка.

Azure Event Hub

Рассмотрим сервис концентратора сообщений Azure Event Hub. Он представляет собой сервис, построенный по модели PaaS. В качестве источников сообщений для Azure Event Hub могут выступать различные группы клиентов (рис. 10.8). Прежде всего, это очень большая группа облачных сервисов, чьи выходы или триггеры можно сконфигурировать для посылки сообщений напрямую в Event Hub. Это могут быть Stream Analytics Job, Event Grid и значительная группа сервисов, перенаправляющих события — логи в Event Hub (прежде всего, построенные с помощью AppService: Api App, Web App, Mobile App и Function App).

Доставленные в концентратор сообщения могут быть напрямую захвачены (capture) и помещены в хранилище Blob Storage или Data Lake Store.

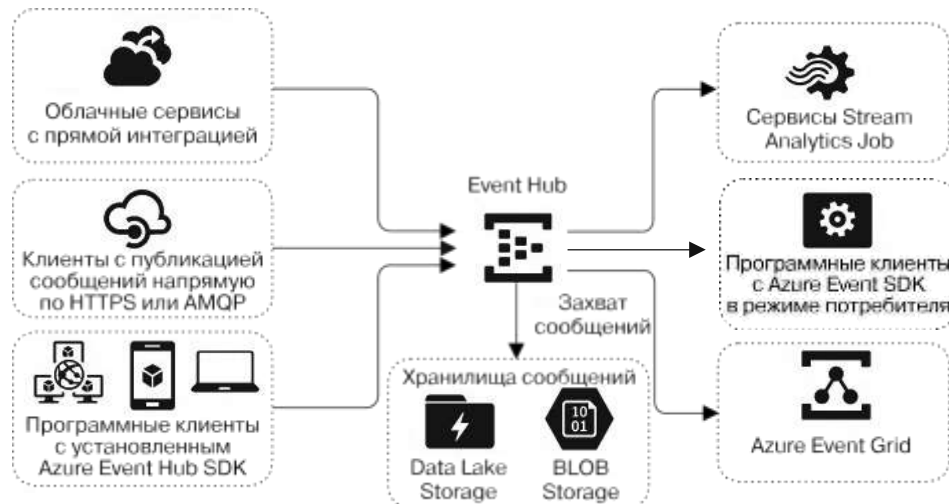


Рис. 10.8. Общая структура взаимодействия Event Hub со сторонними сервисами

AWS Kinesis Data Streams

AWS Kinesis Data Streams представляет собой сервис концентрации и приема потоковых данных от AWS.

Он является частью общей платформы Kinesis, которая работает с потоковыми данными и включает в себя:

- *Kinesis Video Streams* — сервис для приема потоковых медиаданных, в качестве которых могут выступать видео- и звуковые потоки, потоки информации от систем ультразвукового обзора и радиолокации. Он позволяет подключать клиентские приложения, занимающиеся потоковой обработкой аудио- и видеоматериалов. Является новым сервисом группы медиасервисов AWS, его рассмотрение выходит за рамки нашей книги;
- *Kinesis Firehose* — сервис для доставки потоковых данных в хранилища AWS S3, AWS RedShift, Amazon Elasticsearch и дополнительно в Splunk. (Splunk представляет собой не сервис AWS, а отдельную платформу для анализа логов и построения панелей мониторинга, которая разворачивается в экземплярах EC2.) Этот сервис позволяет в ряде случаев отказаться от использования клиентских приложений в целях доставки данных к указанным хранилищам;
- *Kinesis Data Analytics* — сервис потокового анализа данных, принятых AWS Kinesis Data Streams, позволяющий с помощью SQL-подобного синтаксиса строить задания потоковой обработки, а по сути онлайн-фильтрации, агрегирования, объединения из разных потоков и выборки подмножества полей из всех доступных в сообщении. Этот сервис будет подробно рассмотрен в части IV;
- *Kinesis Data Streams* — это сервис концентратора сообщений от AWS. Начнем его подробное рассмотрение.

Итак, AWS Kinesis Data Streams — один из сервисов платформы Kinesis, которые могут конфигурироваться друг с другом (рис. 10.32).

AWS Kinesis Data Streams

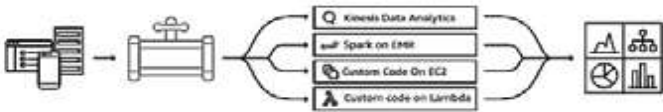
Get started with Amazon Kinesis

Choose the type of Amazon Kinesis resource you want to start with.

Amazon Kinesis resources

Ingest and process streaming data with Kinesis streams


Process data with your own applications, or using AWS managed services like Amazon Kinesis Data Firehose, Amazon Kinesis Data Analytics, or AWS Lambda.



Create data stream

Deliver streaming data with Kinesis Firehose delivery streams


Continuously collect, transform, and load streaming data into destinations such as Amazon S3 and Amazon Redshift.



Create delivery stream

Analyze streaming data with Kinesis analytics applications

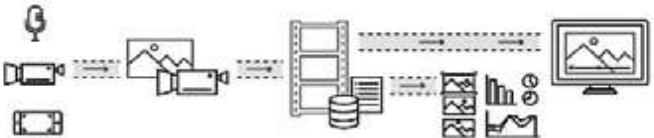
Run continuous SQL queries on streaming data from Kinesis data streams and Kinesis Firehose delivery streams.



Create analytics application

Ingest and process media streams with Kinesis video streams

Build applications to process or analyze streaming media.



Create video stream

Рис. 10.32. Общий вид панели конфигурирования сервисов платформы Kinesis

AWS Kinesis Data Streams

- *Потоки Kinesis Data Streams* — это упорядоченные последовательности записей данных. *Запись данных* — единица хранения информации в AWS Kinesis Data Streams. Каждая запись имеет порядковый номер, который генерируется и присваивается самим сервисом AWS Kinesis Data Streams, ключ раздела и собственно полезной нагрузки данных, представляющей собой неизменяемую последовательность байтов. Kinesis Data Streams никак не изменяет или никак не влияет на эти данные, которых может быть до 1 Мбайт. Но при этом имеет возможность шифрования записей с помощью сервиса AWS KMS.
- *Порядковый номер* — уникальный, в пределах фрагмента, номер записи, который присваивается после добавления новой записи в Kinesis Data Streams. По мере добавления записей текущее максимальное значение этого номера постоянно увеличивается.
- *Ключ раздела* — ключ, используемый для разделения записей на фрагменты. Сам сервис Kinesis Data Streams распределяет записи среди многих разделов, а данный ключ, ассоциированный с каждой записью, определяет, к какому фрагменту относится конкретная запись. Для этого используется хеш-функция MD5, преобразующая ключ раздела в целое значение, являющееся номером фрагмента.
- Записи разделяются на *фрагменты* (shards). Они являются группами записей, на которые сервис выделяет фиксированный объем (unit of capacity).

Облачные сервисы копирования и трансформации данных

Сервисы копирования и трансформации данных, которые переносят их из одного хранилища в другое, подвергая преобразованию, — очень важная составляющая информационной системы.

Выше уже упоминались два сервиса из этой области — Apache Sqoop и DistCp. Но они требуют кластерных решений типа Azure HDInsight и AWS EMS. Эти сервисы предоставляют «ядро» или «движок» копирования и трансформации, однако конвейер данных может состоять из многих этапов копирования и преобразования информации.

Управление этим конвейером берут на себя облачные сервисы копирования и трансформации данных:

- **Azure Data Factory,**
- **AWS Data Pipeline,**
- **AWS Glue.**

Azure Data Factory

Azure Data Factory представляет собой сервис трансформации и копирования от Azure. В настоящее время реализована версия 2 данного сервиса, в то время как для заказчиков, применявших сервис ранее, доступна версия 1, отличающаяся от текущей.

Но мы ограничимся рассмотрением версии 1, поскольку она является наиболее актуальной и единственно доступной для новых пользователей.

Azure Data Factory — сервис прежде всего для управления процессами копирования и трансформации данных. Как правило, этот сервис состоит из следующих компонентов (рис. 11.1).

- *Наборы данных (dataset)* — конкретные поименованные представления данных. В их качестве могут выступать файлы, таблицы и пр. Наборы данных могут быть как входными, называемыми *источниками (source)*, так и выходными, называемыми *воронками (sink)*.
- *Подключенные серверы (linked servers)* — наборы метаданных источников (адреса, учетные данные, протоколы и др.), на которых располагаются источники и приемники данных.
- *Наборы выполняемых заданий (activity)* — одно или несколько выполняемых заданий по копированию или трансформации данных.
- *Конвейер (pipeline)* — набор сервисов по обеспечению периодического выполнения, управления и мониторинга, логически соединенный с подключенными серверами, наборами данных и заданиями для выполнения конкретной задачи.

Azure Data Factory

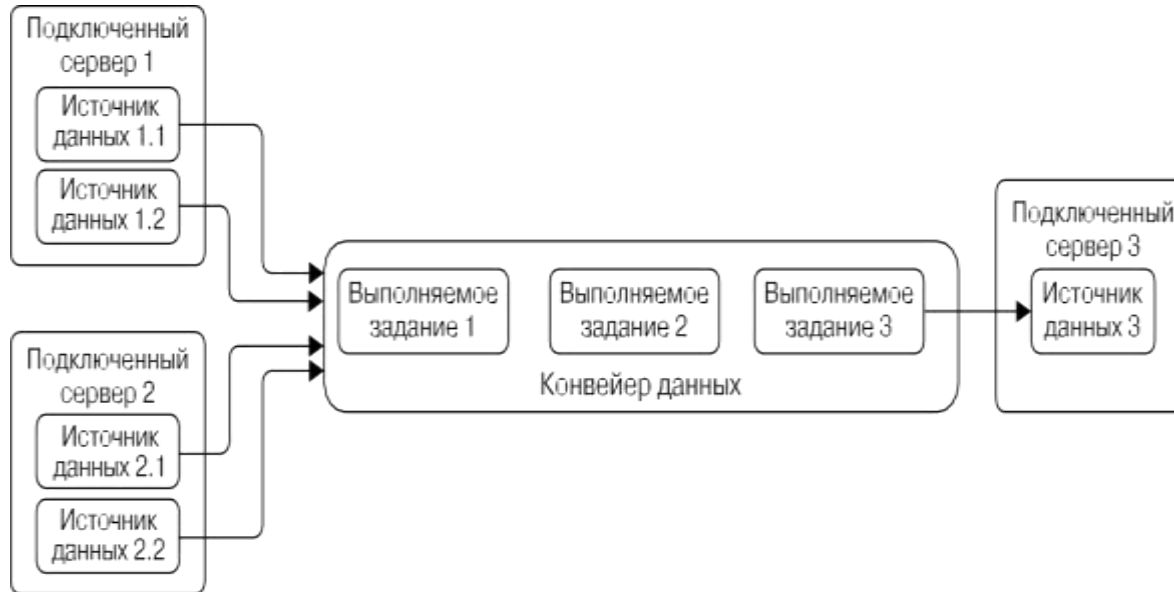


Рис. 11.1. Архитектура сервиса Azure Data Factory

- Рассмотрим, как все компоненты работают и взаимодействуют между собой. Прежде всего, сервис Azure Data Factory состоит из одного или нескольких *конвейеров*.
- Каждый конвейер реализует конкретную задачу ETL/ELT, для чего ему нужны источники и приемники данных, собственно список выполняемых заданий, триггеры (то есть условия начала выполнения) и набор входных параметров, подаваемых на вход.

Azure Data Factory

В настоящее время в качестве источников данных Azure Data Factory выступают хранилища, как облачные (в том числе и в иных облачных средах, например AWS), так и локальные:

- реляционные и нереляционные БД,
- файлы (текстовые, CSV-, HTML-, JSON- и XML-таблицы), хранящиеся в различных файловых системах и доступные по разным протоколам (FTP, SFTP, HDFS, AWS S3),
- реляционные и нереляционные хранилища данных,
- сервисы REST и SOAP (в том числе поддерживающие протокол OData), SAP, Dynamic CRM, Office 365 и др.

В то же время количество приемников данных в значительной степени зависит от того, какой *исполняемый режим (runtime)* выбран для **Azure Data Factory**.

По умолчанию конвейер управляется ресурсами Azure и для пользователя выглядит как **бессерверный**. Такой режим называется **встроенным исполняемым режимом (integrated runtime, IR)**.

В нем пользователю не требуется выполнять никаких действий по администрированию и мониторингу конвейера — все нужные вычислительные ресурсы Azure создает автоматически.

Но в этом случае перенос информации ограничивается облачными источниками и приемниками данных, расположенными в публично доступных сетях. Чтобы преодолеть это ограничение, задействуется вариант режима внешнего хостинга (self-hosted) — размещения исполняемого модуля Azure Data Factory на физическом сервере или виртуальной машине.

Azure Data Factory

Кроме того, существует также очень интересный промежуточный вариант — **интегрированный исполняемый режим SSIS (SSIS IR)**. Он позволяет с помощью Azure Data Factory запускать пакеты SSIS, для чего могут служить как выделяемые машины с установленными Microsoft SQL, так и пользовательские SQL-серверы (обратите внимание: эта возможность сейчас находится в стадии Preview).

Вернемся к остальным компонентам Azure Data Factory. Важнейший и ключевой компонент — **выполняемое задание (activity)**. **Существуют три вида заданий: перенос данных, их трансформация и поток управления (control flow)**.

Рассмотрим их по порядку.

- **Перенос данных** служит для копирования данных из источника в приемник без выполнения каких бы то ни было трансформаций, агрегаций и пр. Это простое копирование из источника в приемник. Оба — источник и приемник — должны быть представлены как наборы данных (data set) в подключенных серверах (linked server). Как раз для подобной активности мы и рассмотрели различные исполняемые режимы (IR, SSIS IR, self-hosted).
- **Трансформация данных** подразумевает применение вычислительных ресурсов кластера, выделяемого по требованию (Azure HDInsight), или ресурсов, выделяемых пользователем. В настоящее время преобразовывать данные позволяют следующие технологии: HDInsight — Hive, Pig, Spark, Hadoop Streaming, MapReduce; Machine Learning; хранимая процедура в базе данных SQL или хранилище данных; U-SQL-сценарий Azure Data Lake Analytics и пользовательская программа, написанная на .NET. При трансформации возможны два исполняемых режима: с выделением ресурсов по требованию (например, кластер Azure HDInsight) или на ресурсах, предоставляемых пользователем (на его виртуальных машинах). Как видим, преобразование в этом случае может применяться в сценариях ETL/ELT, для выполнения пакетных, а также периодических административных заданий (например, запуск хранимой процедуры в Azure DWH для пересчета индексов). И Azure Data Factory при этом играет роль дирижера (orchestrator).
- **Поток управления**, который служит для управления другими заданиями, используя условные, циклические и прочие управляющие задания-операторы.

Задания Azure Data Factory

В настоящее время поддерживаются следующие [типы заданий](#):

- задание условного логического оператора ветвления IF (IF condition activity); работает по тому же принципу, что и логический оператор if в языках программирования: если логическое условие является истинным (true), то выполняется соответствующее задание А, в противном случае выполняется задание Б;
- задание условного цикла с предусловием Until (Until condition activity); представляет собой реализацию оператора повторения выполнения задания до тех пор, пока условие, ассоциированное с этим циклом, истинно или не будет достигнуто тайм-аут;
- задание по выполнению REST-запроса (web activity);
- циклическое задание с фиксированным количеством шагов (for each activity) — оператор повторения задания определенное число раз;
- задание ожидания (wait activity) обеспечивает задержку выполнения основного задания на определенное время;
- задание последовательного перебора (lookup activity) может служить для чтения и последовательного просмотра записей в таблицах и файлах из внешних источников;
- задание по получению метаданных (get metadata activity) используется для получения метаданных от подключенных источников;
- задание активации другого конвейера данных (execute pipeline activity).

Кроме того, Azure Data Factory содержит достаточно богатый доменно- специфичный «язык» из богатого набора операторов, переменных (определяемых пользователем), функций и возможность строить различные выражения из этих элементов, обеспечивая довольно гибкое управление заданиями. Все компоненты Azure Data Factory можно описать с помощью JSON.

Azure Data Factory

Теперь рассмотрим вопрос запуска конвейера данных. Собственно, запуск включает передачу на вход конвейера параметров и активацию соответствующей конечной точки, будь то REST или точка запуска, определяемая в триггере конвейера. При этом каждый исполняемый конвейер получает уникальный идентификатор и становится доступным для средств мониторинга.

Запустить конвейер можно с помощью:

- команды Azure CLI или Azure PowerShell;
- прямого REST-запроса к конечной точке запуска;
- триггера, определяемого в JSON-файле описания конвейера.

Обратите внимание: в настоящий момент Azure Data Factory не поддерживает запуск по событию и *прямую* интеграцию, например, с Azure Function (по крайней мере на момент написания книги такой возможности нет), Azure Logic App и др., но можно построить архитектуру Event Driving *косвенно*, через активизацию конечной точки REST.

Единственно поддерживаемый вид запуска, управляемый напрямую Azure Data Factory, — это триггер, запускаемый по расписанию. Чтобы быть точным, триггер — внутренний сервис Azure Data Factory, который определяет момент времени и периодичность запуска конвейера. Один триггер может служить для запуска нескольких конвейеров. И наоборот, один конвейер может быть запущен несколькими различными триггерами.

Azure Data Factory

Теперь рассмотрим, как создавать и конфигурировать сервис **Azure Data Factory** с помощью веб-портала Azure. В качестве примера возьмем задачу копирования данных из таблицы **Azure Table Storage** в реляционную БД Azure SQL.

Концептуально это ничем не отличается от задачи копирования информации в реляционное хранилище DWH, но по финансовым затратам гораздо ниже.

Итак, сначала следует нажать ссылку добавления новых ресурсов веб-портала управления и ввести в строке поиска Data Factory (рис. 11.2).



Рис. 11.2. Первоначальная страница создания сервиса Azure Data Factory

Azure Data Factory

- После нажатия кнопки Create (Создать) откроется форма конфигурирования сервиса (рис. 11.3). Здесь указываются имя сервиса (Name), подписка Azure (Subscription), ресурсная группа (Resource Group), версия (Version) и местоположение (Location).

Как уже отмечалось, доступны две версии, причем V2 находится в стадии Preview, но с наибольшей долей вероятности станет широкодоступной, поэтому мы ее и рассмотрим.

- Завершив создание этого сервиса, можно перейти на главную панель мониторинга (рис. 11.4).

Но сама по себе эта панель интересна только с точки зрения мониторинга.

- Чтобы настроить Azure Data Factory, следует перейти по внешней ссылке Author & Monitor. В результате откроется внешняя страница конфигурирования вашего экземпляра Azure Data Factory (рис. 11.5).

Azure Data Factory

The screenshot shows the 'New data factory' configuration form. It includes the following fields and options:

- Name:** pokerdatacopy1
- Subscription:** Pay-As-You-Go
- Resource Group:** Create new (selected) / Use existing (radio buttons). Existing resource group: PockerRunExample
- Version:** V2 (Preview)
- Location:** East US
- Pin to dashboard:**
- Buttons:** Create, Automation options

Рис. 11.3. Форма конфигурирования сервиса Azure Data Factory

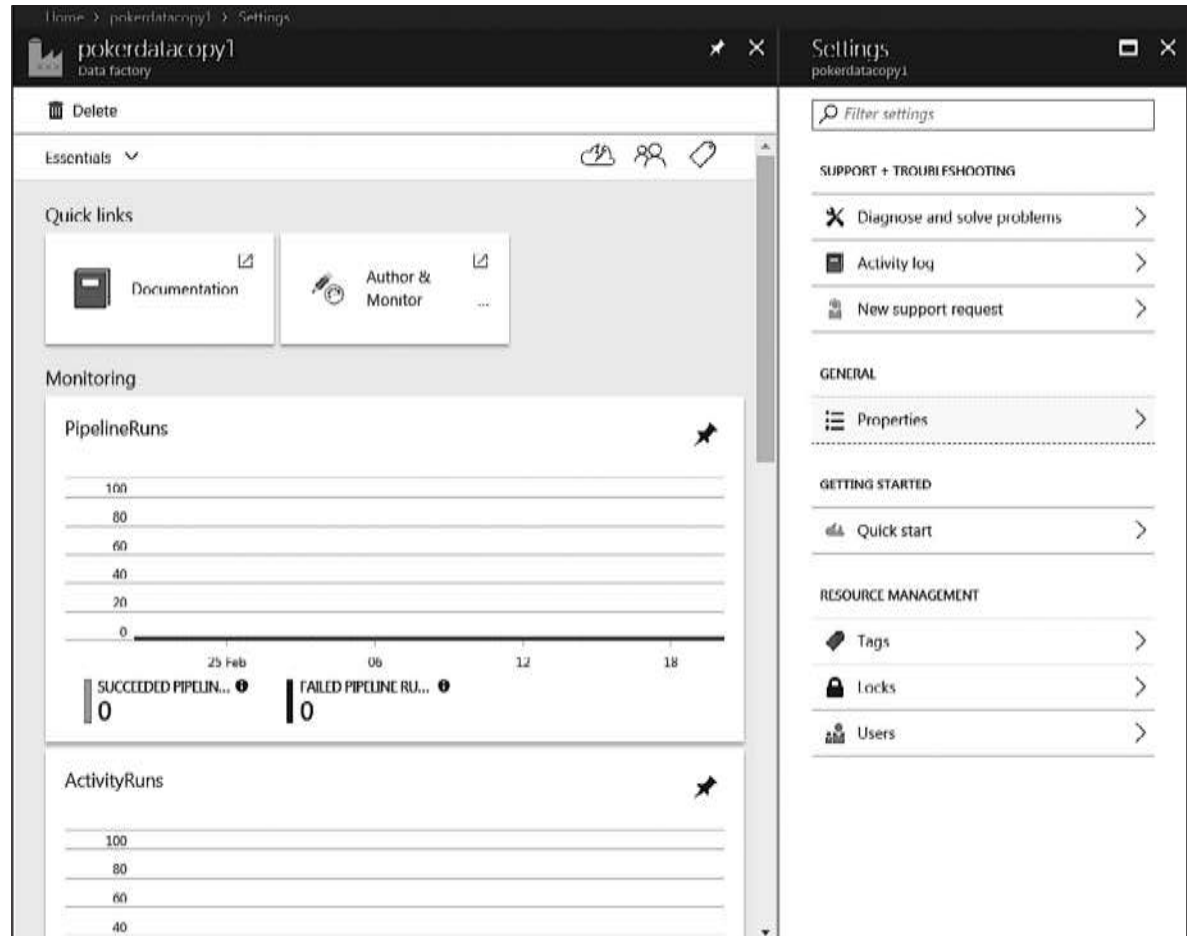


Рис. 11.4. Главная панель мониторинга сервиса Azure Data Factory

Azure Data Factory

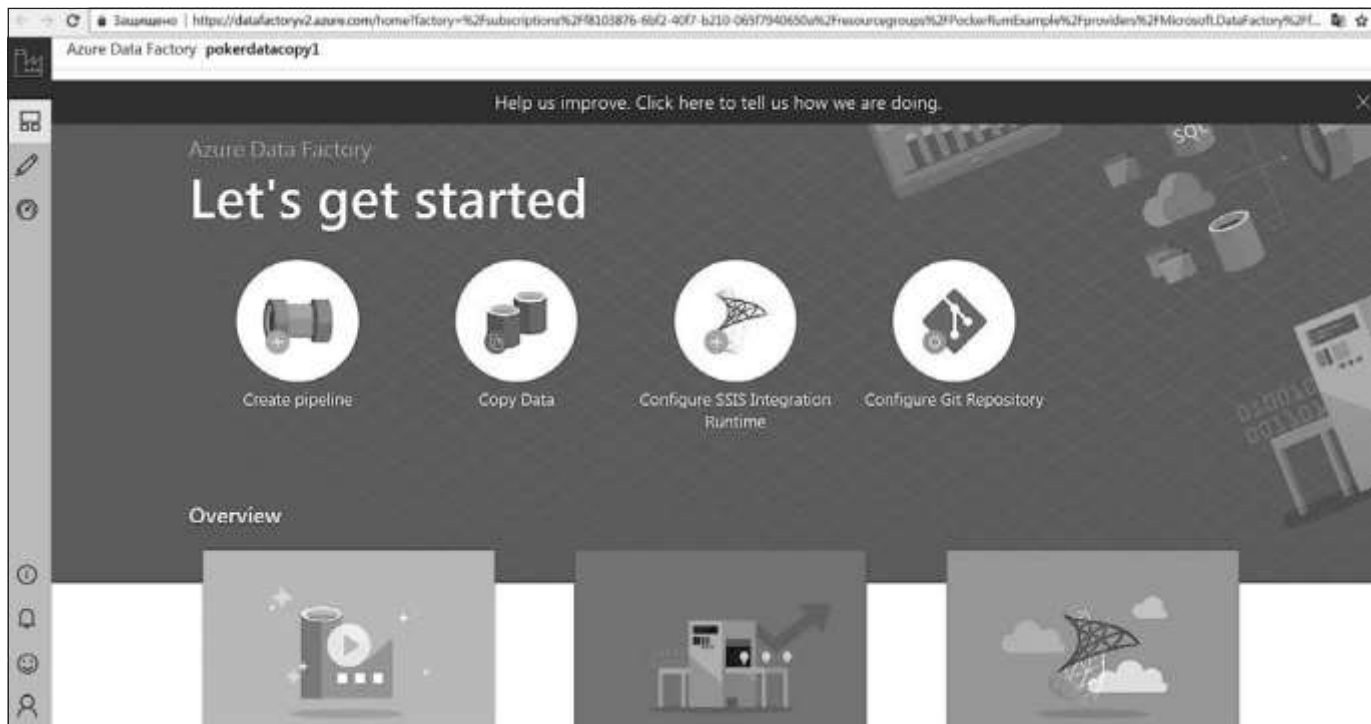


Рис. 11.5. Начальная страница конфигурирования сервиса Azure Data Factory

Прямо на ней имеются ссылки на создание конвейера данных (Create pipeline), копирование данных (Copy Data), настройку интеграции с SSIS (Configure SSIS Integration Runtime) и настройку интеграции с Git (Configure Git Repository).

Пойдем длинным путем и создадим конвейер данных, выбрав ссылку Create pipeline. В результате увидим такую страницу (рис. 11.6).

Azure Data Factory

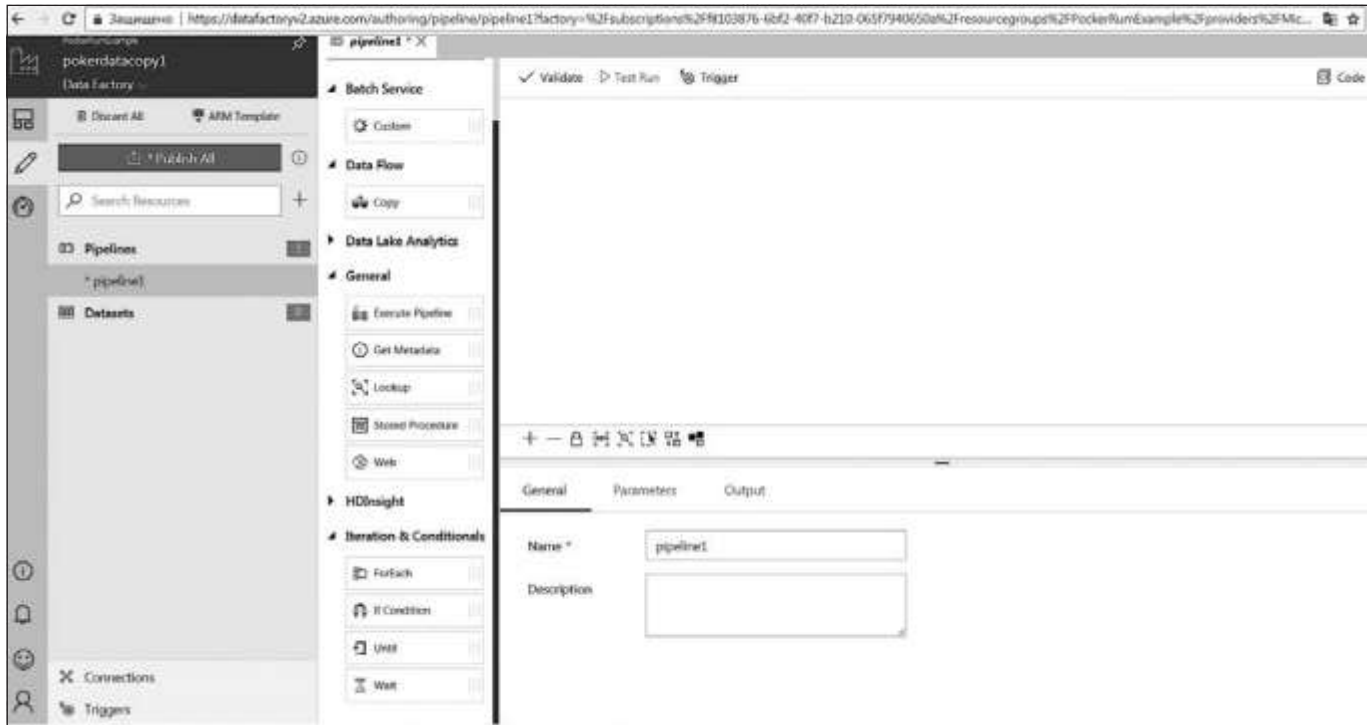


Рис. 11.6. Страница создания и конфигурирования конвейера данных

Здесь мы видим все основные компоненты конвейера данных, включающие в себя активности копирования (Data Flow — Copy), HDInsight, поток управления и пр. Но прежде всего необходимо настроить наборы данных. Для этого следует нажать на ссылку Connections в левом нижнем углу.

Появится следующая форма (рис. 11.7).

Azure Data Factory

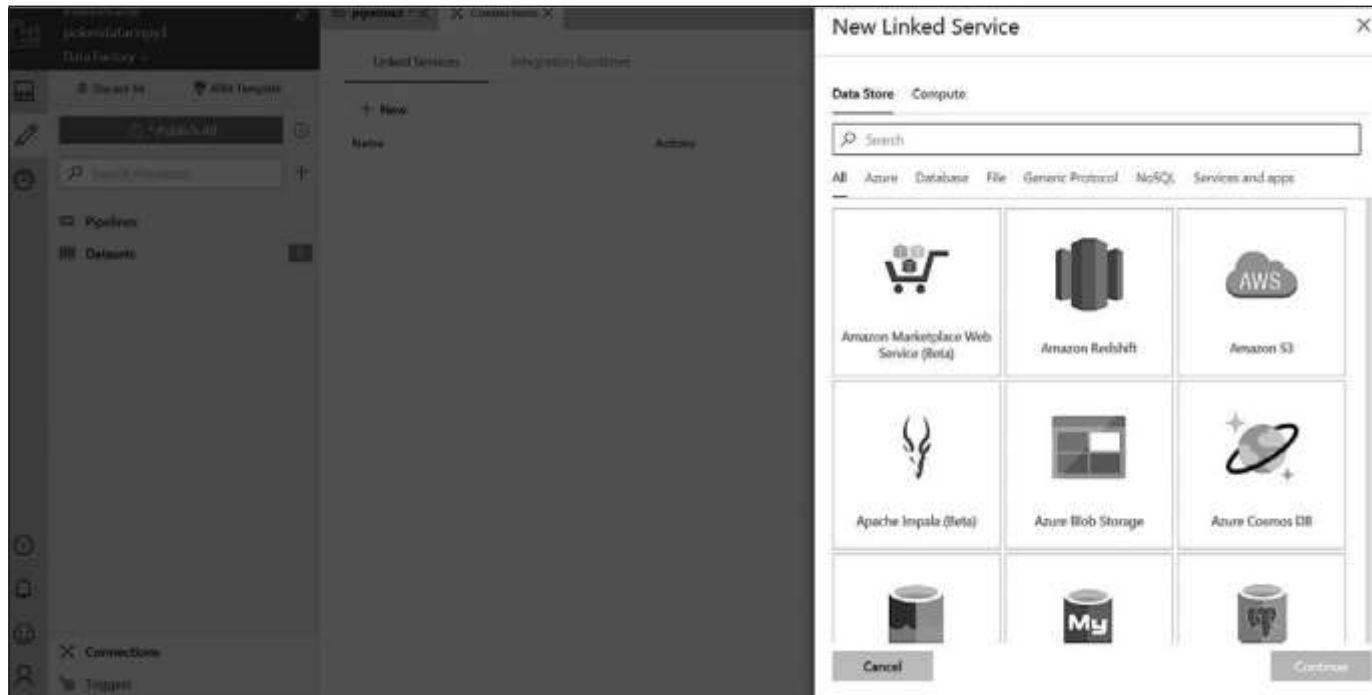


Рис. 11.7. Форма добавления подключенного сервера и настройки наборов данных

Нас интересует источник данных — таблица [Azure Table Storage](#). Для этого в строке поиска следует написать Azure Table Storage, выбрать появившийся значок и нажать его. В результате появится форма конфигурирования (рис. 11.8).

Azure Data Factory

The screenshot shows the 'New Linked Service' configuration window. It contains the following fields and options:

- Name ***: PokerRowTelemetry
- Description**: Poker raw telemetry
- Connect via integration runtime ***: Default
- Account selection method**: From Azure subscription
- Azure subscription**: Pay-As-You-Go (R8103876-6bf2-40f7-b210-065f7940650a)
- Storage account name ***: mainstorage987f2
- Advanced**: collapsed
- Buttons**: Cancel, Test connection (highlighted), Finish
- Status**: Connection successful

Рис. 11.8. Форма конфигурирования подключенного сервера Azure Table Storage

- Здесь указывается имя сервера (PokerRowTelemetry) и дается его описание (**Description**).
- Очень важный параметр — подключение через специальный исполняемый режим (**Connect via integrated runtime**).
- В данном случае следует выбрать режим по умолчанию (Default), поскольку этот подключенный сервис не требует настройки специализированного режима исполнения.
- Далее из подписки Azure (Azure subscription) нужно выбрать наш **Azure Storage Account** и проверить соединение с ним, нажав на ссылку Test connection.
- При успешной проверке соединения следует нажать кнопку **Finish** (Закончить).

Azure Data Factory

Точно так же конфигурируется подключенный сервер Azure SQL (рис. 11.9).
Созданные подключенные серверы можно посмотреть на вкладке Connections (рис. 11.10).

Далее следует перейти на вкладку pipeline1 и вручную перетянуть графический компонент выполняемого задания DataFlow — Copy на общую панель конвейера (рис. 11.11). Нужно дать имя выполняемому заданию, установить тайм-аут (оставим его равным семи минутам), количество попыток повторений в случае отказа (Retry) — два и интервал между попытками — 30 секунд.

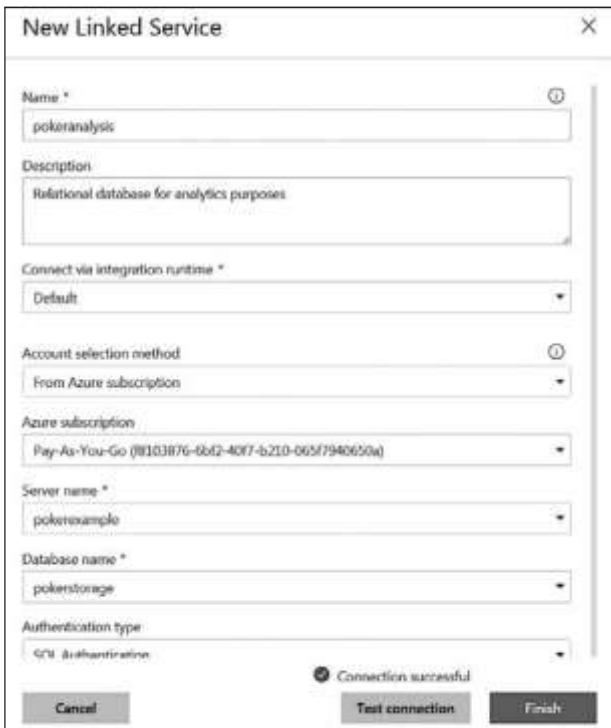


Рис. 11.9. Конфигурирование подключаемого сервера Azure SQL

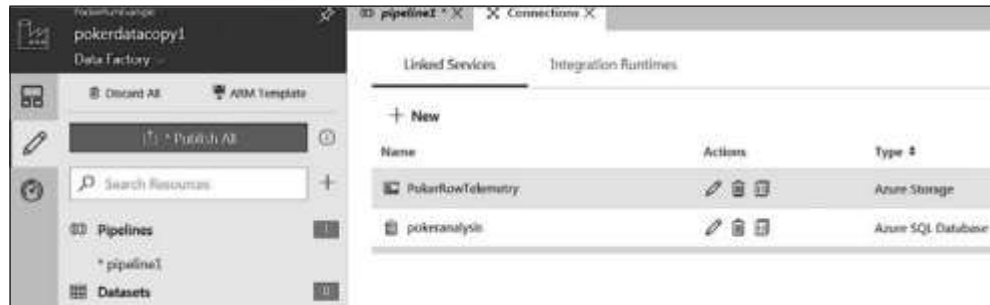


Рис. 11.10. Доступные подключенные серверы

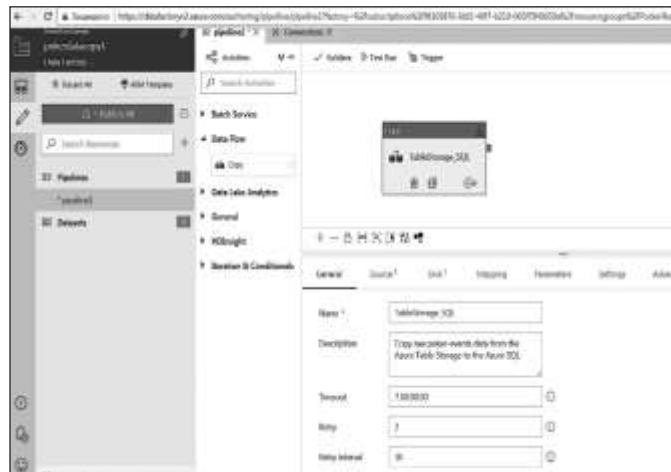


Рис. 11.11. Панель управления копирующей активности

Azure Data Factory

Далее необходимо сконфигурировать источник (Source) и приемник (Sink) данных. Перейдя на вкладку Source, следует нажать на ссылку + New, ввести в появившейся строке поиска SQL и сконфигурировать Azure Table Storage как источник данных (рис. 11.12).

На этой панели нужно выбрать наш подключенный сервер (PokerRowTelemetry), имя таблицы (GameEvents). Далее можно проверить соединение (Test connection) и просмотреть данные (Preview data).

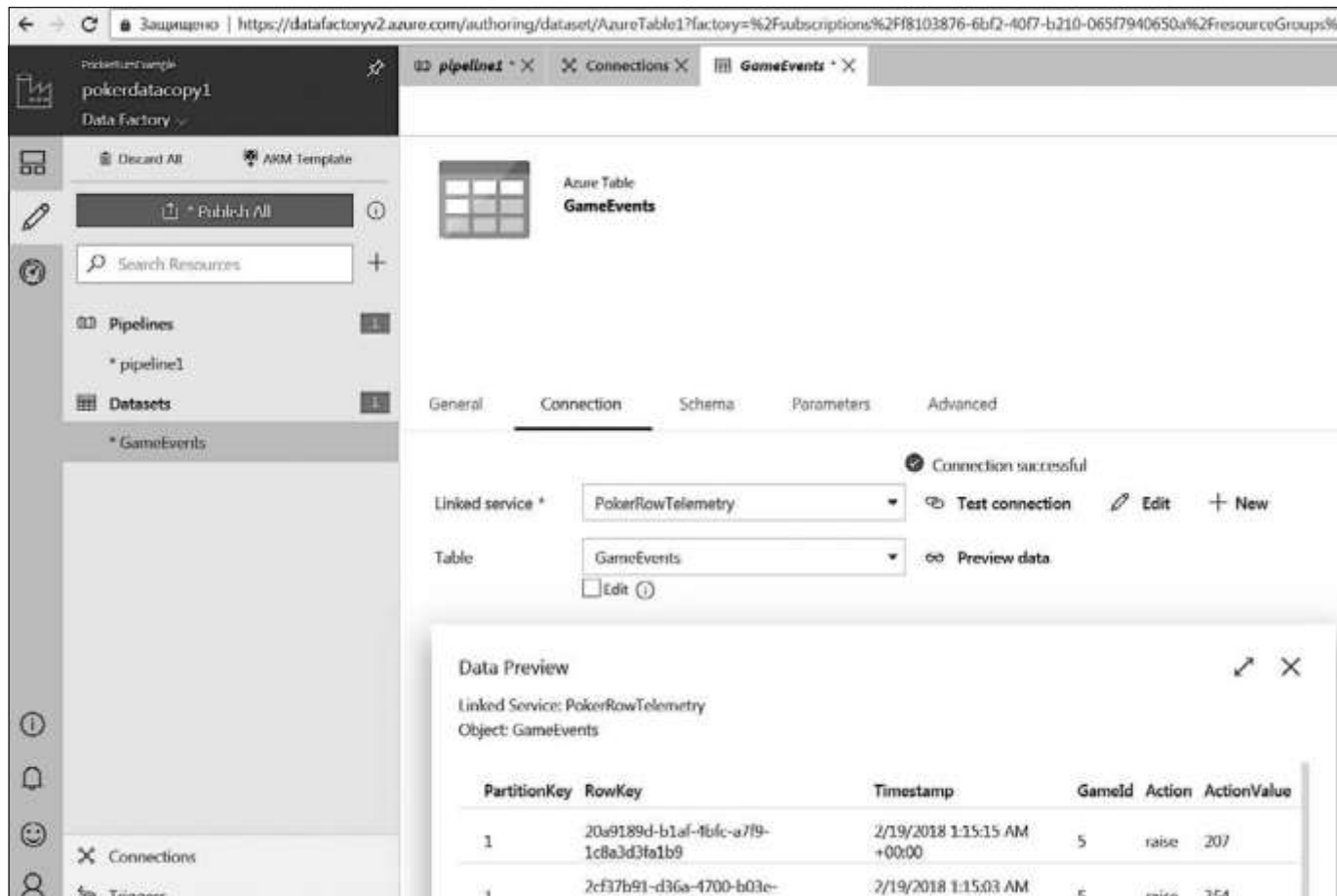


Рис. 11.12. Панель конфигурирования источника данных — Azure Table Storage

Azure Data Factory

Далее в нашей базе данных SQL следует создать таблицу, которая будет хранить данные из **Azure Table Storage**, выполнив следующий сценарий в онлайн-редакторе или в стороннем приложении наподобие **SQL Management Studio** (рис. 11.13). Затем эту таблицу можно подключить как приемник (Sink) таким же образом, как подключали источник (рис. 11.14).

```
1 CREATE TABLE [GameEvents]
2 (
3     [PartitionKey] INT,
4     [GameId] INT,
5     [Action] VARCHAR(20),
6     [ActionValue] INT
7 )
8
9 |
```

Рис. 11.13. Создание таблицы приемника

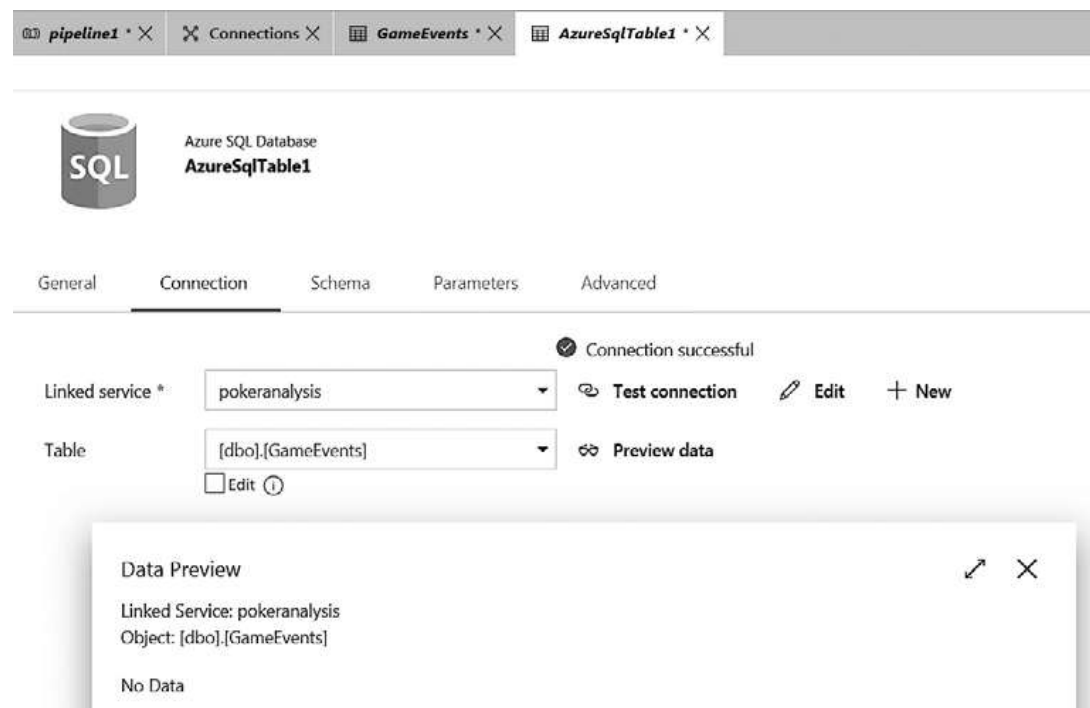


Рис. 11.14. Конфигурирование приемника данных — таблицы Azure SQL

Azure Data Factory

После этого следует перейти на вкладку отображения схем источника и приемника выполняемого задания и при необходимости выполнить согласование схем, выбрав требуемые параметры и указав их типы. Затем, перейдя на вкладку конвейера, можно нажать на ссылку Test Run в верхней части вкладки. Если вы внимательно и правильно все сконфигурировали и настроили, то конвейер отработает успешно (рис. 11.16).

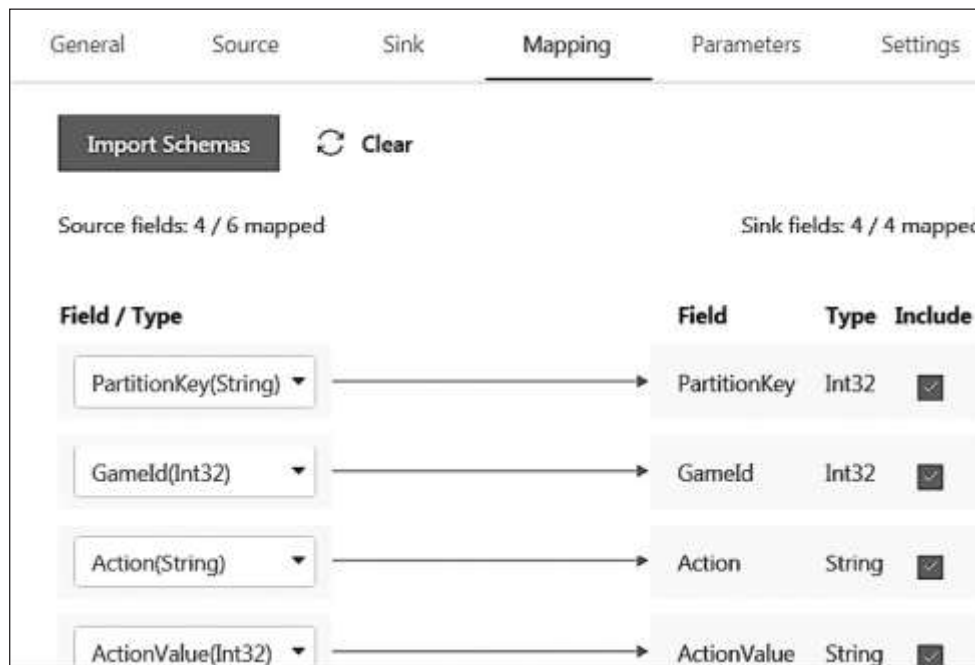


Рис. 11.15. Вкладка отображения схем источника и приемника

Azure Data Factory

Затем, перейдя на вкладку конвейера, можно нажать на ссылку Test Run в верхней части вкладки.

Если вы внимательно и правильно все сконфигурировали и настроили, то конвейер отработает успешно (рис. 11.16). И в итоге можно выполнить запросы к скопированным данным.

На рис. 11.17 показаны примеры запросов, выполненных прямо в онлайн-редакторе портала на вкладке Azure SQL.




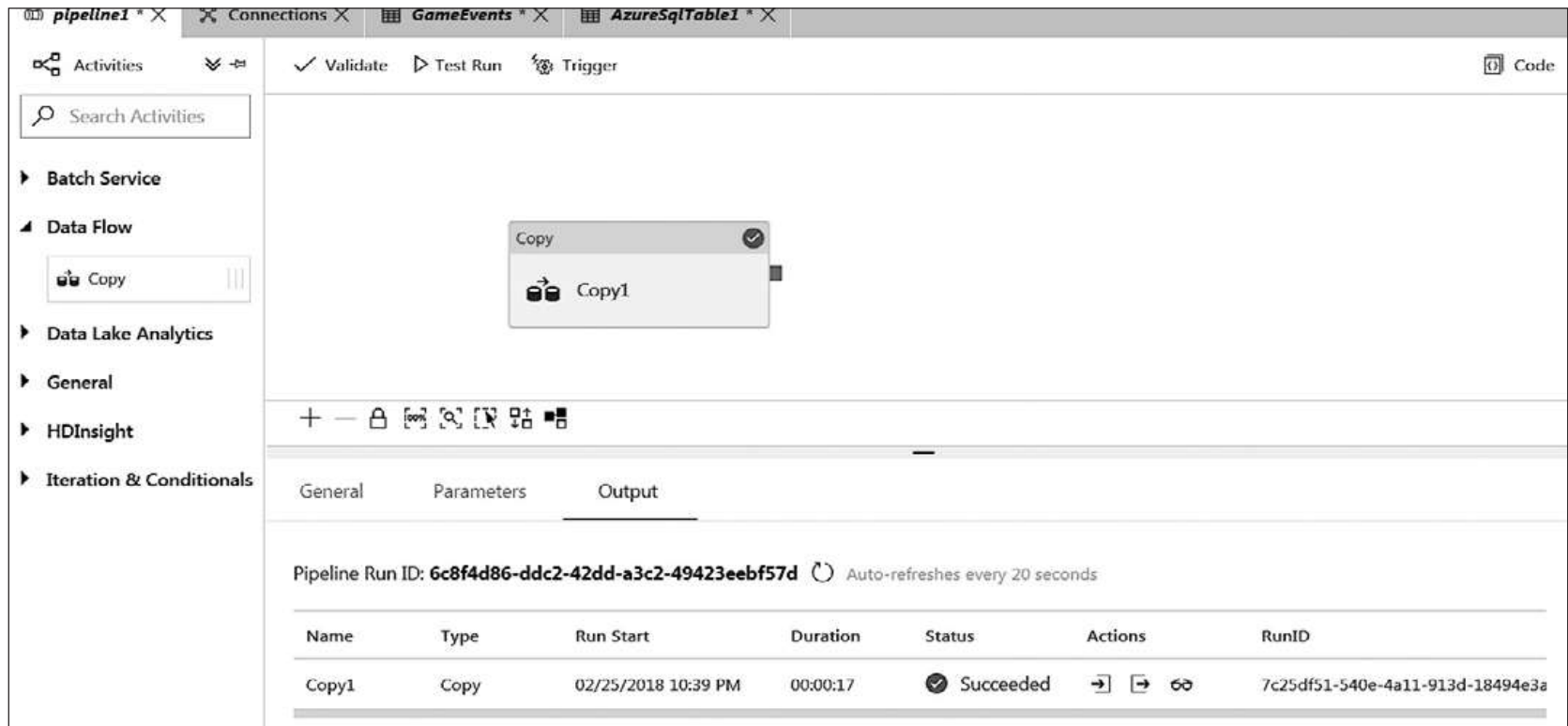
Name	Type	Run Start	Duration	Status	Actions	RunID
Copy1	Copy	02/25/2018 10:39 PM	00:00:17	Succeeded	  	7c25df51-540e-4a11-913d-18494e3a

Рис. 11.16. Результат успешного выполнения копирующего задания Azure Data Factory

Azure Data Factory

На рис. 11.17 показаны примеры запросов, выполненных прямо в онлайн-редакторе портала на вкладке Azure SQL.

В левой половине рисунка подсчитывается суммарное количество ставок, сделанных каждым игроком. В правой изучается статистика типов шагов для заданного игрока.



The screenshot displays the Azure Data Factory interface. The left sidebar shows the 'Activities' pane with 'Copy' selected under the 'Data Flow' category. The main workspace shows a 'Copy' activity named 'Copy1' with a success icon. Below the activity, the 'Output' tab is active, displaying the 'Pipeline Run ID: 6c8f4d86-ddc2-42dd-a3c2-49423eebf57d' and a table of run details.




Name	Type	Run Start	Duration	Status	Actions	RunID
Copy1	Copy	02/25/2018 10:39 PM	00:00:17	Succeeded	  	7c25df51-540e-4a11-913d-18494e3a

Рис. 11.16. Результат успешного выполнения копирующего задания Azure Data Factory

Azure Data Factory

И в итоге можно выполнить запросы к скопированным данным. На рис. 11.17 показаны примеры запросов, выполненных прямо в онлайн-редакторе портала на вкладке Azure SQL.

В левой половине рисунка подсчитывается суммарное количество ставок, сделанных каждым игроком. В правой изучается статистика типов шагов для заданного игрока.

The image shows two side-by-side query editor windows. The left window contains a SQL query that aggregates the total bet amount for each player. The right window contains a SQL query that filters for a specific player and lists the types of bets they made.

Left Query:

```
1 SELECT
2 SUM (ActionValue) AS [Rate sum],
3 [PlayerName] AS [Player name]
4 FROM [GameEvents] AS GE
5 INNER JOIN [Players] AS PL ON GE.PartitionKey = PL.PlayerId
6 GROUP BY PL.PlayerName
7
```

Left Results:

RATE SUM	PLAYER NAME
13148	Alexey Petrov
14352	Dmity Ivanov
9692	Michael Stark
11902	Stepan Egorov

Right Query:

```
1 SELECT
2 COUNT(Action) AS [Game action count],
3 [Action] AS [Game action]
4 FROM [GameEvents] AS GE
5 INNER JOIN [Players] AS PL ON GE.PartitionKey = PL.PlayerId
6 WHERE PL.PlayerId = 1
7 GROUP BY GE.[Action]
8
```

Right Results:

GAME ACTION COUNT	GAME ACTION
4	all-in
18	bet
12	call
10	fold
12	raise

Рис. 11.17. Пример аналитических запросов к полученным данным

Azure Data Factory

После создания и тестирования нужно задеплоить шаблон (по сути, весь этот портал служит графической оберткой для создания JSON-документа), нажав на ссылку Publish All, размещенную на общей панели конвейера rokerdatasoury1. Затем можно создать триггер, который будет запускать этот конвейер в заданные промежутки времени.

Для этого следует нажать на ссылку Trigger в верхней части панели конвейера (рис. 11.18).

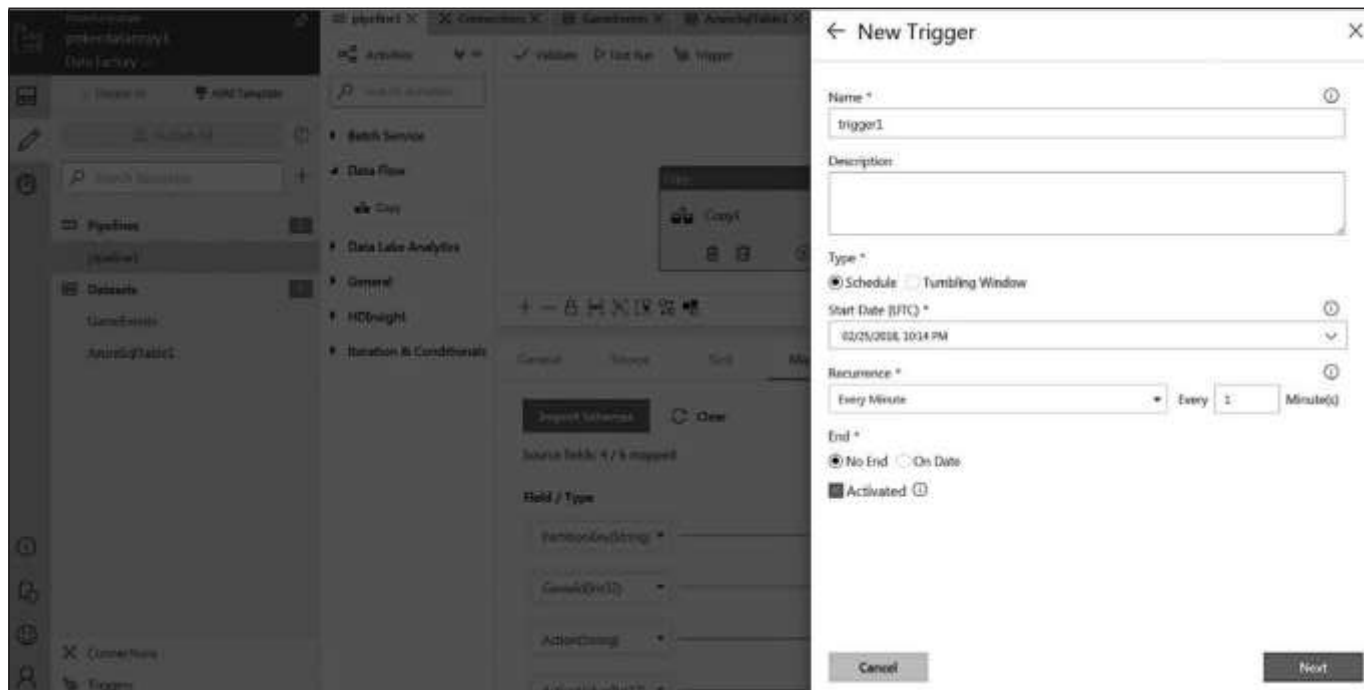


Рис. 11.18. Добавление триггера запуска конвейера данных

Практическая работа №2

Развертывание облачных сервисов копирования и трансформации данных

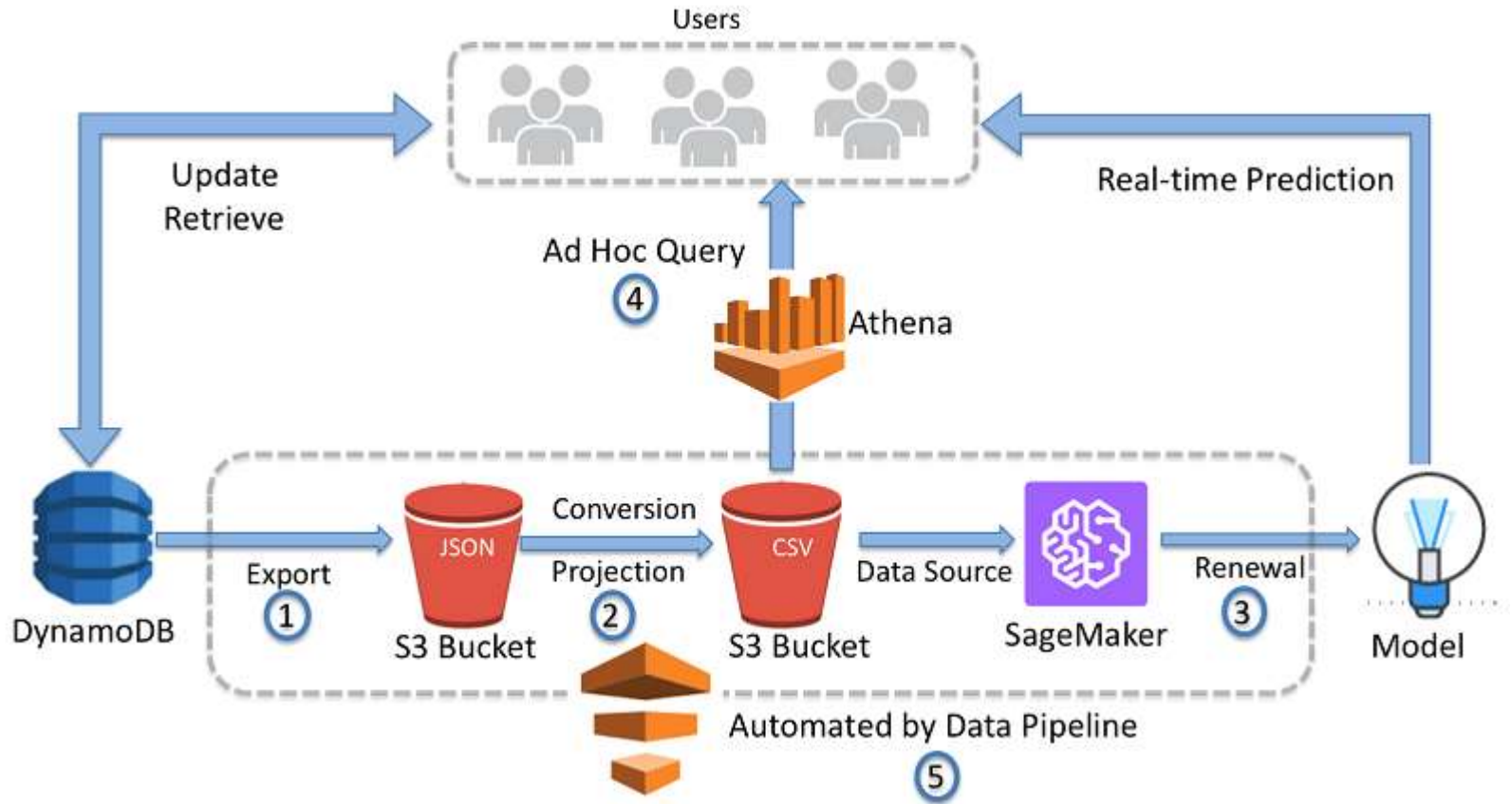
Сервисы трансформации и копирования играют важную роль в построении информационных систем анализа больших данных. Требования к сервисам хранения и анализа противоречивы и не могут быть удовлетворены в рамках одного хранилища, особенно когда данных много. Чтобы удовлетворить эти требования, необходимо использовать хранилища разных типов и сервис, который связывает их информационно.

Сервис [AWS Glue](#) представляет собой удобное средство каталогизации, трансформации и копирования данных, что позволяет одновременно создавать каталоги данных и использовать их для ETL.

Задание:

1. На веб-портале Azure создайте и сконфигурируйте сервис Azure Data Factory. Предложите решение задачи копирования данных из таблицы Azure Table Storage в реляционную БД Azure SQL.
- 2. Разверните сервис AWS Data Pipeline для автоматизации копирования и трансформации данных.**
3. Разверните бессерверный сервис AWS Glue для категоризации данных в каталог данных и выполнения задач ETL. Предложите решение задачи копирования в AWS Redshift данных в виде файла JSON, хранящегося в AWS S3

AWS Data Pipeline



AWS Data Pipeline

Сервис AWS Data Pipeline предназначен для автоматизации копирования и трансформации данных в среде AWS. Он позволяет создавать управляемые конвейеры данных, работа которых будет зависеть от результата выполнения отдельных ступеней этого конвейера. AWS Data Pipeline содержит следующие компоненты.

- **Описание конвейера (pipeline definition)** — JSON-файл с описанием логики переноса данных. Включает в себя:
 - имена, местоположение и форматы источников данных;
 - описание заданий (activities), которые будут преобразовывать данные;
 - планировщики этих заданий;
 - вычислительные ресурсы, которые станут выполнять эти задания;
 - предусловия, которые должны быть удовлетворены перед выполнением заданий;
 - каналы оповещения об изменении статуса выполнения заданий (например, при успешном выполнении задания или сбое).
- **Собственно конвейер (pipeline)**, который планирует и запускает задания. Перед запуском необходимо загрузить его описание. Для запущенного конвейера можно отредактировать описание, загрузить его и снова перезапустить конвейер, чтобы «подхватились» изменения. С конвейером ассоциированы следующие элементы.
 - Компоненты конвейера (pipeline components) представляют собой его бизнес-логику, описанную в разных секциях файла описания. Если конкретнее, то содержат описание источников данных, заданий, параметров планировщика заданий и предусловий для запуска заданий.
 - Экземпляры (instances) — это исполняемые задания, являющиеся компонентами потока исполнения конвейера.
 - Попытки (attempts) — элемент, отслеживающий повторные попытки запуска задания при отказе предыдущей. Количество попыток ограничено параметром, приведенном в файле описания конвейера, однако для разных видов отказов допустимы различные попытки, что и отслеживает этот элемент. При повторном запуске задания используются те же ресурсы (EC2, EMR), что и при предыдущем.

AWS Data Pipeline

Сервис AWS Data Pipeline предназначен для автоматизации копирования и трансформации данных в среде AWS. Он позволяет создавать управляемые конвейеры данных, работа которых будет зависеть от результата выполнения отдельных ступеней этого конвейера. AWS Data Pipeline содержит следующие компоненты.

- **Исполнитель заданий (task runner)** загружает задания из конвейера, затем исполняет их. Например, копирует файлы в S3 EMRFS, а затем запускает EMR для их анализа. Исполнитель заданий автоматически устанавливается и запускается на ресурсах (EC2-экземплярах), приведенных в описании конвейера, или же может быть установлен и сконфигурирован на существующих экземплярах. В качестве исполнителя может выступать приложение, созданное пользователем, или автоматически устанавливаемое приложение, поставляемое AWS Data Pipeline. Исполнитель активно взаимодействует с конвейером данных, забирая исполняемое задание, отсылая статус процесса исполнения и сигнализируя об успешном или неуспешном исполнении (рис. 11.19).
- **Узлы данных (data nodes)** определяют местоположение и тип данных, которые используются как вход и выход конвейера. В настоящее время в качестве узлов выступают:
 - узел DynamoDB (DynamoDBDataNode);
 - узел SQL (SqlDataNode);
 - узел RedShift (RedshiftDataNode);
 - узел S3 (S3DataNode).

Список поддерживаемых баз данных включает в себя AWS RDS, AWS RedShift и базы, поддерживающие стандарт JDBC.

AWS Data Pipeline

Сервис AWS Data Pipeline предназначен для автоматизации копирования и трансформации данных в среде AWS. Он позволяет создавать управляемые конвейеры данных, работа которых будет зависеть от результата выполнения отдельных ступеней этого конвейера. AWS Data Pipeline содержит следующие компоненты.

- **Задания (activities)** — компонент конвейера, определяющий действия, которые необходимо произвести над данными. AWS Data Pipeline содержит несколько готовых к исполнению заданий, представляющих типовые сценарии копирования данных из одного источника в другой, запуска сценария Hive, сценария пользователя (поддерживается Python и Scala) и др. В настоящее время поддерживаются следующие типы заданий:
 - *копирующее задание (CopyActivity)* — представляет собой задание копирования данных из одного источника в другой;
 - *задание EMR (EMRActivity)* — задание, выполняемое в кластере EMR;
 - *задание Hive (HiveActivity)* — задание в виде сценария Hive, выполняемое в кластере EMR;
 - *копирующее задание Hive (HiveCopyActivity)* — задание в виде сценария Hive, выполняемое в кластере EMR, обеспечивающее расширенную поддержку для передачи информации между узлами данных S3 и DynamoDB;
 - *задание Pig (PigActivity)* — задание в виде сценария Pig, выполняемое в кластере EMR;
 - *копирующее задание, исполняемое на RedShift (RedshiftCopyActivity)*, — задание, осуществляющее перенос данных между таблицами AWS Redshift;
 - *задание выполнения сценария командной оболочки (ShellCommandActivity)* — задание в виде пользовательского сценария, выполняемого в командной оболочке Linux/Unix;
 - *задание SQL (SQLActivity)* — задание в виде сценария SQL, выполняемое в базе данных.

AWS Data Pipeline

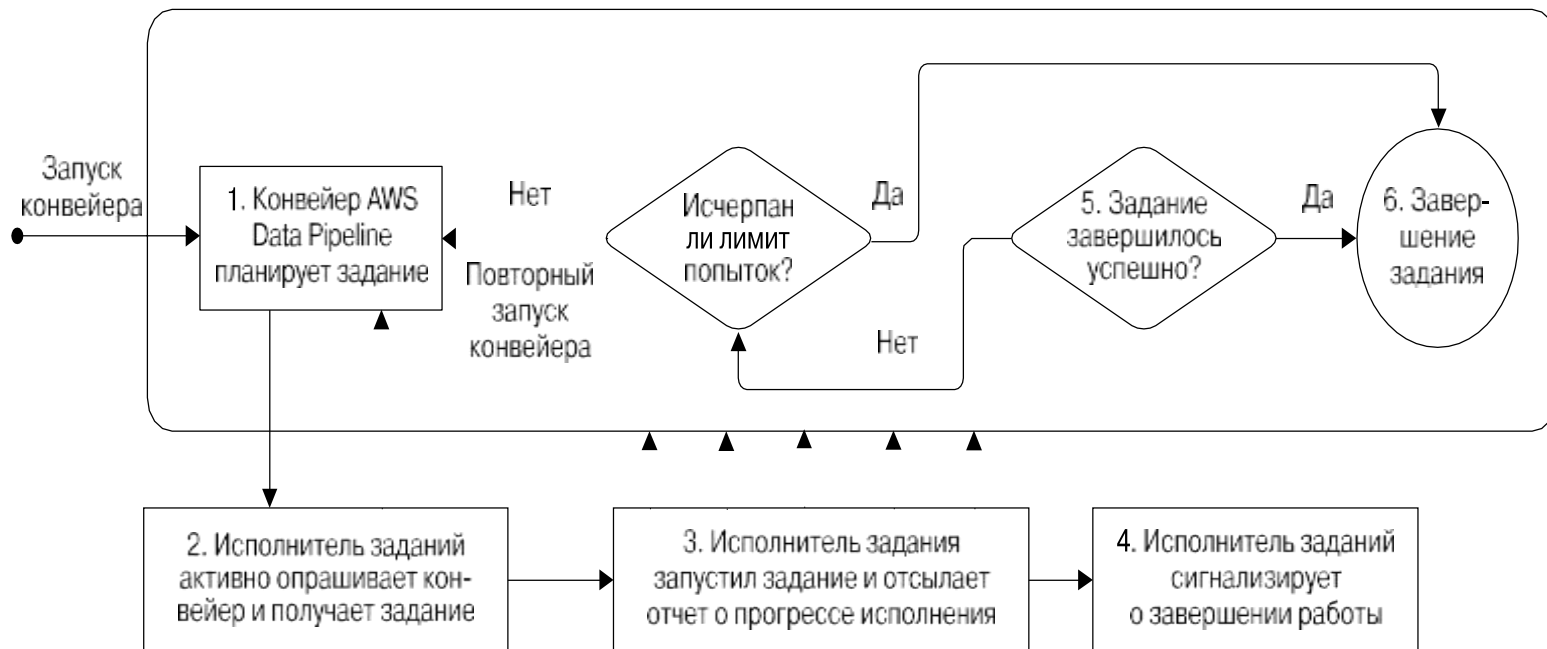


Рис. 11.19. Алгоритм взаимодействия исполнителя заданий с конвейером

AWS Data Pipeline

Сервис AWS Data Pipeline предназначен для автоматизации копирования и трансформации данных в среде AWS. Он позволяет создавать управляемые конвейеры данных, работа которых будет зависеть от результата выполнения отдельных ступеней этого конвейера. AWS Data Pipeline содержит следующие компоненты.

- **Предусловия (preconditions)** — это компоненты конвейера данных, представляющие собой условия, которые должны быть выполнены перед запуском задания. Все они могут быть разделены на две группы: управляемые системой и управляемые пользователем. Первые не требуют дополнительных компьютерных ресурсов для своего функционирования и контролируются самим сервисом AWS Data Pipeline. К таким условиям относятся следующие:
 - `DynamoDBDataExists` — проверка наличия данных в определенной таблице DynamoDB;
 - `DynamoDBTableExists` — проверка существования таблицы DynamoDB;
 - `S3KeyExists` — проверка существования ключа файла в AWS S3;
 - `S3PrefixNotEmpty` — проверка того, что заданный префикс файла существует (то есть не является пустым).

Предусловия, управляемые пользователем, выполняются только на вычислительных ресурсах (EC2-экземплярах), которые пользователь должен явно указать. К этой группе относятся следующие условия:

- `Exists` — проверка существования заданного узла данных;
- `ShellCommandPrecondition` — условие в виде команды оболочки Linux/ Unix.

AWS Data Pipeline

Сервис AWS Data Pipeline предназначен для автоматизации копирования и транс-формации данных в среде AWS. Он позволяет создавать управляемые конвейеры данных, работа которых будет зависеть от результата выполнения отдельных ступеней этого конвейера. AWS Data Pipeline содержит следующие компоненты.

- **Ресурсы (resources)** — вычислительные ресурсы, требуемые для выполнения заданий конвейера. В настоящее время доступны ресурсы в виде EC2-экземпляров и EMR-кластера.
- **Действия (actions)** — шаги конвейера данных, выполняемые его компонентами при наступлении определенного условия, такого как успех, отказ и пр. Сами действия могут состоять в отсылке сообщения в SNS (SnsAlarm) или удалении ресурсов (terminate).

Познакомившись с основными компонентами сервиса AWS Data Pipeline, на конкретном примере рассмотрим, как все это работает в целом.

Прежде всего на панели списка конвейеров следует нажать кнопку Create new pipeline (Создать новый конвейер) (рис. 11.20).



Pipeline ID	Name	Schedule State ▲	Health Status	Creation Time (UTC)
<input type="checkbox"/> df-06994077UCHMCAJWZLY	exportToDynamoDB	SCHEDULED Runs every 100 years	<input type="radio"/> No completed executions	2018-02-26 02:58:28

Рис. 11.20. Список конвейеров данных

AWS Data Pipeline

После этого откроется форма конфигурирования конвейера (рис. 11.21). В ней следует указать источники файла описания конвейера.

The screenshot shows the 'Create Pipeline' form in the AWS console. It includes sections for:
- **Name**: 'Transfer'
- **Description (optional)**: empty
- **Source**: 'Build using a template' with a dropdown menu showing 'Export DynamoDB table to S3'
- **Parameters**: 'Source DynamoDB table name' (PolarEvents), 'Output S3 folder' (s3://exportdynamodb/), 'DynamoDB read throughput ratio' (0.25), and 'Region of the DynamoDB table' (us-east-2)
- **Schedule**: 'Run on pipeline activation'
- **Pipeline Configuration**: 'Logging Enabled' and 'S3 location for logs' (s3://exportdynamodb)
- **Security/Access**: 'IAM roles Default'
- **Tags**: empty table
- **Footer**: 'This pipeline launches an Amazon EMR cluster in your account on every scheduled execution of the pipeline. Normal service charges for this resource will apply in addition to charges for other AWS services used by this pipeline.' and buttons for 'Cancel', 'Edit in Architect', and 'Activate'.

The screenshot shows the 'Schedule' section of the form. It includes:
- **Source**: 'Build using a template'
- **Schedule**: 'You can run your pipeline once or specify a schedule. More'
- **Run**: 'on pipeline activation'
- **Run every**: 'Starting'
- **Ending**: 'never'
- **Frequency**: 'after 1 occurrence(s)'
- **Time**: '2018-02-27 03:14 UTC'
- **Dropdown Menu**: A list of templates including 'Getting Started', 'AWS Command Line Interface (CLI) Templates', 'DynamoDB Templates', 'Elastic MapReduce (EMR) Templates', 'RDS Templates', and 'Redshift Templates'.

Рис. 11.22. Готовые шаблоны трансформации и копирования

Рис. 11.21. Форма конфигурирования конвейера данных

AWS Data Pipeline

Имеется всего три источника:

- можно выбрать из шаблона (Build using a template) (вариант выбора показан на рис. 11.22);
- прямой импорт JSON-файла описания конвейера (Import a definition);
- создание шаблона с использованием специальной панели Build using Architect, которая показана на рис. 11.23.

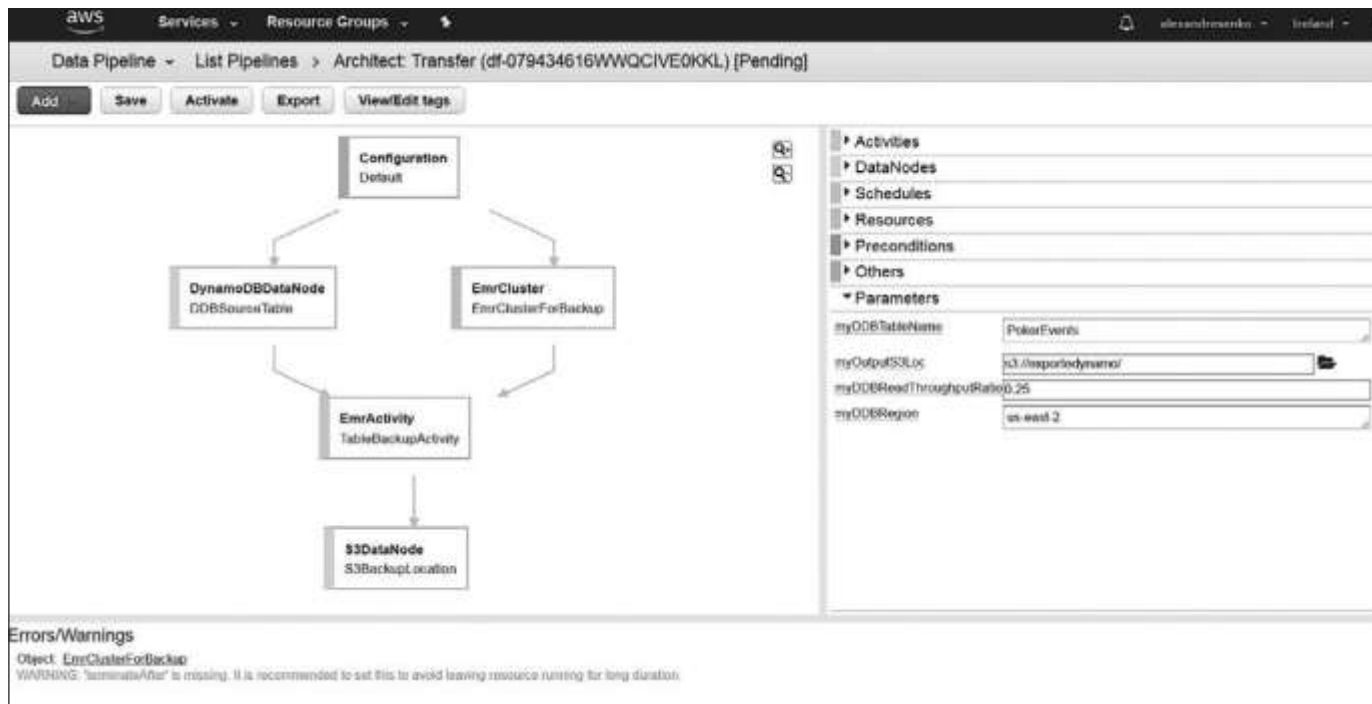


Рис. 11.23. Панель графического конфигурирования файла описания конвейера

AWS Data Pipeline

Для примера на рис. 11.21 показан вариант конфигурирования Data Factory для копирования данных из таблицы PokerEvents DynamoDB (Source DynamoDB table name) в папку S3 (Output S3 folder), указывается регион, где расположена таблица us-east-2 (Region of the DynamoDB table). В поле Schedule можно указать тип запуска: моментальный (on pipeline activation) или по расписанию (on a schedule).

Данная панель позволяет более подробно настроить отдельные компоненты конвейера.

Результаты работы конвейера можно наблюдать на панели списка конвейеров (List Pipelines), показанной на рис. 11.24.

Pipeline ID	Name	Schedule State	Health Status	Creation Time (UTC)			
df-079434616WWQCIVE0KIL	Transfer	PENDING	Pipeline is not active	2018-02-26 03:13:20			
df-00621503CMY7CD6FXGRB	TableCopy	FINISHED Runs every 100 years	ERROR	2018-02-26 03:16:20			
Schedule information		Activity	Type	Health Status	Last Completed Run (UTC)	Active Run (UTC)	Next Run (UTC)
Start 2018-02-26 03:16:25 (UTC) End 2018-02-26 03:16:25 (UTC) Period Runs every 100 years		TableBackupActivity	EmrActivity	ERROR	2018-02-26 03:16:25	2018-02-26 03:16:28	-
Tags View all / Edit		View execution details					
Edit pipeline		View execution details					
df-06904077UCHMCAJWZLY	exportToDynamoDB	FINISHED Runs every 100 years	HEALTHY	2018-02-26 02:58:28			
Schedule information		Activity	Type	Health Status	Last Completed Run (UTC)	Active Run (UTC)	Next Run (UTC)
Start 2018-02-26 02:58:30 (UTC) End 2018-02-26 02:58:30 (UTC) Period Runs every 100 years		ShellCommandActivityObj	ShellCommandActivity	HEALTHY	2018-02-26 02:58:30	2018-02-26 02:58:33	-
Tags View all / Edit		View execution details					
Edit pipeline		View execution details					

Рис. 11.24. Результат работы конвейера Data Pipeline

Практическая работа №2

Развертывание облачных сервисов копирования и трансформации данных

Сервисы трансформации и копирования играют важную роль в построении информационных систем анализа больших данных. Требования к сервисам хранения и анализа противоречивы и не могут быть удовлетворены в рамках одного хранилища, особенно когда данных много. Чтобы удовлетворить эти требования, необходимо использовать хранилища разных типов и сервис, который связывает их информационно.

Сервис [AWS Glue](#) представляет собой удобное средство каталогизации, трансформации и копирования данных, что позволяет одновременно создавать каталоги данных и использовать их для ETL.

Задание:

1. На веб-портале Azure создайте и сконфигурируйте сервис Azure Data Factory. Предложите решение задачи копирования данных из таблицы Azure Table Storage в реляционную БД Azure SQL.
2. Разверните сервис AWS Data Pipeline для автоматизации копирования и трансформации данных.
3. Разверните бессерверный сервис AWS Glue для категоризации данных в каталог данных и выполнения задач ETL. Предложите решение задачи копирования в AWS Redshift данных в виде файла JSON, хранящегося в AWS S3

AWS Glue

AWS Glue — бессерверный сервис, предназначенный для категоризации данных в так называемый каталог данных и выполнения задач ETL. Включает в себя:

- централизованное хранилище метаданных хранилищ данных (AWS Data Catalog);
- подсистему ETL, автоматически генерирующую код на Python или Scala, который описывает трансформацию данных;
- планировщик заданий;
- подсистему мониторинга и перезапуска.

Основной сценарий использования этого сервиса — построение системы доставки данных из различных источников в централизованное хранилище данных. Ключевым компонентом этого сервиса, отличающим его от Data Pipeline, является каталог данных — **Data Catalog**.

Архитектура AWS Glue включает в себя следующие компоненты (рис. 11.25).

- **Выполняемые задания (jobs)** — непосредственно рабочие процессы, производящие требуемое извлечение, трансформацию и загрузку данных из источников в назначения.
- **Сборщик, или краулер (crawler)**, — программа, подключаемая к хранилищам данных, которая используется для наполнения каталога данных метаданными источников. В каталоге сборщику указывается хранилище данных, и он, определив схему источника, создает в каталоге соответствующий объект, называемый таблицей, поскольку все источники данных так или иначе могут быть описаны как таблицы (таблицы SQL, файлы в S3). Вдобавок к описанию таблицы сборщик извлекает дополнительные метаданные, необходимые для построения сценария выполняемого задания ETL.
- **Классификатор (classifier)** — часть сборщика, определяющая схему данных источника. Поддерживает источники типа JSON, CSV, AVRO и XML. Кроме того, имеет возможность работать с реляционными базами данных, поддерживающими протокол JDBC. Пользователь может написать собственный классификатор с помощью специального синтаксиса для описания схемы, например grok.
- **Сценарий трансформации данных** — сценарий, выполняющий задачу трансформации и копирования, генерируемый AWS Glue или предоставляемый пользователем. Для описания задач поддерживаются языки Python, PySpark и Scala.

AWS Glue

- *Триггер (trigger)* — встроенный механизм запуска задания, работающий по расписанию или на основе перехваченного события. При срабатывании триггера происходит основной процесс ETL: на внутренних вычислительных ресурсах, содержащих фреймворк Apache Spark, выполняется сценарий трансформации данных, сгенерированный AWS Glue или предоставляемый пользователем.
- *Сервер Notebook* — поскольку в основе движка AWS Glue лежит Apache Spark, то имеется возможность напрямую взаимодействовать с ним через сервер веб-приложения Notebook (я предпочитаю не переводить, но по сути это блокнот), через который пользователь может интерактивно взаимодействовать с кластером и писать запросы с помощью PySpark.
- *Каталог данных (data catalog)* — централизованное хранилище данных, которое содержит описания таблиц, сгруппированные в базы данных, выполняемых заданий и пр.

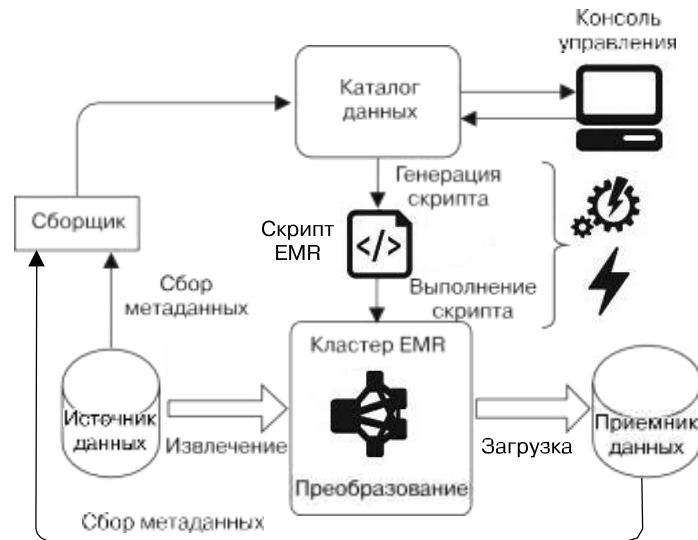


Рис. 11.25. Архитектура сервиса AWS Glue

AWS Glue

Важно четко понимать: таблицы и базы данных в AWS Glue — это объекты хранения метаданных. Они не имеют в своем составе непосредственно данных, а лишь содержат сведения о них (название, описание, ключевые слова, схему и пр.).

Теперь посмотрим на конкретном примере, как работать с сервисом AWS Glue. Поставим задачу скопировать в AWS Redshift данные в виде файла JSON, хранящегося в AWS S3. Этот файл мы получили, экспериментируя с AWS Kinesis Firehouse, доставляя сообщения от работающего bash-скрипта в AWS S3. Консоль управления AWS Glue имеет вид, показанный на рис. 11.26.



Рис. 11.26. Консоль управления AWS Glue

Практическая работа №3

Развертывание облачных сервисов для потокового анализа данных

Потоковый анализ состоит в анализе потока в виде последовательности сообщений, то есть в применении алгоритма анализа к каждому сообщению потока. Этот анализ может состоять в выделении из всего потока сообщений, соответствующих определенным признакам, например, сообщений об ошибках, которые должны быть направлены в отдельное хранилище или сервис, занимающийся обработкой ошибок. Пример прикладной задачи – анализ банковских транзакций и определение в этом потоке потенциально подозрительных (выдача большой суммы в нетипичной для плательщика стране или выдача мелких сумм в разных банкоматах в течение малого интервала времени).

1. Разверните облачный сервис **Azure Stream Analytics** для потоковой обработки Microsoft-сообщений. Создайте экземпляр потокового задания на веб-портале. Обеспечьте интеграцию с сервисом Azure ML: для этого обучите модель с помощью сервиса Azure ML, а затем, после ее размещения в качестве REST-сервиса, получите ее адрес и используйте для интеграции с Azure Stream Analytics.
2. Разверните сервис потокового анализа данных **AWS Kinesis Analytics**. Подключите входной источник. Определите схему сообщений, поступающих на вход сервиса. Сконфигурируйте приложение.

Сравните возможности и ограничения облачных сервисов для потокового анализа данных Azure Stream Analytics и AWS Kinesis Analytics для решения прикладных задач.

Потоковый анализ данных

Потоковый анализ состоит в анализе потока в виде последовательности сообщений, то есть в применении алгоритма анализа к каждому сообщению потока.

Этот анализ может состоять в выделении из всего потока сообщений, соответствующих определенным признакам, например сообщения об ошибках, которые должны быть направлены в отдельное хранилище или сервис, занимающийся обработкой ошибок. Следующий случай — задача маршрутизации сообщений потока в различные направления в зависимости от признаков, ассоциированных с сообщением (например, в зависимости от значения в определенном поле сообщения).

Очень интересный пример потокового анализа сообщений, одновременно относящийся к интеллектуальному, — анализ банковских транзакций и определение в этом потоке потенциально подозрительных (скажем, выдача большой суммы в нетипичной для плательщика стране или выдача мелких сумм в разных банкоматах в течение малого интервала времени и др.).

Потоковый анализ данных

Единица информации, поступающая в систему потокового анализа BigData, — *сообщение*. Это порция информации, состоящая из ряда полей, а именно как минимум из идентификатора, временной метки и собственно поля, несущего полезную информацию (она еще называется *полезной нагрузкой* — payload).

Потоковый анализ данных состоит в анализе потока векторных величин, поскольку каждый элемент данных можно представить векторной величиной, состоящей из набора полей.

Форматы потокового анализа данных могут быть как текстовыми (обычно JSON, CSV), так и бинарными (Apache Avro или ProtocolBuffer). На мой взгляд, самый удобный формат в плане отладки, безусловно, JSON, поскольку он позволяет непосредственно отслеживать сообщения на любом этапе (хранение, отправка в концентратор, прием сервисом потокового анализа, и др.), не прибегая к необходимости программной десериализации.

Облачных сервисов PaaS — они обеспечивают автоматическое масштабирование и требуют минимального администрирования.

- Как правило, сервисы потокового анализа состоят из таких компонентов, как:
- *источники* — входные каналы поступления сообщений. Обычно это концентраторы сообщений или IoT-концентраторы. Возможны и другие сервисы анализа;
- *приемники* — выходные каналы, по которым сообщения будут выходить из сервиса потокового анализа;
- *интеграция* — со сторонними сервисами (например, машинного обучения);
- *выполняемые задания* — алгоритм, применяемый к потоку данных и записанный с помощью специализированного языка, поддерживаемого сервисом потокового анализа.

Широко известные масштабируемые сервисы анализа потоковых данных — Apache Storm, Apache Spark Streaming и Streaming API, входящие в состав Apache Kafka. Apache Spark содержит компонент Spark Streaming, предоставляющий программный интерфейс для работы с потоковыми данными.

Azure Stream Analytics

Итак, рассмотрим облачный сервис потоковой обработки Microsoft-сообщений Stream Analytics. Он позволяет проводить анализ потока сообщений, поступающих на его входы, с помощью специального запроса, написанного с привлечением SQL-подобного синтаксиса. Каждый такой запрос запускается в виде специального задания (job).

Сервис поддерживает следующие форматы сообщений: JSON, CSV и Avro. Источниками сообщений для сервиса потокового анализа могут быть:

- концентратор сообщений (Event Hub),
- шлюз IoT (IoT Hub) или
- напрямую Azure BLOB Storage.

Типовой случай использования источника последнего типа — потоковый анализ лог-файлов, загружаемых в Azure BLOB Storage. В этом случае формат сообщений будет CSV. Применительно к Event Hub или IoT Hub сообщения имеют формат или JSON, или Avro.

Итак, экземпляр сервиса Azure Stream Analytics, называемый *потоковым заданием* (Streaming Job) включает в себя один или несколько входов, запросов и выходов, куда будут направлены результаты применения запроса к одному или нескольким выходным потокам сообщений (рис. 13.1).

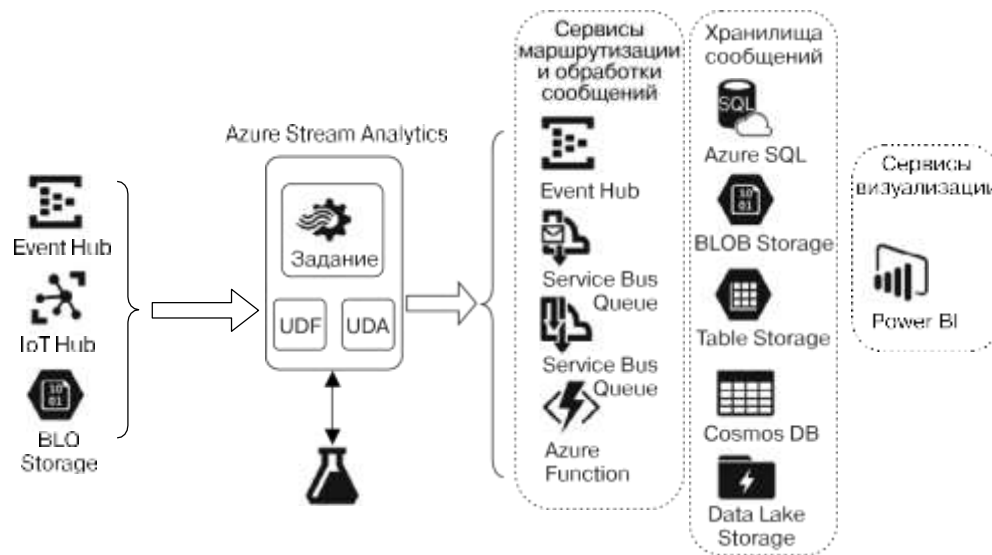


Рис. 13.1. Общая структура сервиса Azure Stream Analytics

Azure Stream Analytics

Все разнообразие выходных потоков может быть разделено на три группы.

- *Сервисы маршрутизации и обработки сообщений* — к ним относятся облачные сервисы, реализующие архитектуру Event Driven. Это концентраторы сообщений (Event Hub) и сервисы шины интеграции (Service Bus Queue и Topic). Предусматривается также прямой вызов бессерверного сервиса Azure Function. Сценарий использования этой группы выходных потоков может быть разным. Например, для построения цепочки потоковых фильтров необходимо соединить их через промежуточные Event Hub. Service Bus Queue или Topic можно применить для интеграции с сервисами обработки и реакции на сообщения, созданные на базе Service Bus SDK и размещенные на виртуальных машинах в облаке или в стороннем окружении. Azure Function может реализовать функцию реакции на сообщения в бессерверном окружении. В настоящий момент не реализована возможность отправки сообщений в сервисы Azure Queue Storage, напрямую в другой экземпляр исполняемого задания и в Azure IoT Hub.
- *Хранилища сообщений* — обеспечивают непосредственное сохранение результатов потоковой обработки в том или ином хранилище. Это может быть Azure SQL, Azure BLOB и Table Storage, Azure CosmosDB и Azure Data Lake Storage. Сохранение сообщений в Azure File Storage не предусмотрено. Подобный вид выхода используется в сценариях, относящихся к BigData: есть входной поток данных, требуется его предварительная фильтрация и сохранение результатов, которые в последующем нужно будет, например, агрегировать за длительный период, прибегнув к потоковому анализу (об этом — в следующей главе).
- *Сервисы визуализации* — при потоковом анализе сообщений зачастую возникают задачи, требующие визуального мониторинга и наблюдения за текущим потоком. Для этого могут использоваться BI-сервисы визуализации, которые и будут графически отображать результаты фильтрации. Наглядный пример подобного применения Azure Stream Analytics — отображение количества ошибок в единицу времени, получаемого путем непрерывной фильтрации лог-файлов, поступающих в BLOB Storage. Само потоковое задание на фильтрацию включает в себя SQL-запрос, выбирающий только те записи логов, которые соответствуют ошибкам, и суммирующий их количество в заданном временно́м окне (о синтаксисе запросов — чуть ниже). В настоящее время прямая интеграция реализована только с сервисом Power BI.

Azure Stream Analytics

Теперь подробнее ознакомимся с тем, как создавать экземпляр потокового задания с помощью портала Azure. Для этого в веб-портале следует нажать ссылку + New и в появившемся окне поиска набрать Stream Analytics Job (рис. 13.2).

Затем должна появиться страничка с описанием Stream Analytics Job. Нажмите кнопку Create (Создать) (рис. 13.3).

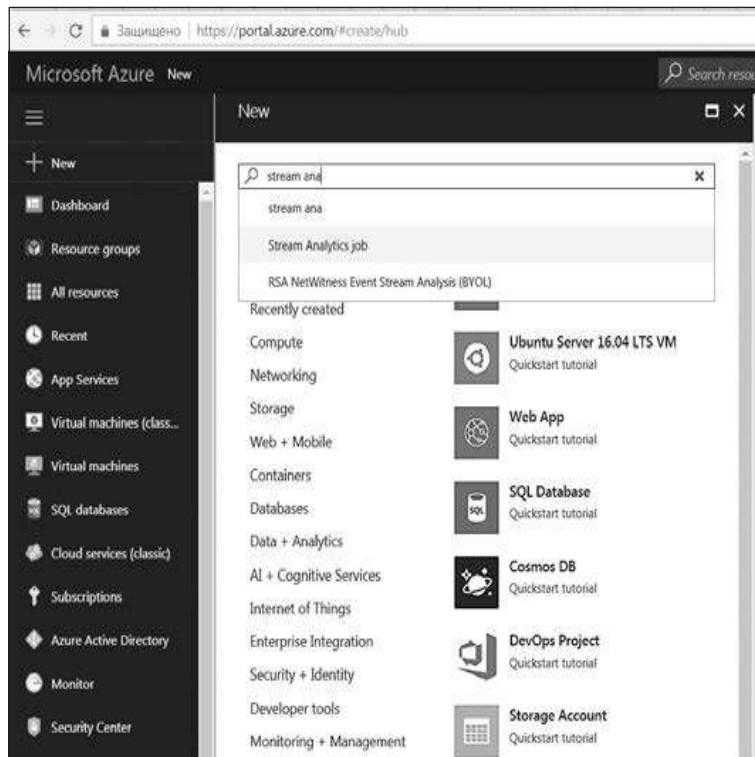


Рис. 13.2. Поиск сервиса Azure Stream Analytics Job среди доступных для создания

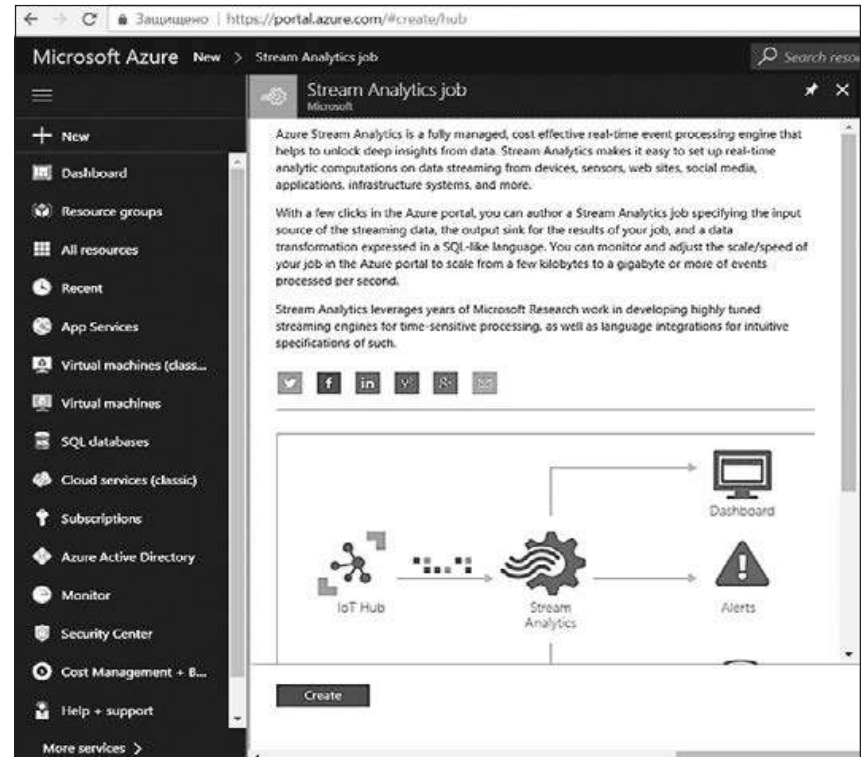


Рис. 13.3. Страница портала с кратким описанием сервиса

Azure Stream Analytics

В появившейся форме (рис. 13.4) следует указать имя вашего сервиса (в нашем примере это mainGameFilter), выбрать существующую или создать новую группу ресурсов (сейчас это общая группа RockerRumExample), задать местоположение дата-центра (в примере это Central US). Далее нужно выбрать облако как хостинг (Hosting Environment — Cloud) и установить флажок Pin to Dashboard (Поместить виджет на диаграмму). Пока что сервис не настроен и содержит пустую стартовую страницу с примером запроса (рис. 13.5).

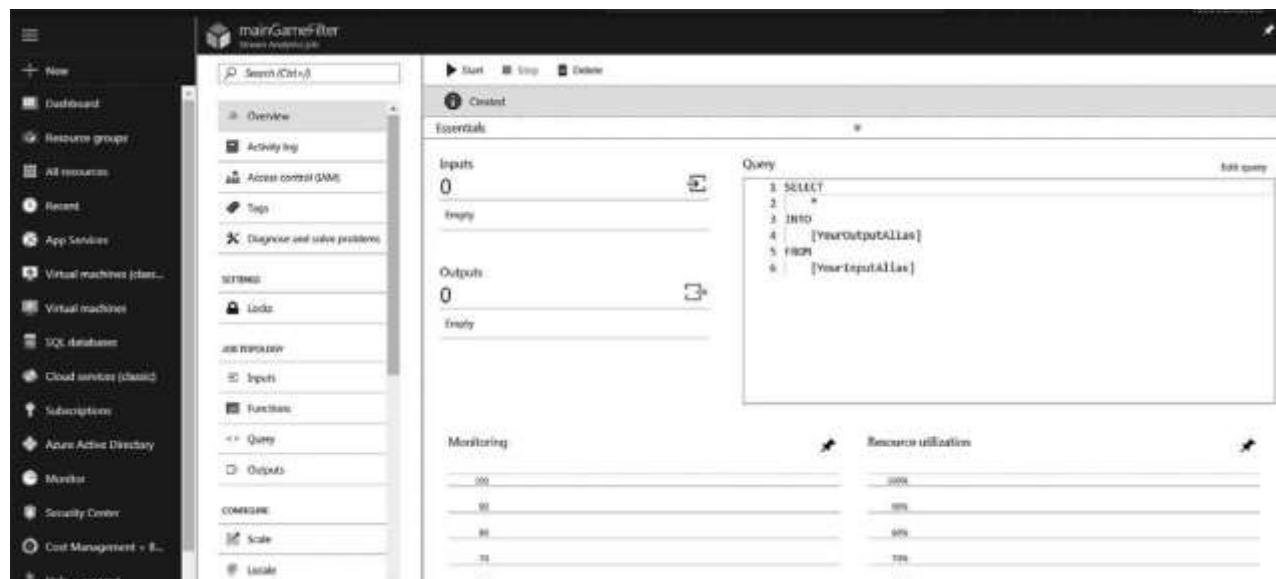
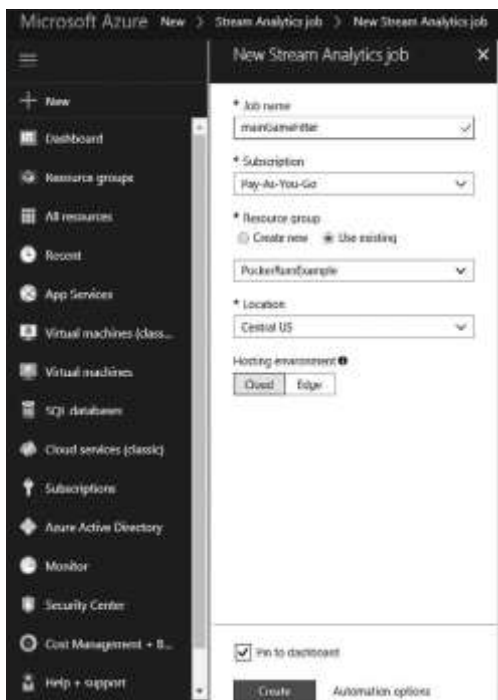


Рис. 13.5. Начальная страница сервиса Stream Analytics Job

Рис. 13.4. Начальная форма сервиса Stream Analytics Job

Azure Stream Analytics

Помимо стандартных аналитических операторов SQL, язык **Azure Stream Analytics** поддерживает широкий набор встроенных функций.

- *Агрегирующие функции* (aggregate functions) выполняют агрегацию набора данных — вычисление среднего значения (AVG), подсчет количества (COUNT), вычисление максимального (MAX) и минимального (MIN) значений.
- *Аналитические функции* (analytic function) являются специфическими для Azure Stream Analytics и обрабатывают именно сообщения, в частности вычисляют аномалии в потоках данных.
- *Функции работы с массивами* (array functions) обрабатывают входные данные типа массива.
- *Функции преобразования типов* (cast function) преобразуют типы данных (CAST, TRY_CAST) или возвращают их тип (GetType).
- *Функции, работающие с временем и датой* (date and time functions) оперируют данными временного формата, в частности вычисляют разность между временными моментами (DATEDIFF), получающие значение дня (DAY), месяца (MONTH), года (YEAR) и пр.
- *Функции, работающие с геопространственными данными* (geospatial functions), облегчают анализ потоковых геопространственных данных.
- *Математические функции* (mathematical functions) выполняют стандартные математические операции (SIN, ABS и др.) над входными элементами данных.
- *Строковые функции* (string functions) работают со строковыми данными.
- *Специальные функции* — группы функций, работающих с различными специальными аспектами сервиса Azure Stream Analytics, например с метаданными источника.

Очень мощный механизм расширения языка **Azure Stream Analytics** — **определяемые пользователем функции (user defined functions, UDF)**. Они создаются на языке JavaScript, и из них напрямую доступны как стандартные функции языка JavaScript, так и встроенные функции Azure Stream Analytics.