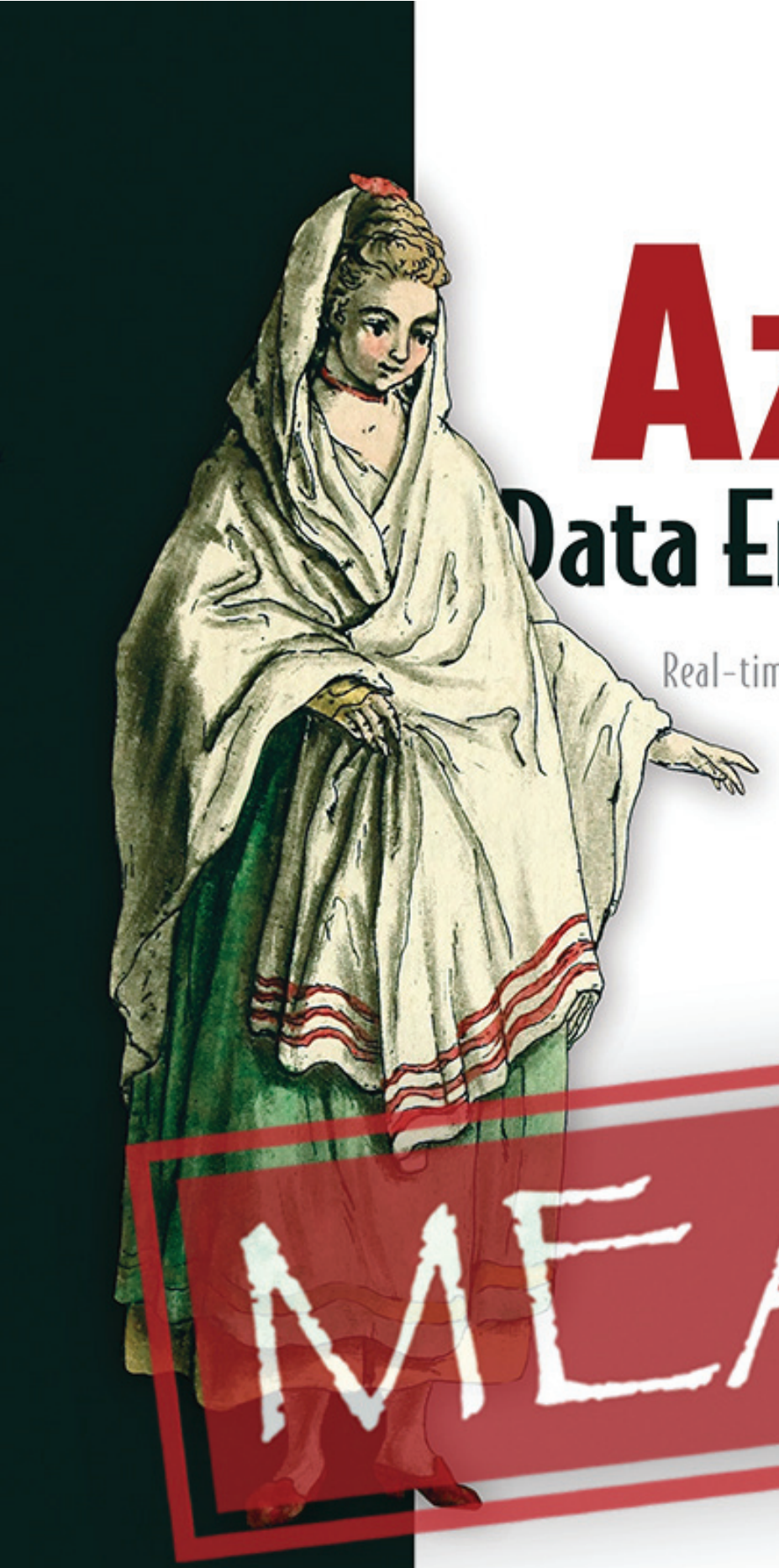


Azure Data Engineering

Real-time, streaming, and batch analytics

Richard Nuckolls

MEAP

MANNING

MEAP Edition
Manning Early Access Program
Azure Data Engineering
Real-time, streaming, and batch analytics
Version 2

For more information on this and other Manning titles go to
manning.com

welcome

Thanks for purchasing the MEAP for *Azure Data Engineering*.

I've spent my career working with Microsoft technologies. Chances are you have too. When my company decided to build a new analytics system, Microsoft Azure was a natural fit. But all the "Big Data" systems use Apache this and open-source that. After years of building PaaS apps in Azure, endless configuration and scattered single-use tools looks like work.

Luckily, Azure offers a set of PaaS services for building high-capacity analytics systems with the easy integration and familiar tool sets we want. This book covers these building block capabilities: storage, ingestion, stream processing, batch processing, querying, and automation. The chapters cover specific services, including Azure Storage, Data Lake, Event Hub, Data Lake Analytics, Stream Analytics, SQL Data Warehouse, and Data Factory. Examples in each chapter add pieces to build a working analytics system, and teach the fundamentals of using each technology. Since these are Microsoft technologies, you'll get to choose between GUI and command-line.

To get the most benefit from this book, you'll need established skill in writing SQL queries and managing SQL databases. You'll want some basic coding skills too, so Powershell scripts and short C# methods won't be unmanageable. Finally, you'll need a subscription to Azure to follow along with the examples. But don't worry, the examples won't cost more than a few cups of coffee.

Please leave questions and comments in the Manning's liveBook's Discussion Forum. Tell me when you've lost the thread. After working with these services so long, it's easy to forget how the breadth of options challenge the newcomer. You can help me introduce a new group of data engineers to Azure!

— Richard Nuckolls

brief contents

1 What is data engineering

This chapter covers:

- What is data engineering?
- What do data engineers do?
- How does Microsoft define data engineering?
- What tools does Azure provide for data engineering?

Data collection is on the rise. More and more systems are generating more and more data every day. According to Marz,

> *More than 30,000 gigabytes of data are generated every second, and the rate of data creation is only accelerating.*
>
> *– Nathan Marz Big Data (2)*

Increased connectivity has led to increased sophistication and user interaction in software systems. New deployments of connected "smart" electronics also relied on increased connectivity. In response, businesses have increased the amount of data being collected, stored, and aggregated by their products. This has led to an enormous increase in compute and storage infrastructure. The scale of data collection and processing requires a change in strategy for processing the data. Businesses are challenged to find experienced engineers and programmers to develop the systems and processes to handle this data. The new role of data engineer has evolved to fill this need. The collection, preparation and querying of this mountain of data using Azure services is the subject of this book. The reader will be able to build working data analytics systems in Azure after completing the book.

According to Gartner,

Big Data is high volume, high velocity, and/or high variety information assets that require new forms of processing to enable enhanced decision making, insight discovery, and process optimization.

– Mark A. Beyer The Importance of Big Data

1.1 What is data engineering?

Collecting the data seems like a simple activity. Take reporting web site traffic. A single user, during a site in a web browser, requests a page. A simple site could respond with an HTML file, a CSS file, and an image. This example could represent one event, three events, or four events. What if there is a page redirect? That is another event. What if we want to log the time taken to query a database? What if we retrieve some items from cache? All of these pieces of data are common log data point today.

Now add more user interaction, like a comparison page with multiple sliders. Each move of the slide logs a value. Tracking user mouse movement returns hundreds of coordinates. Consider a connected sensor with a 100Hz sample rate. It can easily record over 8 million measurements a day. When you start to scale to thousands and tens of thousands of simultaneous events every point in the pipeline must be optimized for speed until the data comes to rest. Once at rest the data must remain secure.

Data engineering is the practice of building data storage and processing systems. Robert Chang, in his "A Beginners Guide to Data Engineering," describes the work as designing, building, and maintaining data warehouses. Data engineering creates scalable systems which allow analysts and data scientists to extract meaningful information from the data.

1.2 What do data engineers do?

Most businesses have multiple sources generating data. Manufacturing companies track the output of the machines, the output of employees, the output of their shipping departments. Software companies track their user actions, their software bugs per release, their developer output per day. Service companies check number of sales calls, time to complete tasks, usage of parts stores, and cost per lead. Some of this is small scale; some of it is large scale.

Analysts and managers might operate on narrow data sets, but large enterprises increasing want to find efficiencies across divisions, or find root causes in multi-faceted systems failures. In order to extract value from these disparate sources of data, engineers have built large scale storage systems as a single repository of data. A software company may implement centralized error logging. The service company may integrate their CRM, billing, and finance systems. Engineers need to plan for the ingestion pipeline, the storage Systems, and reporting services across multiple groups of stakeholders.

As a first step data consolidation means a large relational database. Analysts review reports, CSV

files, and even spreadsheets in Excel, in an attempt to get clean and consistent data. Often developers or database administrators prepare scripts to import the data into databases. In the best case experienced database administrators define common schema and plan partioning and indexing. The database enters production. Data collection commences in earnest.

Typical data systems based on storing data in relational databases have problems with scale. A single database instance, which is the simplest implementation, always becomes a bottleneck given increased usage. There are a finite amount of CPU cores and drive space available on a single database instance. Scaling up can only go so far before IO bottlenecks prevent meeting response time targets. Distributing the database tables across multiple servers, or sharding, can enable greater throughput and greater storage at the cost of greater complexity. Even with multiple shards, database queries under load display greater and greater latency. Eventually query latency grows too large to satisfy the requirements of the application.

The open source community answered the challenge of building web-scale data systems. Hadoop opens vast disk storage for access with ease of maintenance. Spark provides a fast and highly available logging endpoint. NoSQL gives users a way to access large stores of data quickly. Languages like Python and R make deep dives into huge flat files possible. Analysts and data scientists write algorithms and complex queries in order to draw conclusions from the data. But this new environment still requires system administrators to build and maintain servers in their data center.

1.3 How does Microsoft define data engineering?

The systems architecture using these new open source tools looks quite different from the traditional database-centric model. In his landmark book, Nathan Marz coined a new term: Lambda Architecture. He defined this new architecture, "general-purpose approach to implementing an arbitrary function on an arbitrary dataset and having the function return its results with low latency" (Marz, 7). The goals of Lambda architecture cover many of the inherent weaknesses of the database centric model. Figure 1.1 shows a general view of the new approach to saving and querying data. Data flows into both the Speed layer and the Batch layer. The Speed layer prepares data views of the most recent period in real time. The Serving layer delivers data views over the entire period, updated at regular intervals. Queries get data from the Speed layer, the Serving layer, or both, depending on the time period queried.

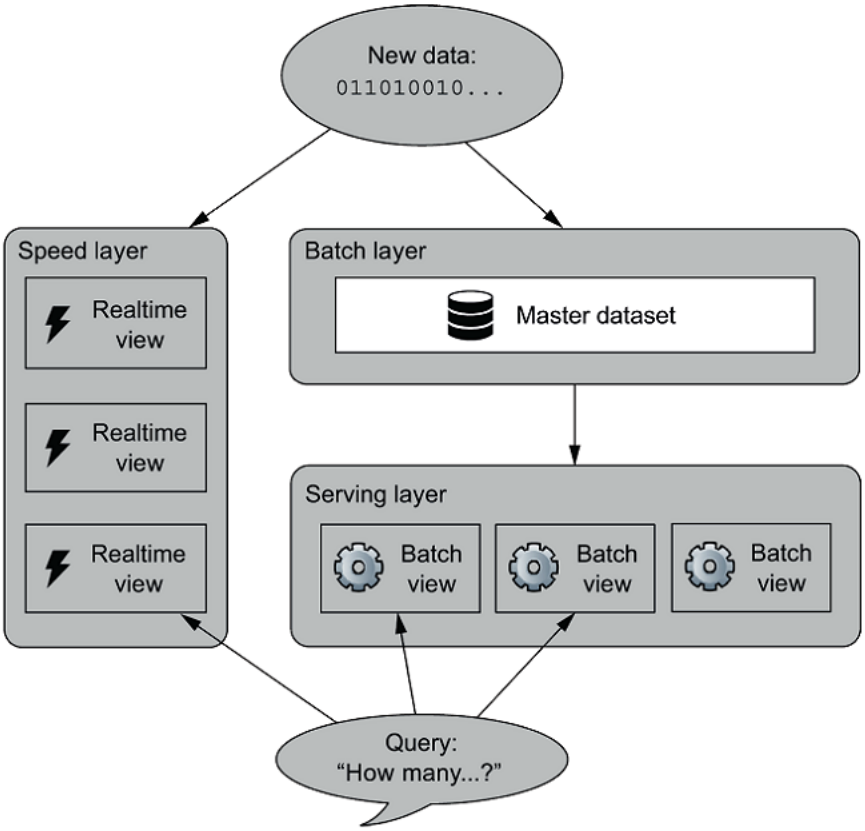


Figure 1.1 Lambda analytics system, showing logical layers of processing based on query latency

Figure 1.2 describes an analytics system using a lambda architecture. Data flows through the system from acquisition to retrieval via two paths: batch and stream. All data lands in long term storage, and scheduled or ad-hoc queries generate refined data sets from the raw data. This is the batch process. Data with short time windows for data retrieval run through an immediate query process, generating refined data in near-real time. This is the stream process.

1. Data is generated by applications, devices, or servers.
2. Each new piece of data is saved to long-term file storage.
3. Each new piece of data is also sent to a stream processer.
4. A scheduled batch process reads the raw data.
5. Both stream and batch processes save query output to a retrieval endpoint.
6. Users query the retrieval endpoint.

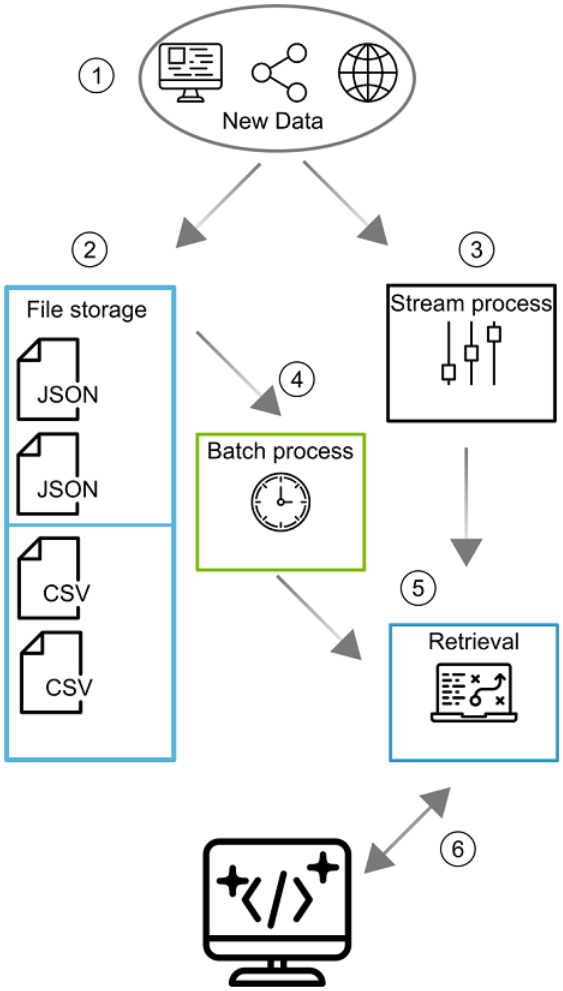


Figure 1.2 Lambda Architecture

Figure 1.2 shows the core principle of Lambda architecture: data flows one way. Only new data is added to the data store; raw data is never updated. Batch processes yield data sets by reading the raw data, and deposit the data sets in a retrieval layer. A retrieval layer handles queries.

Faults caused by human error account for the largest problems in operating an analytics system. Lambda architecture mitigates these errors by storing the original data immutably. An immutable data set, where data is written once, read repeatedly, and never modified, does not suffer from corruption due to incorrect update logic. Bad data can be excluded. Bad queries can be corrected and run again. The output information remains one step removed from the source. In order to facilitate fast writes, new bits of data are only appended. Updates to existing data doesn't happen. To facilitate fast reads, two separate mechanisms converge their outputs. The regularly scheduled batch process generates information as output from queries over the large data set. Between batch executions, incoming data undergoes a similar query to extract information. These two information sets together form the entire result set. An interface allows retrieving the combined result set. Because writes, reads, queries, and requests handling execute as distributed services across multiple servers, the Lambda architecture scales both horizontally and vertically. Engineers can add both more and more powerful servers. Because all of the services operate as

distributed nodes, hardware faults are simple to correct, and routine maintenance work has little impact on the overall system. Implementing a Lambda architecture achieves the goals of fault tolerance, low latency reads and writes, scalability, and easy maintenance.

Six functions make up the core of this architectural design pattern. Mike Wassen describes the architecture pattern for Microsoft in the "Big data architecture style" guide at docs.microsoft.com/azure/architecture/guide/architecture-styles/big-data. Lambda architecture maps onto this style using the following functions.

1.3.1 Data acquisition

Large scale data ingestion happens one of two ways: a continuous stream of discrete records, or a batch of records encapsulated in a package. Lambda architecture handles both methods with aplomb. Incoming data in packages is stored directly for later batch processing. Incoming data streams are processed immediately, and packaged for later batch processing. Eventually all data becomes input for query functions.

1.3.2 Data storage

Distributed file systems decouple saving data from querying data. Data files are collected and served by multiple nodes. More storage is always available by adding more nodes. The Hadoop Distributed File System (HDFS) lies at the heart of most modern distributed storage systems designed for analytics.

1.3.3 Data processing

A distributed query system handles partitioning a single query into multiple executable units and executing them over multiple files. In Hadoop analytics systems the MapReduce algorithm handles distributing a query job over multiple nodes as a two step process. Each Hadoop cluster node maps (Map) requested data to a single file, and the query returns results from that file. The results from all the files are combined, and the resulting set of data is reduced (Reduce) to a set fulfilling the query. Multiple cluster nodes divide the Map tasks and Reduce tasks between them. The MapReduce algorithm enables efficient querying of large scale collections. New queries can be set for scheduled updates or submitted for a single result. Multiple query jobs can run simultaneously, each using multiple nodes.

1.3.4 Data queries

A real time analysis engine monitors the incoming data stream and maintains a snapshot of the most recent data. This snapshot contains the new data since the last scheduled query execution. Queries update result sets in the data retrieval layer. Usually these queries duplicate the syntax or output of the batch queries over the same period.

1.3.5 Orchestration

A scheduling system runs queries using the distributed query system against the distributed file system. The output of these scheduled queries becomes the result set for analysis. More advanced systems include data transfers between disparate systems. The orchestration function typically moves result sets into the data retrieval layer.

1.3.6 Data retrieval

Lastly, an interface for collating and retrieving results from the scheduled and snapshot data sets gives the end user a low latency endpoint for information. This layer often relies on ubiquitous SQL to return results to analysis tools. Together these functions fulfill the requirements of the data analysis system.

1.4 What tools does Azure provide for data engineering?

Cloud systems promise to solve challenges with processing large scale data sets.

- Processing power limitations of single-instance services
- Storage limitations and management of on-premises storage systems
- Technical management overhead of on-premises systems

By using Azure cloud technologies, many difficulties in building large scale data analytics systems on-premises are eliminated. By automating the setup and support of servers and applications, the expertise of system administrators can be used elsewhere. Ongoing expense of hardware can be minimized. Redundant systems are provisioned as easily as single instances. The packaged analytics system is easy to deploy.

Several cloud providers have abstracted the complexity of the Hadoop cluster and its associated services. Microsoft responded to the development of the Hadoop ecosystem with cloud-based systems hosted in Azure. The system is branded HDInsight. According to Jason Howell, HDInsight is "a fully managed, full spectrum, open source analytics service for enterprises." (par. 1). The data engineer can build a complete data analytics system using HDInsight, and the common data tools associated with Hadoop. Many data engineers, especially those familiar with Linux and Apache Software, choose HDInsight when building a new data warehouse in Azure. Configuration approaches, familiar tools, Linux-specific features, and Linux-specific training materials are some of the reasons engineers familiar with Linux would choose HDInsight.

Microsoft also built a set of abstracted services in Azure which perform the functions required for a data analysis system, but without the distinct paradigm of Linux and Apache. Along with the services, Microsoft provides a reference architecture for building a big data system. The model guides engineers through some high-level technology choices when using the Microsoft tools.

A big data architecture is designed to handle the ingestion, processing, and analysis of data that

is too large or complex for traditional database systems.

– Mike Wasson Big data architecture style (1)

This model covers common elements of the Lambda architecture, including data storage, batch and stream processing, and variations on an analysis retrieval endpoint. The model describes additional elements that are necessary but not defined in the Lambda model. For robust and high performance ingestion, a message queue can pass data to both the stream process and the data store. A query tool for data scientists gives access to aggregate or processed information. An orchestration tool schedules data transfers and batch processing.

Microsoft lays out the skills and technologies used by data engineers using Azure services as part of its certification for "Azure Data Engineer Associate." Azure Data engineers are described as those who "design and implement the management, monitoring, security, and privacy of data using the full stack of Azure data services to satisfy business needs." (par. 1) This book focuses on the Microsoft Azure technologies described in this certification. This includes Event Hubs, Stream Analytics, Data Lake Store and Storage accounts, SQL Database, and Data Factory. Data engineers can use the services to build big data analytics solutions in Azure.

1.5 Azure Data Engineers

Microsoft Azure gives engineers resources to build new systems without requiring new on-premise hardware and software installs. Platform as a service (PAAS) tools in Azure allow engineers familiar with Microsoft technologies to build new systems without requiring any hardware or software support. While HDInsight provides an open source architecture for handling data analysis tasks, Microsoft Azure also provides another set of services for analytics. For engineers familiar with Microsoft languages like C# and T-SQL, Azure hosts several services which can be linked to build data processing and analysis systems in the cloud.

Using the Microsoft toolset in Azure for building a large scale data analysis system requires some basic and intermediate technical skills. First, SQL is used extensively for processing streams of data, for batch processing, for orchestrating data migrations, and for managing SQL databases. Second, CSV and JSON files facilitate transferring data between systems. Data engineers must understand the strengths and weaknesses of these file formats. Reading and writing these files are core activities of the batch processing workflows. Third, the Microsoft data engineer should be able to write basic C# and JavaScript functions. Several cloud tools, including U-SQL, are extendable using these languages. Processing functions and helpers can run in Azure and be triggered by cloud service events. Lastly, experience with the Azure portal and familiarity with the Azure CLI or Powershell allows the engineer to create new resources efficiently.

1.6 Example Application

Marz defines the function of the data analytics system this way: "A data system answers questions based on information that was acquired in the past up to the present" (Marz, 6). In this book, you will build a practical web-scale data analytics system using Azure cloud technologies. You will base your design on the principles of the Lambda architecture. The system will provide a scalable queue for inbound messages, and a data store for loading data files. The system will collect data and store it immutably. The system will allow batch processing of queries over the entire dataset. The system will schedule the batch executions and move data into the retrieval endpoint. Concurrently, incoming data will stream into the retrieval endpoint.

Figure 1.3 shows a diagram of your application using Azure technologies. Let's look closer at the proposed system.

1. Sources like Azure Functions, Azure Event Hubs SDK code, or API calls submit messages to the Event Hubs queue.
2. Stream Analytics subscribes to the Event Hubs topic and continually reads the incoming message queue.
3. Stream Analytics writes to a JSON file in the Data Lake Store. Stream Analytics creates a new file each hour.
4. Stream Analytics writes an aggregate query result record to the SQL Data Warehouse each minute.
5. Data Lake Analytics reads the new JSON file from the Data Lake Store each hour, and outputs an aggregate file to the Data Lake Store.
6. Data Factory reads the new aggregate file from the Data Like Store, delete the previous hour's data from the SQL Data Warehouse, and writes an aggregate query result record to the SQL Data Warehouse for each minute of the hour.

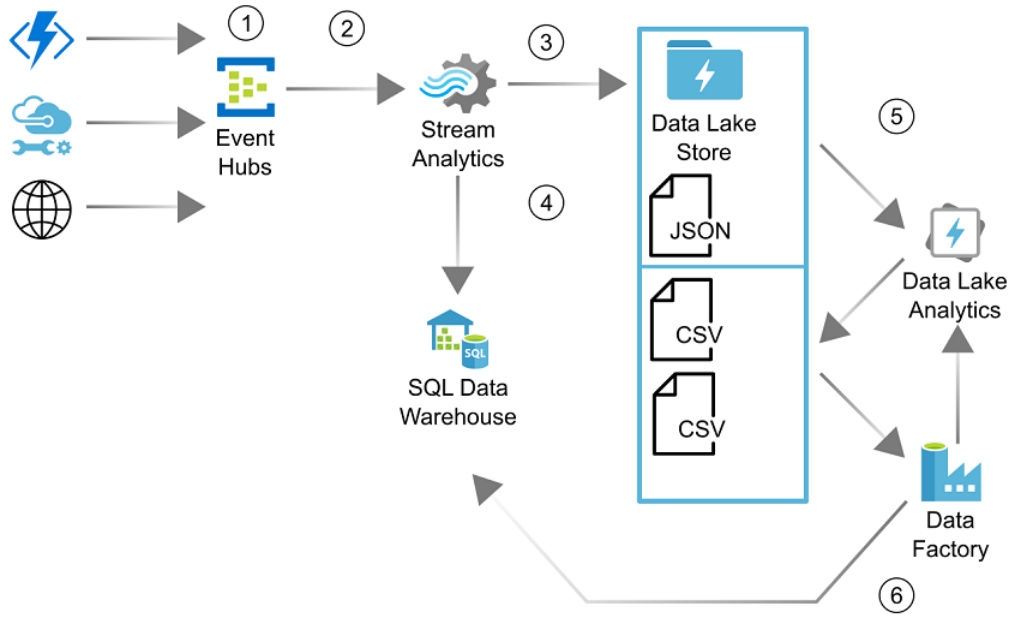


Figure 1.3 Azure Cloud Services Lambda Architecture

The SQL Data warehouse provides a familiar endpoint for querying aggregate data. Engineers and data scientists can submit new queries to the Stream Analytics and Data Lake Analytics to generate new data sets. They can run SQL queries against existing data sets in the Data Warehouse with low latency. So the proposed system fulfills the requirements of a Lambda architecture big data system.

In order to build this analytics system, you'll need an Azure subscription. Signing up for a personal account and subscription takes an email address and a credit card. You may want to install Powershell Core on your development computer. Nearly all of the examples in this book can be done using the Azure Portal. Powershell scripts, with the Azure Powershell module, allow a more repeatable process for creating and managing Azure services. A recent version of an integrated development environment (IDE) like Visual Studio is optional, if you want to build the C# code examples.

1.7 Summary

Many challenges come with the growing data collection and analysis efforts at most companies. Older systems struggle under increased load. Battling shortages of space and time take up valuable developer resources. Increased usage drives increased disruption of unplanned outages. Risk of data loss is always present. The database-centric model for data analysis systems no longer meets the needs of the business.

Businesses need a new approach for handling web-scale data volumes. The Lambda architecture separates data processing into a hot path and a cold path. It reduces system complexity by minimizing the effort required for low latency queries. It scales horizontally simply and easily because its distributed nature underpins the entire application. Batch processing can run multiple processes concurrently on multiple nodes. Multiple stream readers keep the data for queries updated in near-realtime. Lambda architecture solves many problems of the database-centric model.

Building a Lambda architecture analytics system with cloud technologies reduces low value work for engineers even further. New servers can be provisioned in mere minutes. Most applications can be purely in the cloud, without concern for underlying server infrastructure. Data storage scales without limit. Microsoft packaged the popular open-source Hadoop cluster services into a cloud service called HDInsight. Data engineers and scientists familiar with these open-source tools gain all the benefits of cloud services when using HDInsight.

For data engineers and scientists preferring Microsoft tools, Azure provides equivalent technologies for building a web-scale data analytics system. Event Hubs, Stream Analytics, Data Lake and Data Lake Analytics, SQL Data Warehouse, and Data Factory connect in an

interoperable web. These technologies form the building blocks we'll use to build our general purpose web-scale data analytics system. We'll dig deep into setup and configuration of these services in the following chapters. The next chapter introduces each service.

2 Building an analytics system in Azure

This chapter covers:

- Introducing the six Azure services we'll use to build our analytics system
- Joining the services into a working analytics system
- Calculating fixed and variable costs of these services
- Applying Microsoft big data architecture best practices

Cloud providers offer a wide selection of services to build a data warehouse and analytics system. Some services are familiar incarnations of on-premises applications: virtual machines, firewalls, file storage, and databases. Increasing in abstraction are services like web hosting, search, queues, and application containerization services. At the highest levels of abstraction are products and services that have no analogue in a typical data center. For example, Azure Functions executes user code without setup of underlying servers, runtimes, or program containers. Moving workloads to more abstract services reduces or eliminates setup and maintenance work, and brings higher guaranteed service levels. Conversely, more abstract services remove access to many configuration settings, and constrain usage scenarios. This chapter introduces the Azure services we'll use to build our analytics system. These services are abstract to very abstract implementations, which allows you to focus on functionality immediately without needing to spend time on the underlying support systems.

2.1 Lambda architecture

Lambda architecture seeks to combine the best of real-time processing and fast querying, with the ability to query over huge amounts of collected data. All of the data that enters the system gets a time stamp. The time stamp gives an order to all the data, and various processing steps out the data into multiple time windows based on the time stamp. Data follows two paths through the system. The real-time "hot" path prepares data for querying with low latency, on the order of seconds or minutes. The hot path has access to the most recent data; therefore its calculations are accurate over a short window of time, but may not be accurate over all time. The batch "cold" path prepares data over the entire data window. The cold path has access to all the data before the batch execution, so the calculations are accurate up to the time of the last batch. Typical latency for the cold path is on the order of hours or days. Together the hot and cold paths contain data from all the time windows.

In an analytics system designed with a Lambda architecture, user queries are submitted to two processors, depending on the targeted time window. For real-time or low latency data sets, a "speed layer" returns query results from the hot path. For longer windows of time, a "serving layer" returns results from various batch processes which cover the time window.

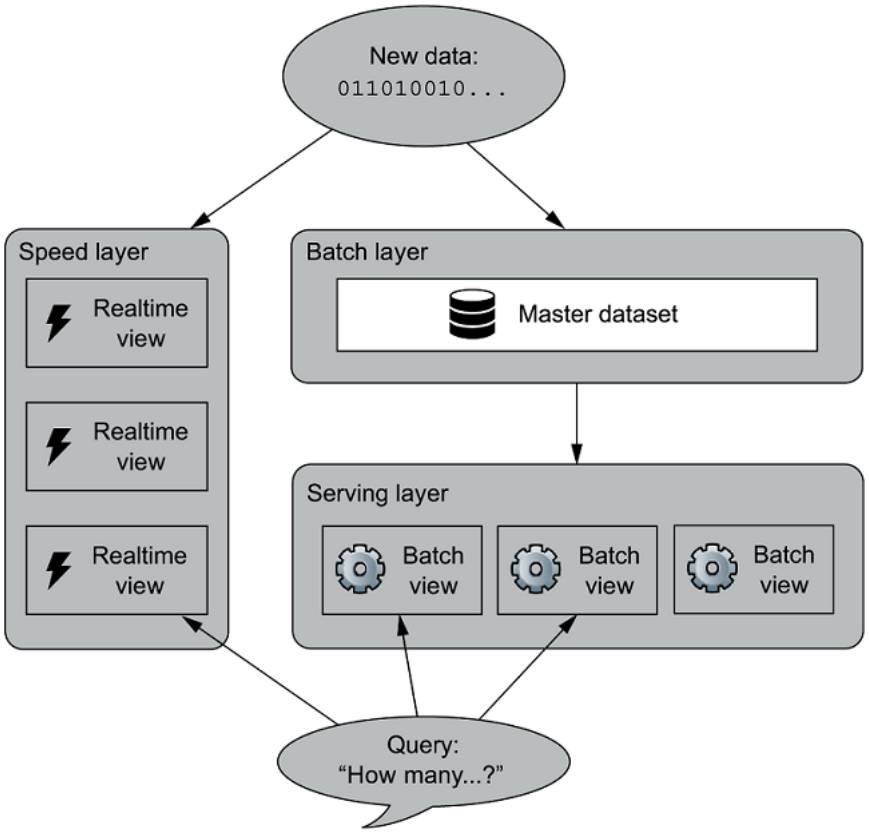


Figure 2.1 Lambda analytics system, showing logical layers of processing based on query latency

To build a Lambda analytics system in Azure, you need to select technologies and services that

provide the functions of the Lambda speed, batch, and serving layers. Let's look at some of the services offered by Azure that can be used.

2.2 Azure cloud services

Microsoft Azure is a cloud services provider. This means Azure provides datacenter services and software that an enterprise traditionally hosted in their offices, in a data center building of their own, or in a hosting providers data center. Information technology resources an enterprise hosts are referred to as "on-premise" resources. This distinguishes them from resources hosted "in the cloud". IT engineers usually have physical access to on-premise resources, but not to cloud resources.

Cloud services providers, like Microsoft Azure and Amazon Web Services, provide three main types of services, classified by the end-user management of the underlying operating system. and software. The lowest level of abstraction provides **Infrastructure as a Service** (IaaS). IaaS services provide resources like virtual machines, routers and firewalls. The provider manages the hardware in their data center, and the end user manages the software and operating system. IaaS resources require technical and developer support, to manage operating system and software installation, and create code to run on the servers. The next level of abstraction provides a **Platform as a Service** (PaaS). PaaS services provide server application hosting such as web servers, databases, and storage. The provider manages the hardware and operating system running in their data center, and manages server applications running on the operating system. The end user configures and uses the applications. PaaS resources require developer support, to create code to run on the server applications. The next level of abstraction provides a **Software as a Service** (SaaS). SaaS services provide user applications delivered over the internet. Typical SaaS applications include web-based email services or web-based file sharing services, that charges a subscription. The SaaS provider manages all aspects of the hardware, operating system, and software. The end user configures and uses the application. Microsoft has transitioned many of their operation systems, desktop and server applications to IaaS, PaaS, or SaaS resources available in Azure.

Microsoft Azure offers both open-source and Microsoft technologies in its cloud services. Azure provides HDinsight for Hadoop engineers and data scientists. HDInsight manages containerized Hadoop processing nodes, with plenty of configuration access and overhead. Azure also provides DataBricks, a SaaS abstraction of the Apache Spark analytics engine. Both provide viable options for operating large analytics systems in the cloud. A third option exists for the Microsoft technologist. By using tight integrations provided by the Azure products, the Microsoft data engineer can build a sophisticated and flexible analytics system using familiar technologies like C#, SQL, and GIT. This book will discuss these services and how to use them to build a complete analytics system in the cloud. Let's look at each of these services.

2.2.1 Event Hubs

Azure Event Hubs provides a PaaS scalable message queuing endpoint, including built-in integrations with Azure Storage and Stream Analytics. Our analytics system will use Event Hubs as the entry point to our data processing pipeline. Using Event Hubs provides our system with a scalable and reliable buffer to handle spikes in the volume of incoming events. Event Hubs accepts both HTTP and Advanced Message Queuing Protocol (AMQP) packets for event messages. There are plenty of clients available for these protocols in your language of choice. These message queues can be read by one or more subscribers.

Events Hubs scales in two ways. First, the endpoint processes incoming messages with a *throughput unit*, a measure of maximum throughput at a fixed cost. Adding more throughput units allows a higher message rate, at a higher cost. Second, Event Hubs partitions the queue. Adding more partitions allows the Event Hub to buffer increased numbers of messages and parallel reads by subscribers. We'll cover setup and design considerations with Event Hubs deeply in Chapter 3.

2.2.2 Stream Analytics

Azure Stream Analytics processes streaming data. Streaming data is ordered by a time element, which is why it's often referred to as events or event data. Stream Analytics accepts streams of data from Event and IoT Hubs and Blob Storage, and outputs processed information to one or more Azure endpoints. It uses a structured query language to query the data stream. The data process can be thought of as fishing a river with a net made of particular shapes. The data flows by the net, and the net captures the matching bits. The fisherman hauls in the net regularly to review his catch. Just so, the queries pull result sets out of the stream as it flows by.

Stream Analytics scales in two ways. First, each Stream Analytics job can utilize one or more *streaming units*, a synthetic metric describing CPU and memory allocation. Each step in the job uses between one and six streaming units. Second, planning parallelism in the stream queries allows Stream Analytics to take advantage of the available parallel processes. For example, writing data to a file in Azure Storage or Data Lake Store can use multiple connections in parallel. Writing data to a SQL Server table uses a single connection, for now. At most, a single query operation can use six streaming units. Each Stream Analytics job can have more than one query operation. Planning the streaming unit allocation along with the query structure will allow for maximum throughput.

2.2.3 Data Lake Store

Azure Data Lake Store stores files. It provides a folder structure interface over an Apache Hadoop file system, which supports petabytes of data. Multiple open-source and native Azure cloud services integrate with Data Lake Store. Fine grained access via integration with Azure Active Directory make securing files a familiar exercise.

2.2.4 Data Lake Analytics

Azure Data Lake Analytics (ADLA) brings scalable batch processing to the Data Lake Store and Blob Storage. ADLA jobs use familiar SQL syntax to read files, query the data, and output results files over data sets of any size. Because ADLA uses a distributed query processor over a distributed file system, batch jobs can be executed over multiple nodes at once. Running a job with parallel processing just takes moving a slider past one.

Azure Data Lake Analytics uses a new coding language called U-SQL. U-SQL is "not ANSI SQL." (Rhys 1) For starters, WHERE clauses use C# syntax. Declarative statements can be extended with C# functions. Query data comes from tables or files. We'll discuss this new language more in Chapter 8.

2.2.5 SQL Data Warehouse

SQL Data Warehouse (SQLDW) bears superficial resemblance to a standard SQL Server database, especially Azure SQL databases. Most functionality matches: CRUD actions, views, stored procedures. There are minor changes to table creation, indexing, and partitioning. The naive user could create a table, insert some data, and use the Data Warehouse like SQL Server databases.

Harnessing the power of SQL Data Warehouse comes from understanding the distributed nature of the underlying technology. Data resides in 60 shards, managed automatically. Queries are distributed across compute nodes, from 1 to 60 as you have configured. Data imports from multiple files are spread across available compute nodes. The user controls scaling compute capacity, while storage relies on Azure Storage for scaling and redundancy.

2.2.6 Data Factory

Azure Data Factory automates the data movement between layers. With it, you can schedule an ADLA batch job for creating aggregate data files. You can import those files into SQLDW, and execute stored procedures too. Data Factory connects to many different endpoints for input and output, and can build structured workflows for moving data between them. Data Factory operationalizes these repeated activities.

2.2.7 Azure Powershell

Azure offers Cloud Shell as an option for managing resources in Azure via the command line. You can access Cloud Shell from the Azure portal, or by connecting to shell.azure.com. With Azure Cloud Shell, you can run Powershell or Bash commands to manage your Azure resources. This book uses Azure Powershell scripts throughout for provisioning Azure resources.

2.3 Azure analytics system architecture

Imagine your company wants to analyze user behavior in their main website to provide relevant suggestions for further reading, to promote user retention and higher page views. A solution would allow generating suggestions based on historical data, recent personalized actions, and machine learning algorithms. Further, the same system could also analyze error events in real-time and provide alerts. You can build a system in Azure which can do the analysis work. The rest of this book walks through use cases, technical tradeoffs, and design considerations when creating and operating each piece of a proposed analytics system which fulfills these functions. Before we dive deeply into each of the services, let's take a look at the system as a whole. This architectural design uses the six Azure services discussed in this chapter.

- Events Hubs for real-time ingestion
- Stream Analytics for real-time query processing
- Data Lake Store for data retention and batch query processing support
- Data Lake Analytics for batch query processing
- Data Factory for batch scheduling and aggregate data movement
- SQL Data Warehouse for interactive queries

2.4 Walkthrough of processing a series of event data records

Figure 2.2 shows how all six services can be assembled into an analytics processing system to monitor error rates and provide "users also viewed" suggestions. In this system, incoming event data follows both a hot and cold path into the user query engine. To illustrate how the event data flows through both paths, lets trace the flow of a typical user action event through both paths. We can see how each path fulfills part of our imagined business requirements for this system.

2.4.1 Hot Path

1. New data events are submitted to the Event Hub endpoint. The data events contain both error data and page view data.
2. Event Hubs endpoint queues the events for retrieval.
3. A Stream Analytics job read events from Event Hubs as they are submitted.
4. The Stream Analytics job runs two steps.
    a. The first step prepares real-time updates for a Power BI dashboard.
    b. This could be a simple error rate query from the last hour, or a Machine Learning API call.
    c. The second saves data directly to SQL Data Warehouse.
    d. To match the project requirements, the query reads the last 24 hours of streaming data and finds the top 3 most visited content pages by same-session users.
    e. The job step then writes this data to the SQL Data Warehouse.
5. The web site reads the most recent list of recommendations for a particular page view from the SQL Data Warehouse.

2.4.2 Cold Path

1. New data events are submitted to the Event Hub endpoint. The data events contain both error data and page view data.
2. The Event Hub endpoint queues the events for retrieval.
3. A Stream Analytics job read events from Event Hubs as they are submitted.
4. The Stream Analytics job runs one step. This step writes the raw data events to Data Lake Store files.
5. Each day, Data Factory runs a scheduled pipeline. This pipeline has multiple steps.
6. Data Factory submits a Data Lake Analytics job to calculate next content page visit probabilities from previous content page visits.
    a. To match the project requirements, the job query reads the last 30 days of stored page visit data files and finds visited content pages by same-session users.
    b. The job step then writes this data to an aggregate file.
7. Data Factory runs a copy step to import the aggregate file into the SQL Data Warehouse.
    a. Run a copy step to move 24 hour "hot" data to an archive table.
8. The web site reads the most recent list of recommendations for a particular page view from the SQL Data Warehouse.

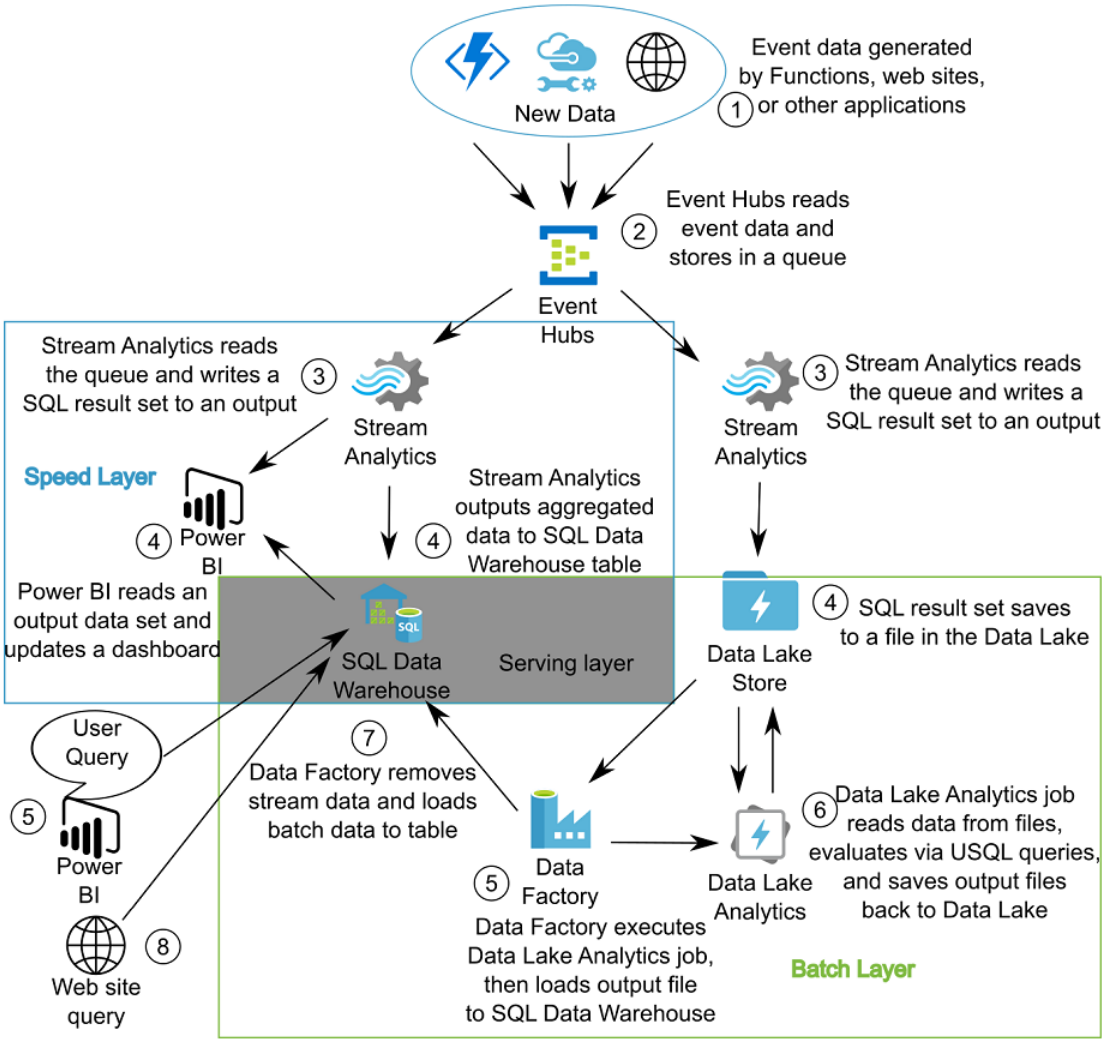


Figure 2.2 Azure abstract service analytics system, showing Lambda architecture layers over hot path and cold path data processing steps

2.4.3 Choosing abstract Azure services

First, the system must collect event data. This can be user input, server actions, or error logging. If you can build new logging functionality into your systems, using Event Hubs as a logging endpoint makes sense because:

- Event Hubs can handle high traffic loads.
- Event Hubs can easily save all event data to disk.
- Event Hubs comes with a good service level agreement.
- Event Hubs is a convenient gateway to Stream Analytics, for real-time processing of the event data.

For applications built on the .NET framework, a NuGet package from Microsoft provides a client for integration into application code for logging events. The .NET client includes built-in retry logic.

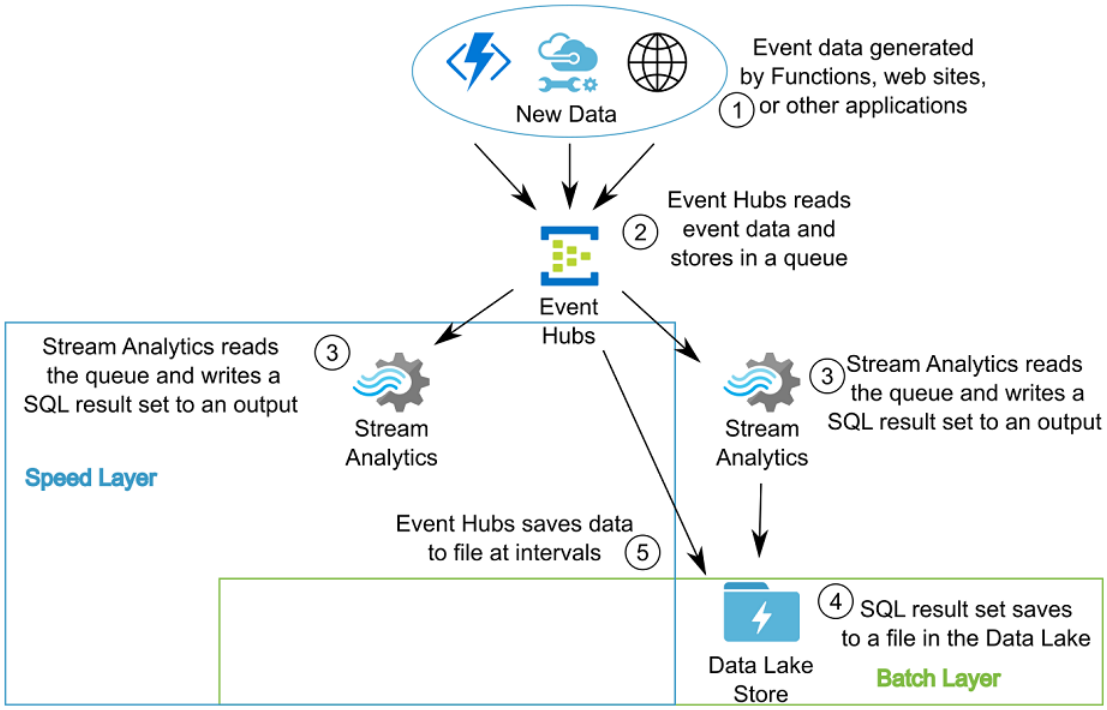


Figure 2.3 Event Hubs read discrete messages and serves the set to readers

Second, the system must process events in real-time. Stream Analytics queries a streaming data set from Event Hubs in real-time.

- Stream Analytics includes built-in support for reading from Event Hub queues as a streaming source.
- Stream Analytics includes built-in support for outputting results to SQL and Cosmos databases, Azure Storage, Power BI, and Azure Functions.
- Stream Analytics has multiple methods for enhancing data with Machine Learning algorithms.
- Stream Analytics lets us leverage our existing SQL skills.

The result set will be emitted in small batches when there's a calculation completed. Power BI integration makes adding results to dashboards quick work. Writing the emitted result set to a SQL database keeps queries up to date, continuously adding new data as it enters the system. For projects where real-time logging is not an option, log files can be loaded into a Blob Storage account for near real-time processing by Stream Analytics.

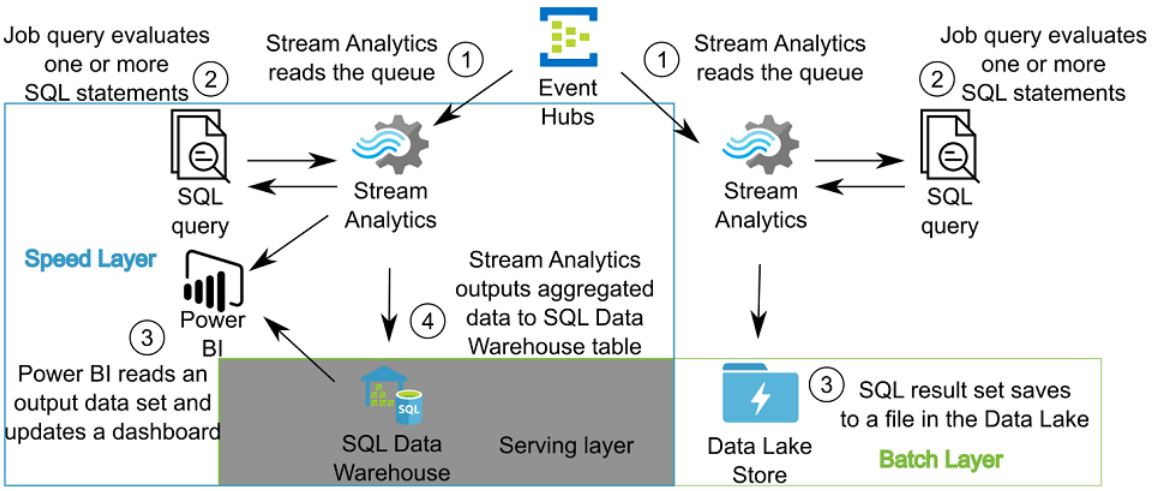


Figure 2.4 Stream analytics reads streams of data, evaluates SQL queries, and emits data sets

Third, the system must store all the data securely while allowing easy access. Both Storage accounts and Data Lake Store can fulfill this role. Data Lake Store has features which Storage accounts lack, making it a better candidate.

- Data Lake Store has no practical limit to the volume of data that can be stored.
- Data Lake Store uses a familiar read/write/execute security model over a hierarchical folder structure.
- Authentication is controlled through Azure Active Directory accounts.

Data Lake storage, which is based on Hadoop File System, allows access by many analytics services, including Data lake Analytics, HDInsight, and Databricks. For projects where real-time logging is not an option, log files can be loaded into Data Lake Store for batch processing by Data Lake Analytics.

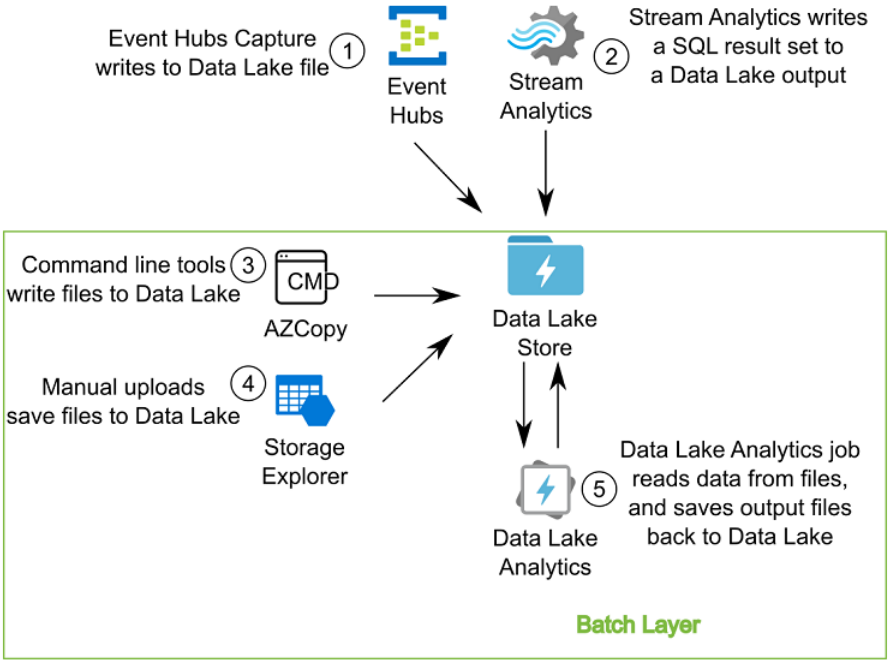


Figure 2.5 Multiple services can access Data Lake storage for read and write operations

Fourth, the system must enable batch analytics over any and all of the data. Data Lake Analytics, as the name suggests, runs analytics jobs over data lake files. The Data Lake Analytics jobs execute using distributed compute by design, with a simple slider defining the number of nodes to use. Engineers and data scientists write the data processing job in SQL. Data Lake Analytics job processing nodes read one to many data files and output one to many data files.

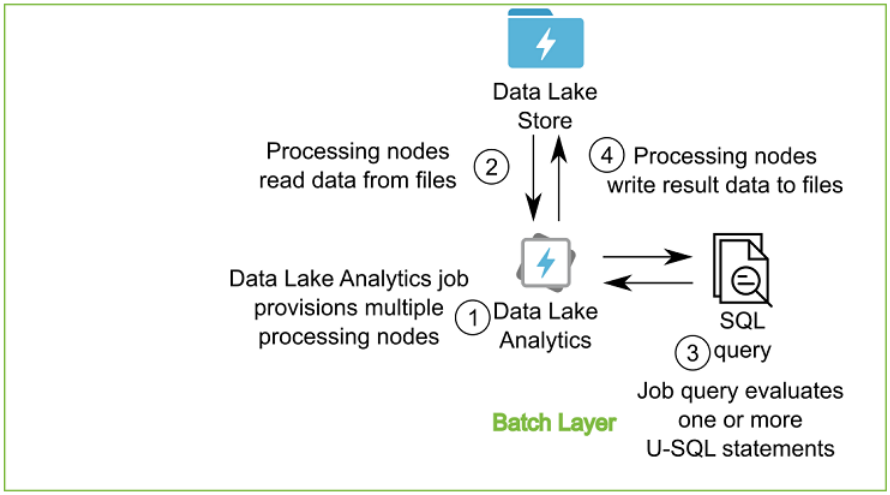


Figure 2.6 Data Lake Analytics reads data files, evaluates SQL queries, and save result sets as files

Fifth, the system must be capable of updating user query result sets. Data Factory is a great fit for automating these other services.

- Data Factory connects to Azure endpoints like Storage accounts, Data Lake stores, and databases.

- Data Factory provides multiple methods for moving and transforming data between endpoints.
- Data Factory can access on-premises files and databases, through an on-premises Integration Runtime.
- Data Factory pipelines can copy file data to and from databases, execute ADLA jobs, and run stored procedures.

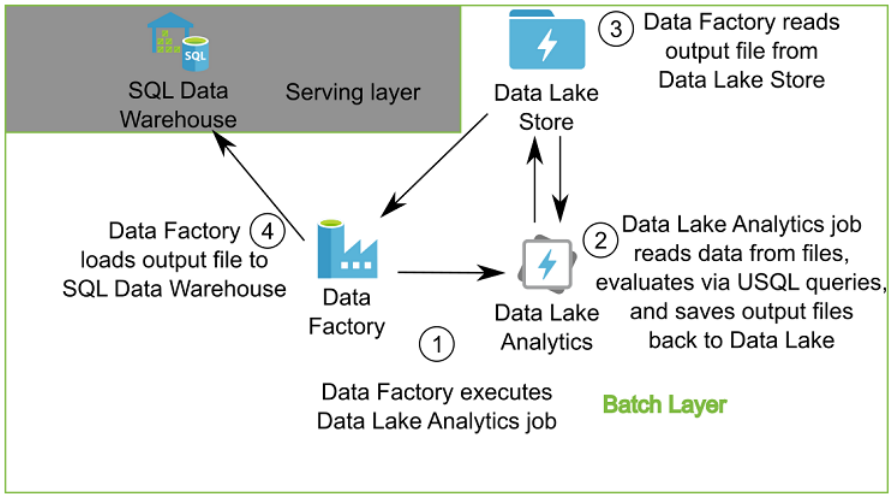


Figure 2.7 Data Factory schedules jobs and moves data between files and databases

Data Factory orchestration completes the end-to-end data movement requirements of this analytics system.

Sixth, the system must be capable of returning results of user queries in a near real-time manner. SQL Data Warehouse uses a familiar SQL Server RDBMS query engine with a distributed storage system which supports 100's of TB of data. Because SQL Data Warehouse uses SQL for querying data, many tools exist which can connect and submit queries. Power BI is one such tool.

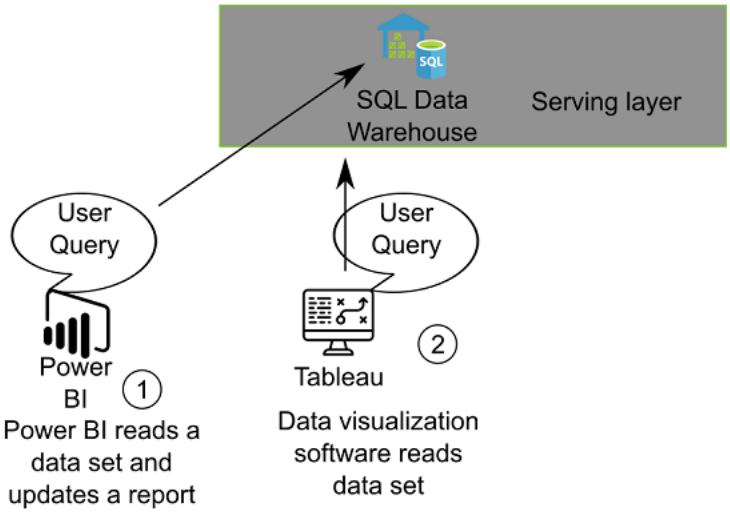


Figure 2.8 SQL Data Warehouse provides data query endpoint

This example focuses on connecting these abstract services together into a working system. Each service offers an optimized tool designed for easy setup, high throughput, and minimal maintenance. The services take full advantage of the consumption model of cloud computing. Let's look more closely at the main cost considerations for building a system with these services.

2.5 Calculating cloud hosting costs

When building systems in any cloud provider, an engineer must include a detailed estimation of costs of the new system. The engineer must understand the long-term costs of the new system, including service, storage, and usage costs. Poor setup choices, lax account security, and lack of planning for resource cleanup can ruin a successful project with unplanned expensive costs.

Microsoft Azure includes a powerful tool for estimating costs. The pricing calculator (azure.microsoft.com/en-us/pricing/calculator/) provides a graphical interface for collecting all the resources required for a project, and generating estimates for cost based on usage. There's little substitute for practice with Azure resources for learning to judge the estimates from the Pricing calculator. Business decisions involve risk and cost calculations, and the output from the tool can validate a decision to use a cloud architecture over an on-premises installation.

Azure resource pricing changes as new systems are added, more capacity comes online, and new versions of existing services are released. These changes tend to lower existing costs or raise the free threshold for usage of services. Your agreement with Microsoft for software, services, and support also modify the final amount billed. Some services, like Data Lake Store, can take advantage of pre-purchased capacity at reduced rates. A knowledgeable data engineer can make a huge impact on TCO for their company in this type of scenario. Reviewing the Azure documentation and subscribing to the Azure newsletter helps keep the engineer current with changes to pricing and capabilities.

2.5.1 Event Hubs

Event Hubs pricing include tiering, scaling, and usage costs. Event Hubs Dedicated includes everything for very high throughput for a very large price per hour. Event Hubs Basic does not include several important features, like multiple consumer groups and a direct file capture, but costs about 50% less than the Standard tier. See Event Hubs pricing azure.microsoft.com/pricing/details/event-hubs/ for specific prices. Most scenarios will use the Standard tier and include file capture for an additional cost. The primary cost decision for the data engineer configuring a new Event Hub involves choosing a base number of throughput units. Generally two throughput units are used at a minimum, but the volume of data ingress should be calculated to minimize throttling of any application message submission.

2.5.2 Stream Analytics

The Azure Stream Analytics cost structure consists of streaming unit usage per hour. Typical scenarios involve one to six streaming units, depending on the complexity of the queries. Each job has a fixed base cost per month, but each job can have multiple inputs and outputs. Most systems need only a single job per event hub. See Stream Analytics pricing azure.microsoft.com/pricing/details/stream-analytics/ for specific prices. We'll cover capacity planning for Stream Analytics in chapter 4.

2.5.3 Data Lake Storage

Azure Data Lake Storage (ADLS) pricing covers three activities: space utilization, read and write transactions, and data transfers. Monthly average file storage feeds the storage calculation. Read and write transactions are inexpensive. Outbound data transfers incur charges; Inbound transfers are free. Prepurchasing storage space provides a discount. See Azure Data Lake Storage pricing azure.microsoft.com/pricing/details/data-lake-storage-gen1/ for specific prices. ADLS gen 2 adds tiered storage to the mix, with modified costs structures for space utilization and transactions. The design of the analytics system should take into account these differences in long-term costs of file storage. See Azure Data Lake Storage Gen2 pricing azure.microsoft.com/pricing/details/storage/data-lake/ for specific prices.

For most scenarios, storage follows compute in terms of cost. Balancing tiered access cost saving with latency requirements adds to the design complexity. However, hot and cool tiers have the same latency. Most data should be shifted to the cool tier within a month. ADLS gen 2 reduces costs significantly. Because of the cost savings, most projects will want to use ADLS gen 2, or plan a migration from gen 1 to gen 2.

2.5.4 Data Lake Analytics

Azure Data Lake Analytics (ADLA) cost structure consists of analytics unit usage per hour. Prepurchasing unit-hours provides a discount. See Data Lake Analytics pricing azure.microsoft.com/pricing/details/data-lake-analytics/ for specific prices.

Any ADLA job will run with a single analytics unit at 100% efficiency, but most jobs can take advantage of parallel processing to shorten the duration of the job. The data engineer can help determine the optimum allocated units for a particular job. The engineer can also plan for monthly usage and assist in prepurchasing unit-hours.

For example, suppose a job imports two files, aggregates the data, and writes a summary file.

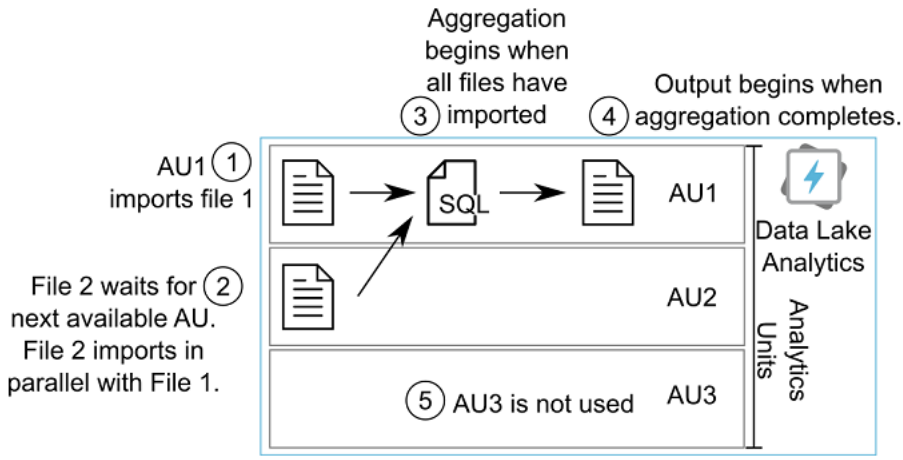


Figure 2.9 Parallel processing in Azure Data Lake Anaytics jobs with multiple analytics units

This job can run with 1, 2 or more analytics units doing the processing.

1. Using 1 unit, both files will import sequentially, the aggregation step will execute, and finally the summary file will be written.
2. Using 2 units, both files will import in parallel, then the aggregation step will execute, then the summary file will be written.
3. Using 3 or more units will run just like using 2 units. Any extra units beyond 2 will incur charges for the duration of the job without doing any work.

Each analytics unit allocated to the job incurs charges for the duration of the job. In the best case, a job that takes 4 hours with one AU would take 2 hours with 2 AU and 1 hour with 4 AU. All three runs would cost the same: 4hr*1AU = 2hr*2AU = 1hr*4AU = 4AUhrs. In reality, jobs are not entirely parallel, and analytic units will be idle for some portion of the job. The optimization work determines how much parallelization can be achieved for which portion of the job, and then balances the requirements for total job duration with minimizing cost. We'll look more closely at optimizing ADLA jobs and costs in chapter 7.

2.5.5 SQL Data Warehouse

The parallel compute design of the SQL Data Warehouse (DW) separates the query processing from storage, and this separation is reflected in the pricing structure. DW bills compute time on an hourly basis, based on the selected compute level. DW bills storage by the hour, based on the size of the data stored.

The release of gen 2 of DW adds an adaptive cache layer, but also raises the minimum compute service level. Both versions of DW allow on demand scale of compute level both up and down. This feature makes calculating costs more complicated and increases the risk of unexpected expenses. For example, during loading of data to the DW increasing the compute level will reduce the data load duration. If this scaling happens as a manual process, the user must remember to return the compute service level to the previous level, or unneccesary expenses will

occur. It's a good practice to create alerts around DW service level scaling once the typical service level is determined.

2.5.6 Data Factory

There are two versions of Data Factory available. Data Factory gen 1 is a configuration-driven service using JSON files to define the various endpoints, data movements, and transforms in each job. Data Factory gen 2 builds on gen 1, providing a GUI for building the JSON configurations. Calculating the ongoing costs of Data Factory use involves several variables. These variables fall into two main categories: data operations and orchestration. Data operations charge for the duration of copy events between source and sink. A synthetic usage metric called a Data movement unit (DMU) measures the duration of data activities and meters the movement rate. DMU usage is billed per hour. For gen 2, Azure adds charges for both read/writes and monitoring operations against entities like a dataset or list of pipeline activities.

Jobs which move data between cloud resources are more expensive than moving from on-premises to cloud. Orchestration charges cover executions of the Integration Runtime both in the cloud and on-premises. Orchestration charges are lower for executing more jobs in the cloud, or executing longer running jobs on-premises.

To formulate a recommendation, the data engineer needs to plan for the most cost effective scenario which accomplishes the requirements of the project. For example, 10 jobs which run for 10 seconds each minute would be more expensive than a single job which runs 4 hours each day. The data operations counts would be higher, and the number of job runs would be larger.

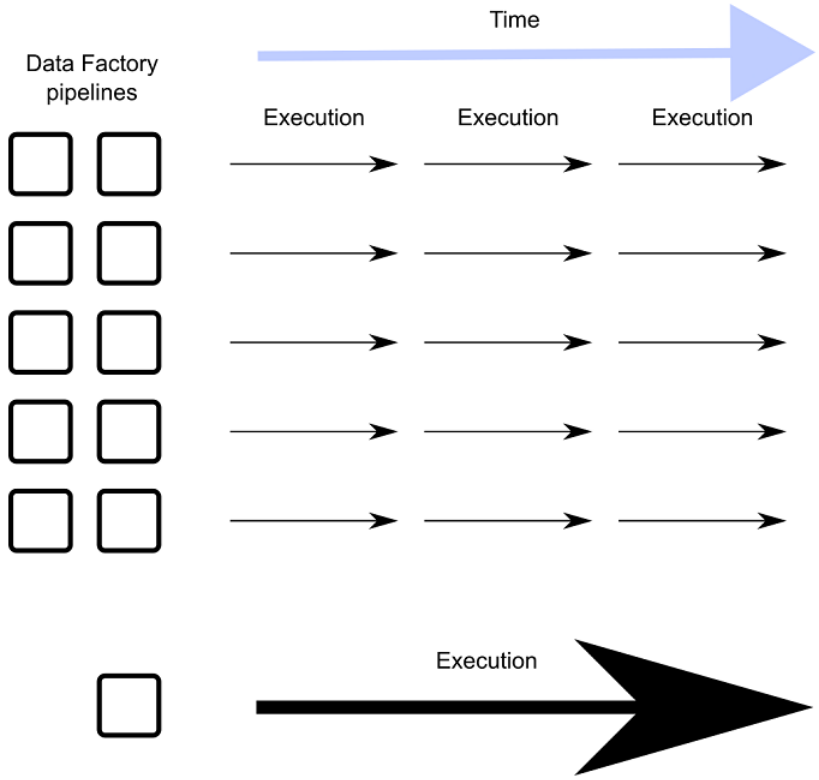


Figure 2.10 Multiple repeated Data Factory jobs vs single job

SIDEBAR **Data Factory Setup**

We'll look more closely at Data Factory setup and configuration in chapter 9. Setting up the self-hosted Integration Runtime (IR) will clarify the differences between cloud and on-premises IR functions.

2.6 Summary

- Using highly abstract cloud services reduces setup and maintenance time, but limits options.
- Microsoft Azure offers several well integrated services that can combine to form an analytics processing system.
- Calculating costs requires estimating volume and size of the data collection.

3 Azure Storage Blob service

This chapter covers:

- Naming Azure services
- Choosing a storage service
- Setting up an Azure Storage Blob service
- Configuring file access in a Blob service

In the previous chapter you explored a set of services provided by Microsoft Azure: Event Hubs, Stream Analytics, Data Lake, Data Lake Analytics, and SQL Data Warehouse. You saw at a high level how these services can work together to create an analytics system. This chapter begins showing you how to design and set up these services to lay the foundation of an analytics system. You'll learn how to implement a storage system that is secure and scalable. The storage system will be an integral part of the system you build in Azure.

3.1 Azure naming conventions

Every Azure service, also called a resource, must have a name. Consistently applying a considered naming convention helps users find services, and identify ownership and usage of services. From a resource group to a SQL database to Azure Storage Blobs service, you will be browsing and searching for the specific resource you need to work with.

All Azure services will have a Subscription, a Resource Group, a name, and a location.

- A subscription groups services together for access control and billing.
- A resource group groups related services together for management.
- A location groups services into a regional data center.
- Names are globally unique identifiers within the specific service.

TIP	Because caching exists in many levels of Azure infrastructure, and syncing changes occurs between regions, recreating a service with the same name can be problematic in a short timeframe, on the order of minutes.

3.1.1 Resource group

Resource groups in Azure are organizing containers. Every Azure service has one. Every resource group has a region. The resource group anchors a service to a region, with the primary configuration data for the service stored in that region. This is especially true for some services, like Cosmos DB and Traffic Manager, which are global services and have infrastructure in every region. Deleting a resource group deletes all the services attached to it. This book uses "ade-dev-eastus2" as the resource group for any Powershell (PS) scripts which require it. This book uses the "East US 2" region for any PS scripts which require it, since all resources in the book are available in the region.

Listing 3.1 New resource group

```
New-AzResourceGroup -Name "ade-dev-eastus2" -Location "East US 2"
```

Execute this line in PS with the Azure Modules loaded. This PS script will return an error if a group by that name exists. Otherwise it will create a new resource group. There are many regions, or locations, for hosting Azure resources across the globe, including the Americas, Europe, Asia Pacific, and the Middle East and Africa. You can see the current list of services by region at Microsoft's Azure website azure.microsoft.com/global-infrastructure/services/.

You should create a Resource Group for your services before creating any services. Use this first step to plan your naming conventions and region affinities at the start of your project. A resource naming convention should be applicable across all resources types and follow these guidelines:

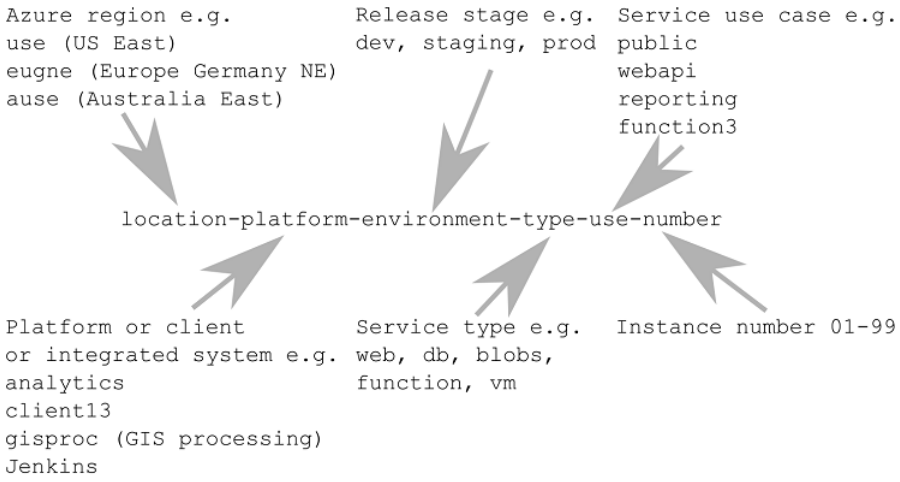
- It should align with security or management boundaries.
- It should decrease the cognitive load for the user in identifying a resource.
- It should produce a name which is globally unique within the service.
- In the best case, a naming convention will allow meaningful sorting by name.

An approach moving from broadest to narrowest classifications can fulfill these requirements.

1. The broadest element is the Azure region or location. Most services will have a definite location. Define a set of acronyms to use. Some services, like Azure Storage, have a length limit for the name.
2. Define a platform name. What project, product, or client is the collection of services supporting?
3. Separate services by release promotion. Continuous deployment, to a development or production environment, is one of several release strategies that rely on deployment to more than one environment for testing and validation. Dev, Stag, Test, Load, and Prod all

work for defining release environments.

4. Adding a service type descriptor further groups like services.
5. A use case descriptor is more expressive of the function of a particular instance of a service than a number. For example, a system may use multiple Azure databases such as Content, Users, and Logs, each with distinct usage periods and patterns. Adding use descriptors to the database names makes their purpose clear.
6. All other options being equal, a fixed-width numeric ID is a final differentiator. A random string of alphanumeric characters also serves as a valid ID for automatically provisioned services.



As an example, picture a design for an analytics system hosted in the Australia Southeast region.

- A resource group name of "ause-analytics-dev-grp"
- An App service hosting a Web API site named "ause-analytics-dev-web-api-01"
- Two databases named "ause-analytics-dev-db-raw-01" and "ause-analytics-dev-db-curated-01"
- An Azure Storage account named "auseanalyticsdevblob"

You'll see the specific requirements for setting up Azure Storage later in this chapter, including the naming restrictions. Other resource creation throughout the book will present similar details. Use the components of the naming convention that make sense for your situation. Your services may all be in a single location, and not need location in the name. Your projects can be separated by subscription, or all your services can be a single type. If you plan for expanded use of Azure services and features at the start, your naming convention will be flexible enough to cover your system.

NOTE For a different take on service names, Microsoft recommends some formats at docs.microsoft.com/azure/architecture/best-practices/naming-conventions.

3.2 Searching for services

In the Azure Portal, common filters include name, subscription, resource group, service type, location, and tags. Tags are key-value pairs you can add to any service for filtering. Azure provides multiple methods for filtering by type. NOTE: Remember, in the Azure portal the content layout containers are called "blades."

Using the Azure portal, you can search in the "All services" blade, use the type filter in the "All resources" blade, or navigate directly to a specific service type blade with the "Favorites" left navigation. Filtering by name, location, and tags is available from the "All resources" blade, or from a specific service type's blade. You can even use Azure Powershell to get services by type. Access Azure Powershell by visiting Azure Cloud Shell at shell.azure.com/, or clicking the ">_" header menu in the Azure portal. See Appendix A for more details about setting up Powershell, and using Powershell to create and configure services in Azure. Listing 1 shows the Azure Powershell module command for getting a list of Azure Storage accounts.

Listing 3.2 Azure Powershell list Storage services

```
Get-AzResource -ResourceType Microsoft.Storage/storageAccounts | ft
```

Listing 1 includes the Format-Table alias ft, which formats the output of the command as a table instead of a column of property values. Listing 2 shows the output of the Get-AzResource command. The subscription contains a single Azure Storage account.

Listing 3.3 Azure Powershell list Storage services output

```
Name         ResourceGroupName ResourceType                    Location
----         ----------------- ------------                    --------
adedeveastus ade-dev-eastus    Microsoft.Storage/storageAccounts eastus
```

From Listing 2, you can discern the naming convention in use is platform-environment-location. Now that you can locate services in Azure, let's take a look at Azure Storage.

3.3 Cloud storage services

Azure Storage is Microsoft's first cloud storage service. Azure Storage provides multiple specialized types, of essentially key/value stores: queues, blobs, tables, file shares. Key/value pairs store data in a flat hierarchy. The key is used for lookup, and the value stores simple or complex data. Blob storage and file shares just have a particularly large value, and a key that looks like a file path. Tables are standard key/value stores. Queues store ID and text fields. Azure Storage also provides backing for virtual machine (VM) disks in Azure VMs.

- Blobs service stores files as byte collection blobs, with a programmatic command interface.
- Files service stores files, with a Server Message Block (SMB) protocol interface. Use

Files services as network shares.
- Queues service stores messages in a queue for sequential programmatic retrieval.
- Tables service stores collections of key/value pairs, with unique IDs.

Now you'll compare the services by solving a common use case, long-term file storage.

3.3.1 Problem definition: backup IIS logs

In this scenario, the IT department wants to copy all IIS logs from 10 web servers to cheap long-term storage ahead of deletion. The log files will be rarely accessed, and lowest cost is important. You need to set up a service in Azure and copy the log files.

To do that, you need to evaluate your options for file storage in Azure. Once you examine the services, you'll be able to choose one that fits the project needs. You can create the service in Azure and evaluate them before committing to copying.

3.3.2 Create an Azure Storage account

Azure offers multiple methods for creating new resources, including Azure Powershell, Resource Manager (ARM) templates, and the Azure Portal. While Powershell and ARM templates allow you to more easily automate your activities in Azure, the Azure Portal presents a lower bar to entry. This chapter uses both examples from the Azure Portal and Powershell scripts to demonstrate procedures and features in Azure.

> **NOTE** You should already be familiar with the Azure Portal and have a subscription. If not, please visit azure.microsoft.com/en-us/free/. You will need an email address and a credit card to create a subscription.

AZURE STORAGE CREATION STEPS

Setting up an Azure Storage account requires a few pieces of information common to all Azure services, including a subscription, name, location, and resource group.

- A subscription groups services together for access control and billing.
- A resource group groups related services together for management.
- A location groups services into a regional data center.
- Names are globally unique identifiers within the specific service.

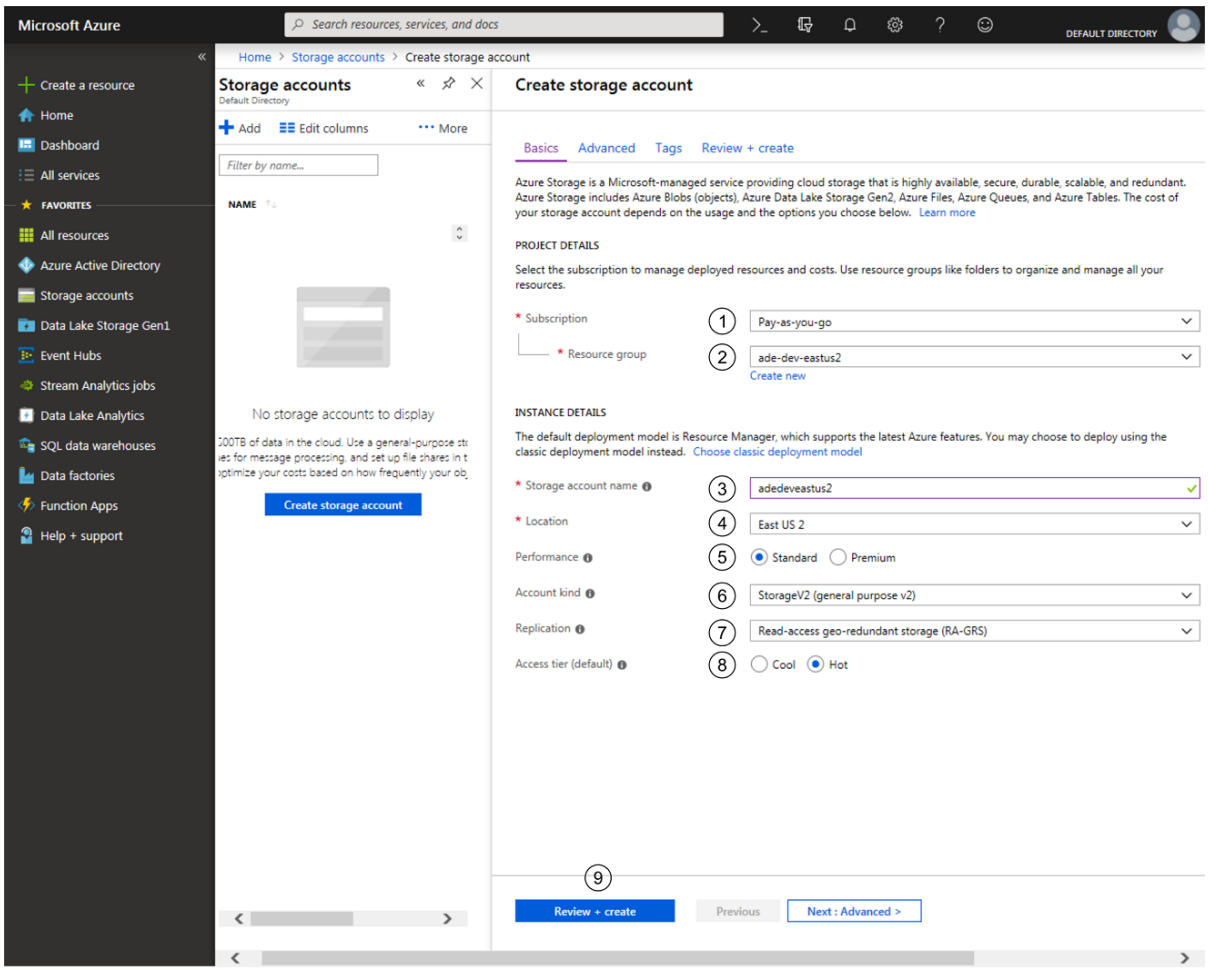
It also requires some options specific to Azure Storage.

- A performance tier selection allows choosing solid state drives when configuring Azure Storage for VM disks.
- Account kind selection provides backward compatibility for existing implementations of Blob storage or general purpose Storage services.
- Azure Storage account contents can be replicated for increased redundancy. Options are

available for single data center redundancy (LRS), multi-data center redundancy (ZRS), and multi-region, multi-data center redundancy (GRS) with read access (RA-GRS). See Azure Storage replication later in this section for more details.

Use the following steps to create a new Storage account with these options.

1. In the Azure Portal, use the "Create a resource" menu or use the "Add" button on the Storage accounts blade to open a new "Create storage account" blade. You can also browse directly to portal.azure.com/#create/Microsoft.StorageAccount-ARM.
2. Choose a subscription. The default will be the oldest subscription, if you have access to more than one.
3. Choose a resource group. (See Appendix A for instructions if you haven't created one.)
4. Choose a name ("[XYZ]deveastus2"). The Storage account name must be lowercase alphanumeric, between 3-24 characters, and globally unique.
5. Choose a location. Azure Storage are available in all regions; choose one close to you. Keeping Azure resources which interact in the same region minimizes network latency. You may choose a region to match your user base, as some governments restrict movement of data outside their zone of control.
6. Choose the default Standard performance level. Premium is only for VM disks.
7. Choose the default kind StorageV2. This is the latest version, and there are no benefits to using the previous version for new projects.
8. Choose the LRS replication type, or leave the default RA-GRS. (LRS costs less than RA-GRS.)
9. Choose the Hot access tier. This minimizes costs for your tutorial usage. The Cool tier and Archive tier include a minimum storage duration cost for each file. Access tiering allows you to balance content retrieval latency with cost.
10. Review and Create the Storage account.



1. Use the same subscription for related resources.
2. Use the same resource group for related resources.
3. Name must be alphanumeric only.
4. Keep related resources in the same location for best performance.
5. Premium is only for VM disks.
6. Leave the default StorageV2.
7. Redundant storage incurs higher costs.
8. Choose Cool is you will rarely access these files.
9. Review, and then Create the Storage account in two steps.

Figure 3.1 Creating a Storage account

Execute the single line of code Create new Storage account with powershell in Azure Powershell. This PS script will return an error if a Storage account by that name exists.

Listing 3.4 Create a Storage account with powershell

```
New-AzStorageAccount -ResourceGroupName "ade-dev-eastus2" -AccountName "adedeveastus2" `   ❶
-SkuName Standard_RAGRS -Location "East US 2" `   ❷
-EnableHttpsTrafficOnly 1 -Kind "StorageV2"   ❸
```

❶ Account name must be alphanumeric

❷ Choose RA-GRS for maximum redundancy, LRS for minimal redundancy

❸ Allowing only HTTPS traffic increases security

BLOB TIERING

Azure Storage services should be configured for each use case. Access tiering in Blob services allows end users to balance content retrieval latency with cost. For example, cheap long-term archival of files is a common use case. Which tier would suit this use case? A hot tier provides the fastest retrieval; a warm/cool tier slower retrieval, or a different cost structure. A cold tier would be the least expensive for long term storage, but may have long retrieval times and additional retrieval fees.

Table 3.1 Azure Storage blob access tiers

Access Tier	Latency	Storage cost	Access cost	Minimum period
Hot	milliseconds	Highest	Lowest	N/A
Cool	milliseconds	Middle	Middle	30 days
Archive	< 15 hours	Lowest	Highest	180 days

Tiering options are available when targeting General purpose v2 or Blob storage types. These types allow changing the blob access level between Hot, Cold, and Archive levels, to set a more favorable cost for your Storage account usage. You can choose between only Hot and Cool tiers as a default for the Azure Storage Blob service. You can only move blobs into the Archive tier at the blob level, not as a default on the Blob service. You can see how to set the access tier later in the chapter, when you create a container (Blob service) in the Azure Storage account.

Frequently accessed files, or files supporting systems with fast performance targets, prioritize low latency over price. For frequently accessed files, or transitory files, you would choose the Hot tier as a default. Some use cases, like archival storage, need to write once and be read rarely. In this case, price would rank higher than retrieval latency, so you would choose the Archive tier. First copy the blobs to a Blob service with a default at the Hot tier, to prevent an initial retention period charge. Then shift the blobs to the Archive tier with your selected tool. Retrieval latency is not the only configurable property for the Azure Storage services.

AZURE STORAGE REPLICATION

A common working space for collaboration would likely need to be highly available. Picking a location near your user's physical location will reduce network latency over the Internet. Azure Storage provide some flexibility in configuration to allow for differing availability requirements. Storage account contents can be replicated to multiple locations.

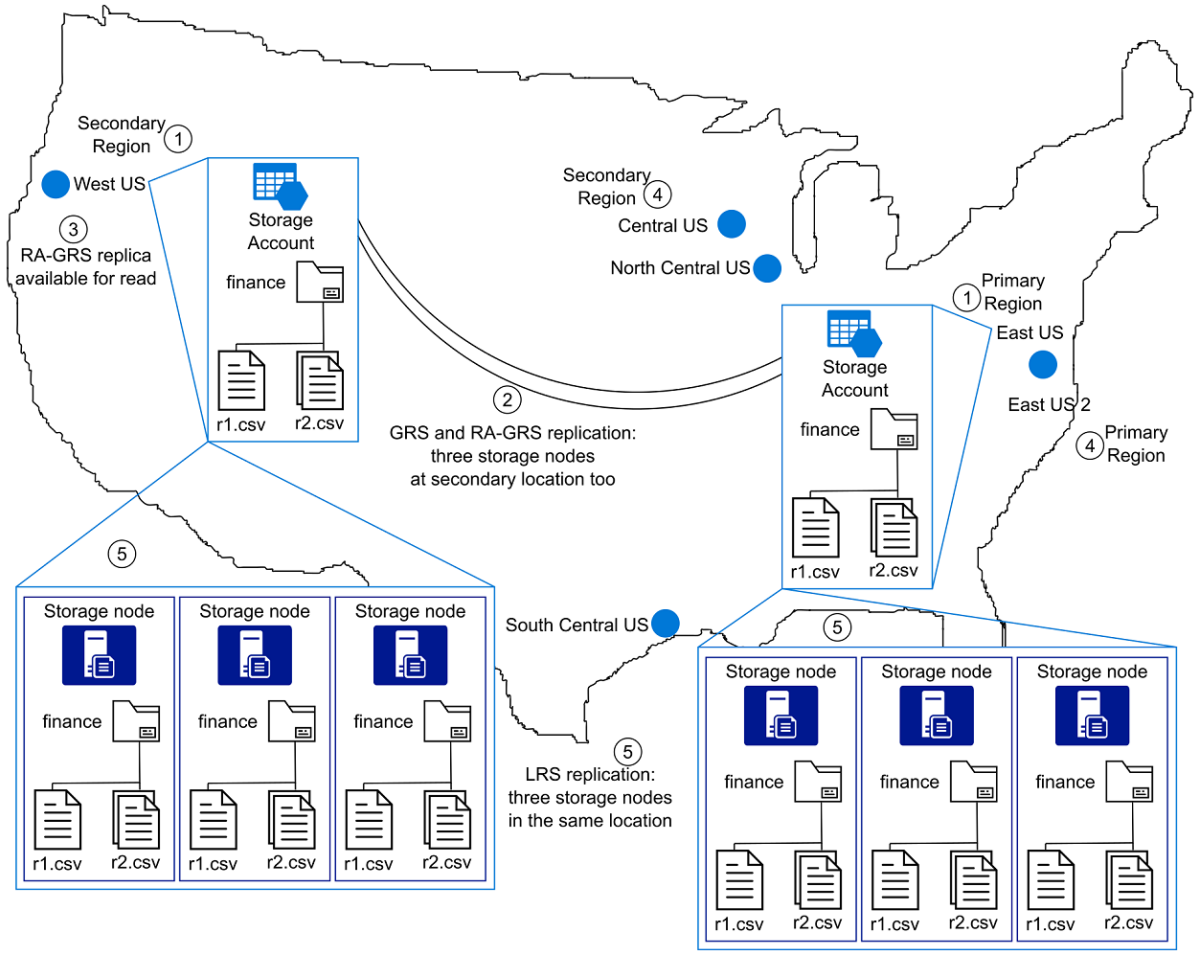


Figure 3.2 Storage account replication to multiple zones

At minimum, for the Local Redundant Storage (LRS) level three copies are kept in the local datacenter. Zone redundant storage (ZRS) replicates contents to another datacenter in the same region. Geo-redundant storage (GRS) replicates contents to a separate region. Read-access geo-redundant storage (RA-GRS) replicates contents to a separate region, and allows read access to the Storage account contents. Adding replication to Azure Storage gives your applications redundancy, and raises their cost and complexity. This redundancy does come at a price. For the latest pricing details, see azure.microsoft.com/pricing/details/storage/blobs/.

Microsoft chooses the region pairing, and when to declare a region unrecoverable and failover to the secondary region.

- Regions are separated physically, with at least 300 miles between regions datacenters.
- Regions are paired geographically, to comply with data residency requirements.
- Paired regions receive updates sequentially, to minimize downtime and effects of adverse outcomes due to updates. See docs.microsoft.com/azure/best-practices-availability-paired-regions for the latest list of paired regions.

For greater control or shortened recovery times, you may wish to backup data to a second

Storage account in a separate region using a tool like AZCopy. You can find an introduction to AZCopy at 3.3.5 later in this chapter.

> **IMPORTANT** Microsoft maintains Azure datacenters in locations around the world. Every Azure service you create is available in at least one location. Some services, like Azure Storage, can run in any Azure location. Not all services are available in all locations. For best performance, keep services which need to connect to each other in the same location.

3.3.3 Selecting a Storage account container

Azure Storage provide cheap cloud storage options, including services targeted at NoSQL data, queues, and VM disks. The Storage account consist of five services: Files, Blobs, Disk, Queues, and Tables. What do each of the services offer?

Table 3.2 Azure storage properties

Type	Feature	Limitation
Files	SMB interface for file shares; high throughput; folder-file hierarchy	TB scale files and shares
Blobs	Global replication; high throughput; flat-file hierarchy	PB scale storage
Disk	high throughput; folder-file hierarchy	attached to VM; no access outside VM
Queues	message store; no hierarchy	messages only
Tables	NoSQL store; collection-item hierarchy	table items only

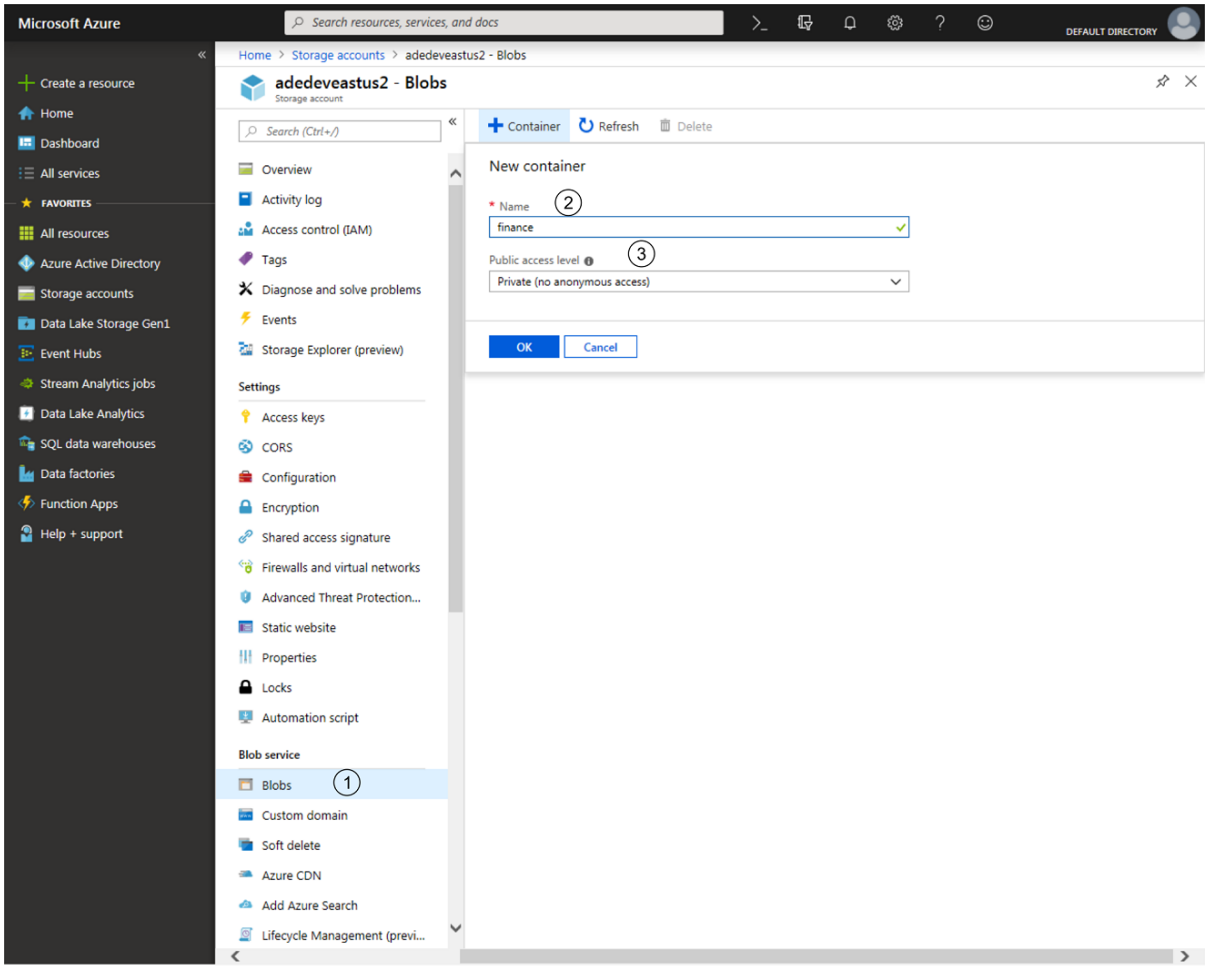
For this scenario, you want to use a service that can store and access large files. Since you want to use this service for file storage, Queue and Table stores are out. For Disk stores, a dedicated VM would be required to access the files. This requirement doesn't match your low cost directive. So you can eliminate the Disk store option. In an analytics system, other native Azure services will access these files, so you don't need or want the features from File or Disk stores. Blob service would be most useful in your systems.

3.3.4 Create a Storage account container

In order to copy the IIS logs from the web servers to the new Storage account, you need to create a container. A container works just how it sounds. It's a logical grouping of a set of objects. Since you want to store text files in this Storage account, you won't be setting up a queue, table, or disk container. You have no requirement for enabling remote file share access to the log files, so you don't need a file service. You'll set up a blob container using the Azure Portal.

1. In the Azure portal, browse to the Azure Storage account blade. One approach is via the "All services" blade, filtering to Storage accounts, and clicking the Storage accounts icon.
2. Click your newly created Azure Storage service "[XYZ]deveastus2".
3. In the Storage account, click **Blobs** under Blobs service to show the Blobs blade.
4. Click the "Container" button to add a new container.

5. Enter container name **iislogs**. This name must be lowercase alphanumeric and hyphens, between 3-63 characters, and unique in the Storage account.
6. Leave the default Public access level "Private". Other access levels allow read or read and list access by anonymous public users to the container.
7. Click **OK** to create the container.



(1) Blobs containers

(2) Use lowercase letters, numbers, and hyphens

(3) Leave this Private.

Figure 3.3 Creating a Blob container

You can now use the Azure portal or other tools to copy the IIS log files into the container.

3.3.5 Copy tools for Blob service

Looking back at the scenario, now that you've created a Storage account, and a blob container for the IISlogs, you can start copying the files. Figure Figure 3.4 shows 4 options for tools to copy files. The Azure Portal provides a web-based interface for uploading and downloading files. AZCopy is a command-line tool from Microsoft for copying files to and from Storage account services, including Blobs and Files. Azure Data Factory (ADF) uses cloud scheduling and on-premises integration to copy data. You can read about Azure Data Factory in chapter 10. Azure Storage Explorer provides a desktop GUI interface for uploading files to multiple Azure services, including Azure Storage.

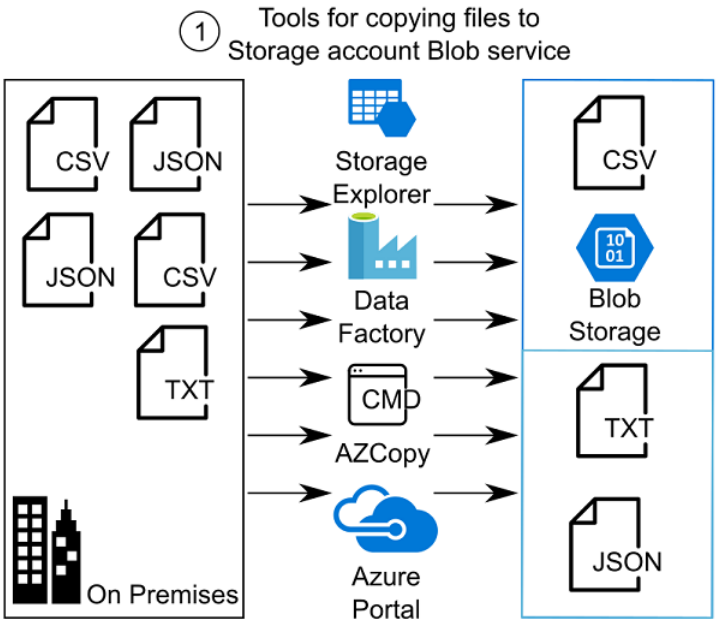
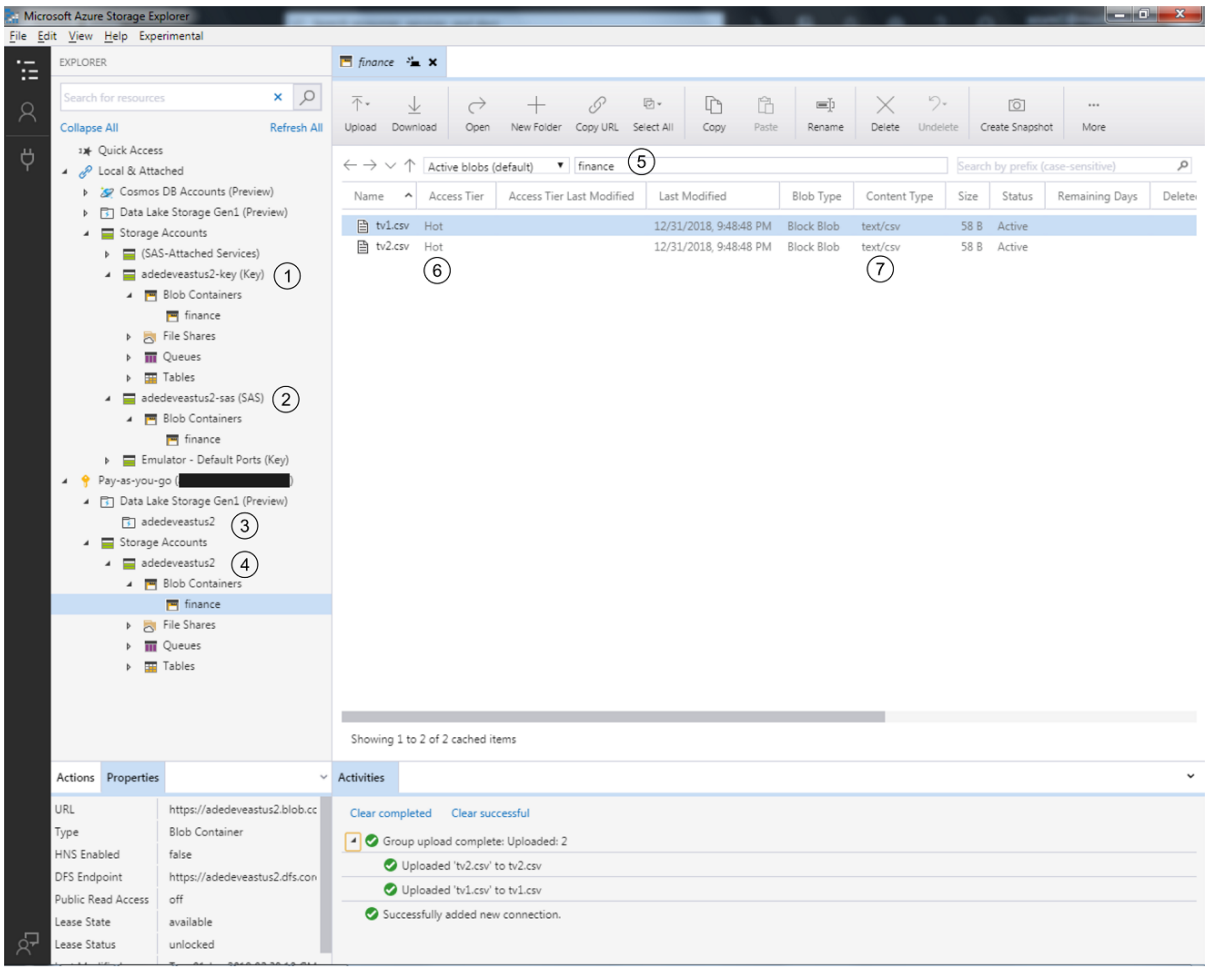


Figure 3.4 Blob service copy tools

Each tool has a strong use case.

- The Azure Portal is available without an install.
- AZCopy can be used for automated file copies without user interaction.
- File copies with Azure Data Factory can be included in multi-step workflows, and integrate with other Azure services.
- Storage Explorer provides an easy to use GUI and status tracking of actions.

Azure Storage Explorer has a flexible authentication process. It can authenticate to Azure Storage using Azure Active Directory (AAD), shared access signature (SAS) tokens, or the Storage keys themselves. You'll see more about access controls for Azure Storage later in this chapter. Figure Figure 3.5 shows Storage Explorer connecting with multiple types of authentication. You can use the drag-and-drop function to upload and download files, or use the individual function buttons in Storage Explorer.



1. Use the account key to connect to a Storage account. All resources and services are accessible.
2. Use a shared access signature to connect to a Storage account. Only resources and services in the SAS are accessible.
3. Connect to Data Lake with your Azure Active Directory (AAD) user.
4. AAD user provides access to Storage account resources and services based on role. Nearly all roles grant read access.
5. Storage Explorer interprets the folder structure of the files in a Storage account Blob service.
6. Use Storage Explorer to change Access Tier: Hot, Cool, and Archive.
7. Store many file types with recognized MIME types.

Figure 3.5 Storage Explorer configured to connect to Azure Storage with access keys and SAS tokens

AZCopy works well for simple copies. Both Storage Explorer and AZCopy allow you to copy files into new folders during the copy process. AZCopy uses wildcard matching to select files for copying.

Listing 3.5 Command line log shipping with AZCopy

```
"C:\Program Files (x86)\Microsoft SDKs\Azure\AzCopy\azcopy"
<linearrow  />/Source:C:\csvLogs\aa /Dest:https://abc.
<linearrow  />blob.core.windows.net/project-abc/v1/v1.1    ❶
<linearrow  />/destkey:==StorageKey== /Pattern:"ch*.csv" /Y   ❷
```

❶ Plan for version of files

❷ /Y switch for suppressing acknowledgments

AZCopy, as a command-line application, can be scheduled to run automatically. AZCopy includes many options for selecting files to include, and configuring the output commands.

- Omitting the "/Pattern" option copies all files in the selected folder.
- Adding "/S" option to the command recursively copies subfolders and their files.
- Blobs are assigned a MIME type of "application/octet-stream" during upload by default. Add the option "/SetContentType" to have AZCopy assign the MIME type of the blob based on the file extension. AZCopy comes with most MIME types preconfigured in the AzCopyConfig.json file, which is located in the AzCopy directory.
- For copies of the same file type, or to override the configured MIME type, add option "/SetContentType:[MIME-type]" to explicitly set the type.
- By default, an operation journal is output at %LocalAppData%\Microsoft\Azure\AzCopy\AzCopy.jnl. This file tracks the progress of the execution, and AZCopy can use it to restart the process if there is a problem. This file is deleted upon successful completion.
- Add the option /V to output a verbose log file at %LocalAppData%\Microsoft\Azure\AzCopy\AzCopy\AzCopyVerbose.log.

Listing 3.6 Setting specific MIME type with AZCopy

```
"C:\Program Files (x86)\Microsoft SDKs\Azure\AzCopy\azcopy"
<linearrow />/Source:C:\csvLogs\aa /Dest:https://abc.
<linearrow />blob.core.windows.net/project-abc/v1/v1.1
<linearrow />/DestKey:==StorageKey== /Pattern:"ch*.csv" /XO /Y   ❶
<linearrow />/SetContentType:text/csv   ❷
```

❶ /XO switch to exclude older files from copy when present in the target folder.

❷ Pattern is copying CSV files, so set MIME type to CSV

See docs.microsoft.com/azure/storage/common/storage-use-azcopy for download instructions. You can read more about file versioning and data drift in chapter 4.

3.3.6 Blob tiering

Use the Azure Portal, Azure Storage Explorer, or the Azure SDK to change blob tier to Hot, Cool, or Archive. Moving a blob from the Cold or Archive tiers to another tier incurs a prorated charge, if the minimum storage period has not been met. Deleting a blob from the Cold or Archive tiers incurs a prorated charge, if the minimum storage period has not been met. Moving a blob from Hot to Cool or Cool to Archive or Hot to Archive tiers incurs write operation charges. Moving a blob from Archive to Cool or Cool to Hot or Archive to Hot incurs read operation charges.

Once you've created the Storage account, created a container, and copied some files, you'll be prepared to set up multiple containers and multiple Azure Storage on demand. Creating Azure Storage is one of the basic skills needed to store data files in Azure. If you are the sole admin for

your business, this may be sufficient. For most use cases, user access configuration follows initial creation. Giving access to users expands the usefulness of the storage services. Let's look at methods for allowing secure access to the storage services.

3.4 Storage access

You have created the Storage account and a Blob service. But the service is not useful for anyone else in its current state. In order for others to use it, you need to define, plan, and implement an access scheme. An access scheme defines who is allowed to access resources and what they are allowed to access. In order to provide secure access, you'll look at how the file hierarchy of Azure Storage influences its security model. Then you'll see how to plan and implement file access by working through an access scenario. By the end of the section, you'll be able to configure secure file access for Azure Storage.

3.4.1 Problem definition: Backup files from two departments to common cloud storage. Maintain separate security access.

Finance and Operations would like to have a joint file archive that can be used with Data Lake Analytics. Separate Finance and Operations users and systems will be used to upload files from each department. Access for uploading files for one department will not allow reading files from the other department. You need to design and implement an appropriate access scheme in a Storage account which satisfies these requirements.

Up to this point, the Azure Storage you've created have been accessible only by the owner, you. Now you need to broaden access to include other users. This scenario applies the principle of least priviledge by restricting access to department files and folders to members of the department. By working through this scenario, you'll learn some approaches to configure security controls on Azure Storage and Data Lake stores.

And there is a new element to consider; an Azure service, Data Lake Analytics. As the name suggests, this service runs analytics jobs over Data Lake stores. You'll cover creating and using Data Lake Analytics in Chapter 7. For now, keep these things in mind:

- Data Lake Analytics can use files from Data Lake stores **and** Azure Storage.
- An Azure user grants Azure Storage access, to Data Lake Analytics for running jobs, via an access key.

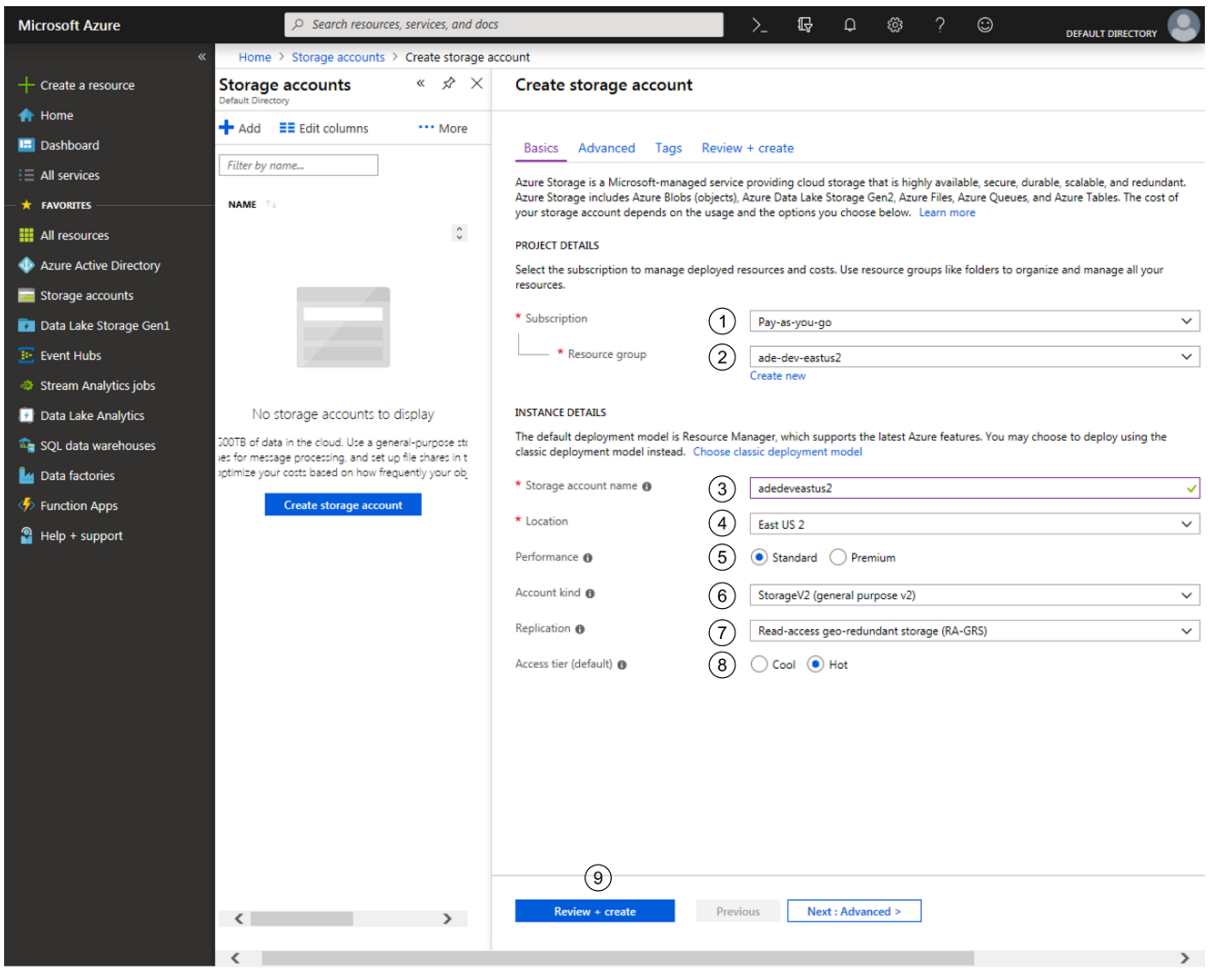
3.4.2 Designing Storage account access

How do you set up a Storage account so that you can have a common file location but still restrict access to certain files? You can create two Blob containers in the Azure Storage account, one for each department. Separate SAS tokens for each department would be used to control access. The single Azure Storage account approach relies on SAS tokens for access; giving the department's AAD user access to the Azure Storage account would give both departments access to all the Blob containers.

Or you can create two Azure Storage accounts, with a single Blob container each. Each department will have access to their respective Storage account and any containers within it via AAD authorization. This approach doesn't quite meet the narrow requirements of a "joint file archive", but you can add multiple Azure Storage to a single Data Lake Analytics service so it will still meet the goal.

CREATE AN AZURE STORAGE ACCOUNT

Let's create a single Storage account with two Blob containers, one for each department. You can refer back to figure 3.1 for more instructions.

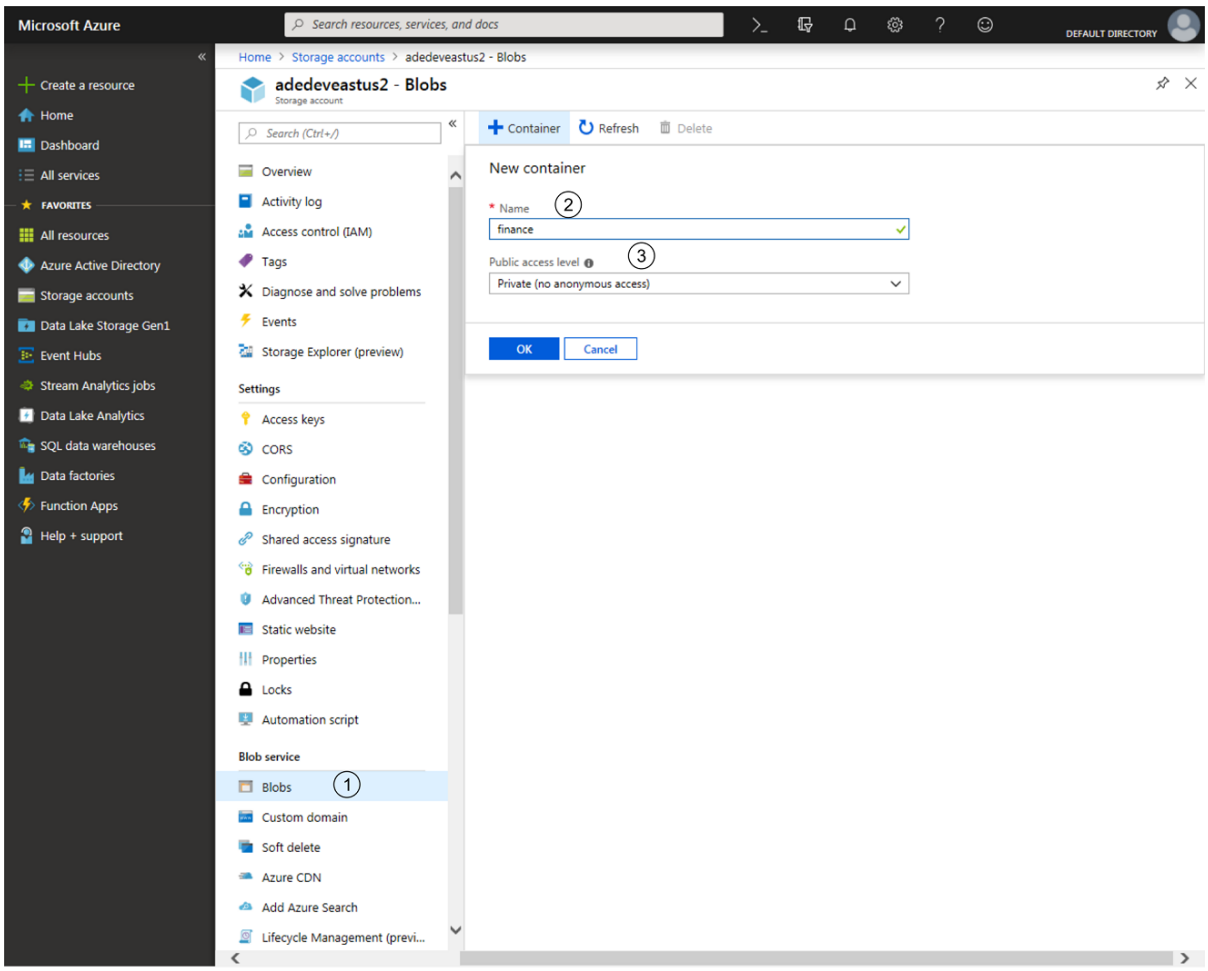


1. Use the same subscription for related resources.
2. Use the same resource group for related resources.
3. Name must be alphanumeric only.
4. Keep related resources in the same location for best performance.
5. Premium is only for VM disks.
6. Leave the default StorageV2.
7. Redundant storage incurs higher costs.
8. Choose Cool is you will rarely access these files.
9. Review, and then Create the Storage account in two steps.

Figure 3.6 Creating a Storage account

Now create two Blob containers using the Azure Portal.

1. In your Storage account, click **Blobs** under "Blob service" to show the "Blobs" blade.
2. Click **Container** to show the "New container" wizard.
3. Enter "finance" for the Name.
4. Leave Public access level at the default "Private".
5. Click **OK** to create the container.
6. Repeat for "operations".



(1) Blobs containers

(2) Use lowercase letters, numbers, and hyphens

(3) Leave this Private.

Figure 3.7 Creating a Blob container

The Storage account is ready for access controls to be added.

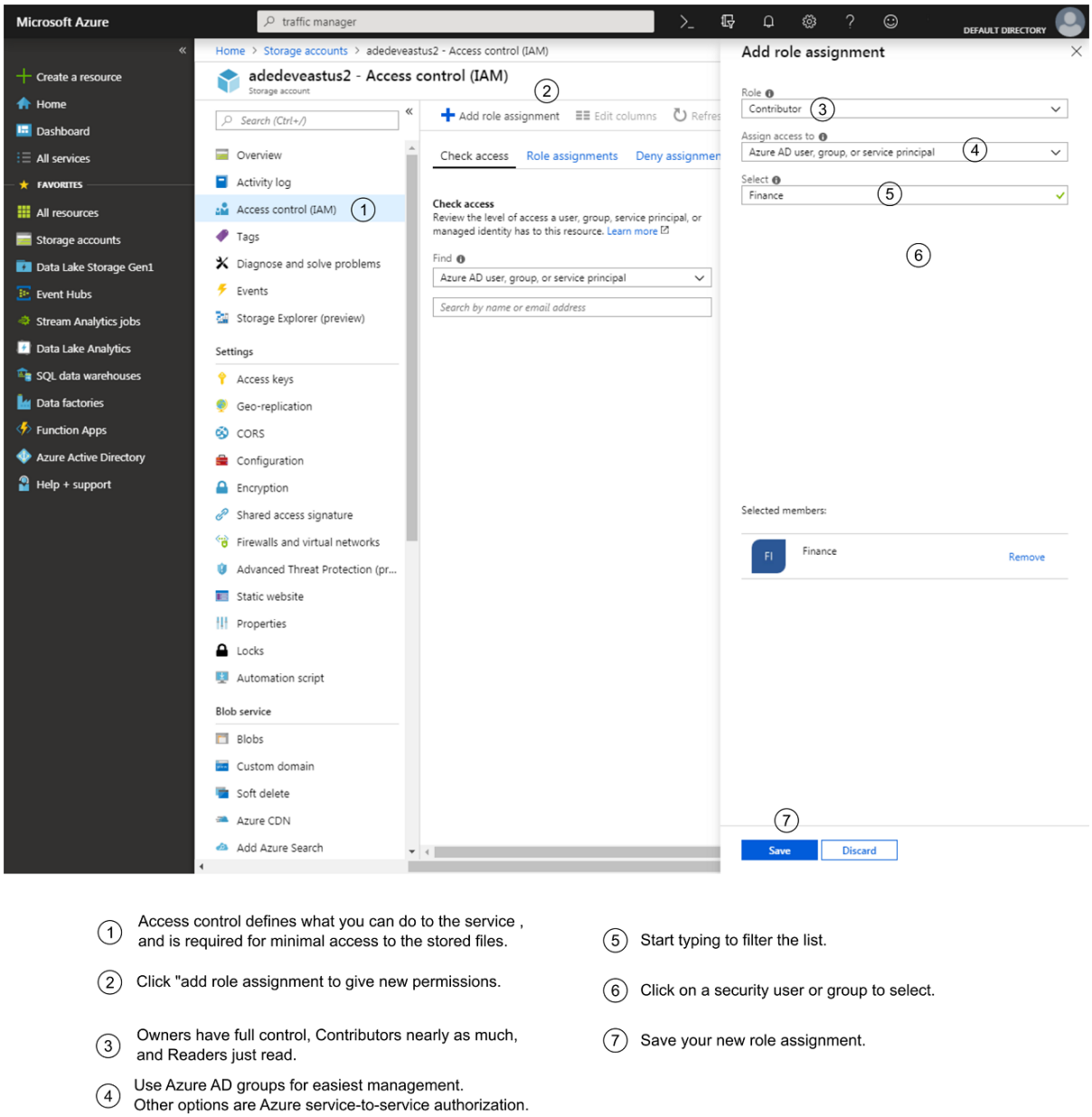
HIERARCHY STRUCTURE IN AZURE STORAGE

When you created the Storage account, you also needed to create a container in order to do anything useful. The container is the root of the file hierarchy for Blobs and Files, so it must be present in order to address any files. Both Files and Blobs emulate a folder structure, and have an addressable URL, like ABCDEF.blob.core.windows.net/ABCDEFcontainer/ABC/DEF/123.csv. Clicking the blob in the Azure Portal container Overview blade brings up the properties window, which displays the URL.

As the owner of the Storage account, you have full access to any of the files in the Blobs and Files containers via your Azure login account. You can assign role-based access to other Azure login accounts too, and grant them access at the Azure Storage account level. This access covers

all services in the Storage account, including Blobs and Files containers. You can use the Azure Portal to configure the roles.

1. Click **Access control (IAM)** in the Azure Storage service blade to show the Access control blade.
2. Click **Add role assignment** to show the Add role assignment blade.
3. Select "Reader" for the Role.
4. Select the default "Azure AD user, group, or service principal" to Assign access to. Other options allow authorization by Azure services.
5. Click "Finance" group to select the group.
6. Click **Save** to add the role assignment.



1. Access control defines what you can do to the service , and is required for minimal access to the stored files.
2. Click "add role assignment to give new permissions.
3. Owners have full control, Contributors nearly as much, and Readers just read.
4. Use Azure AD groups for easiest management. Other options are Azure service-to-service authorization.
5. Start typing to filter the list.
6. Click on a security user or group to select.
7. Save your new role assignment.

Figure 3.8 Assigning AAD group to Azure Storage service roles

But you can't set access permissions on a folder. Blobs and Files services set access permissions

at the container level. Blobs services allow individual file access permissions as well, via a SAS. Access permissions for Files storage are checked with root access keys during network share set up. Access permissions for Blobs storage are checked with root access keys or SAS tokens. Figure 3.9 describes the various access schemes.

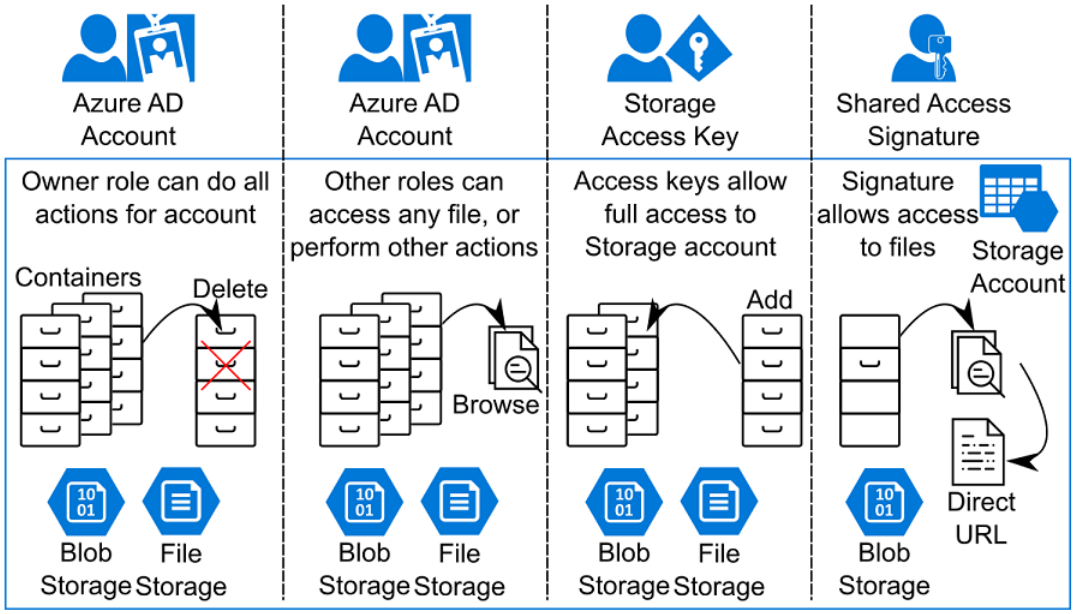


Figure 3.9 Access Storage account resources with Azure AD accounts, access keys, and SAS

SHARED ACCESS SIGNATURES

A SAS, also known as a SAS token, is a URI-friendly hashed string derived from the Storage account root access key. It is generated by an authorized user or application for distribution to an end user. The SAS provides granular access to Storage account resources, like files.

A SAS has multiple properties which define the access granted.

- A start and end date provide a limited window for access.
- A set of permissions grant actions to the user.
- A stored access policy allows a reusable and easily revocable container for permissions.
- A SAS can be generated for the entire Azure Storage account, service container, or individual files.

Service SAS tokens at the container level must be created using the Azure REST API or Azure Powershell module. There isn't a method in the Azure portal to specify the policy when creating a SAS token. You can use Azure Powershell to create a container-level SAS with a 24-hour effective window by providing these options:

- The name of the service container
- A valid start date
- A valid expiration date

- The Azure Storage account containing the service container, as context
- The permissions for the SAS

Listing 3.7 Create shared access signature for container

```
$Now = Get-Date
$StartTime = $Now.AddMinutes(-15.0)   ❶
$EndTime = $startTime.AddHours(24.0)   ❷
New-AzStorageContainerSASToken -Name "finance"   ❸
<linearrow /> -StartTime $StartTime -ExpiryTime $EndTime   ❹
<linearrow /> -Context (New-AzStorageContext -StorageAccountName
<linearrow /> "adedeveastus2" -StorageAccountKey "<storage key>")   ❺
<linearrow /> -Permission rwcl   ❻
```

❶ Set start date back 15 minutes, to prevent errors due to time sync between services.

❷ Set expiration date for 24 hours hence.

❸ Specify the Container finance.

❹ Use the Powershell variables.

❺ Include the Context, which is the Storage account. Use your Azure Storage key to authenticate.

❻ Specify permission of Read,Write,Create,List as string.

Powershell returns the value of the SAS immediately. Add option -FullUri to return the full URL to the Azure Storage, container or file targeted. The default expiration is one hour after the start time. Any valid date will work for the expiration, as long as it follows the start time.

STORED ACCESS POLICY

To create a new stored access policy for a container, you must choose a container, a name, and a permission set. SAS tokens should have a start and expiry time, but you can remove access after a set time by adding a start and end time to the associated policy. Using shorter duration SAS tokens will improve security by minimizing the window of use of compromised keys. SAS tokens can also be revoked by modifying the associated policy.

To create an access policy, execute the script Create Storage account access policy in a Powershell window with Azure Powershell module. The script creates a policy "AddFiles" on the "Finance" container you created earlier. The script makes a connection to the Storage account where you created the "Finance" container service.

Listing 3.8 Create Storage account stored access policy

```
New-AzStorageContainerStoredAccessPolicy -Container "finance"
<linearrow /> -Policy "AddFiles" -Permission rwcl   ❶
<linearrow /> -Context (New-AzStorageContext -StorageAccountName   ❷
<linearrow /> "adedeveastus2" -StorageAccountKey "<storage key>")   ❸
```

❶ Create a policy named "AddFiles". Specify permission of Read,Write,Create,List as string.

❷ Include the Context, which is the Storage account.

❸ Use the Azure Storage key to authenticate.

Create and write permissions allow uploading files, while read and list allow viewing the contents of the container. You can use a service-level SAS token to execute this command instead of the root service key, to avoid exposing the root access key. To create a SAS token using the access policy, execute the script Create SAS with access policy in a Powershell window with Azure Powershell module.

Listing 3.9 Create SAS with access policy

```
$Now = Get-Date
$StartTime = $Now.AddMinutes(-15.0)   ❶
$EndTime = $startTime.AddHours(24.0)   ❷
New-AzStorageContainerSASToken -Name "finance" -Policy "AddFiles"   ❸
<linearrow /> -StartTime $StartTime -ExpiryTime $EndTime   ❹
<linearrow /> -Context (New-AzStorageContext -StorageAccountName
<linearrow /> "adedeveastus2" -StorageAccountKey "<storage key>")   ❺
<linearrow /> -FullUri   ❻
```

❶ Set start date back 15 minutes, to prevent errors due to time sync between services

❷ Set expiration date for 24 hours hence

❸ Specify the Container and the Policy set up previously

❹ Use the Powershell variables

❺ Include the Context, which is the Storage account. Use the Azure Storage key to authenticate.

❻ Powershell returns the value of the SAS immediately, and adds the URL of the Storage account and Container

Using Powershell variables enables reuse of values in a session. It also makes your script easier to read, by shortening the lines. The Create SAS with access policy script can be executed multiple times, each time returning a new signature and expiration. Make sure you have created the stored access policy before trying to create a SAS tied to it.

TIP Add Immutable blob storage policy to a Blob container to prevent **any** modifications to files in the container, even for the Owner role. It does not need to be attached to an SAS token to be effective.

Now that you've learned how to create the Storage account and access keys, you can create useful storage services. In this scenario, you explored securing access in Azure Storage. While it's possible to create discrete security configurations in a single Storage account, creating SAS tokens relys on an external service for authorization. Ongoing usage of the Storage account using SAS tokens requires the client to renew the keys periodically. Alternately, separate Azure

Storage provide clear security boundaries but create difficulties in addressing the stored files without switching contexts.

3.5 Summary

In this chapter, you learned the following:

- Every Azure service must have a name. Designing an effective naming convention makes managing resources easier.
- Azure Storage consists of several services. Blobs service provides cost effective storage in the cloud.
- Microsoft provides several tools for copying files into Azure Storage, each with a strong use case.
- Azure Storage access is granted at the Storage account level using AAD. Granular access at the container/service level is granted using access keys. Granular access at the container/service level and the blob level is granted with shared access signatures.

4 Azure Data Lake storage

This chapter covers:

- Choosing between Blob storage and Data Lake storage
- Setting up a Data Lake store
- Configuring file access in Data Lake storage
- Understanding and planning for data drift

In chapter 2, you explored a set of services provided by Microsoft Azure: Event Hubs, Stream Analytics, Data Lake store (ADL), Data Lake Analytics (ADLA), and SQL Data Warehouse (SQLDW). You saw at a high level how these services can work together to create an analytics system. This chapter continues the design and set up of these services to lay the foundation of an analytics system with ADL. You'll learn how to implement a storage system with ADL that is secure and change-tolerant. The storage system will be the central service around which you construct the analytics system.

4.1 Storage services compared

Azure provides two types of storage services: Azure Storage accounts and ADLs. Azure Storage accounts are covered in chapter 3. Unlike Azure Storage account services, ADL resembles a local file system, with folders and files. Azure Active Directory (AAD) controls access to folders and files, with assignable read/write/execute permissions.

- Both types of storage offer huge amounts of storage space, and both integrate easily with other Azure services.
- Both can be used to store files.
- Both have similar steps for set up.

In chapter 3, you worked through the backup IIS logs scenario using a Storage account. Working

through creating and configuring a Storage account can help with set up of ADLs, but it's not required. Let's look at the steps for creating an ADL, by using this common archiving scenario again.

4.1.1 Problem definition: backup IIS logs

In this scenario, the IT department wants to copy all IIS logs from 10 web servers to cheap long-term storage ahead of deletion. The log files will be rarely accessed, and lowest cost is important. You need to set up a service in Azure and copy the log files.

4.1.2 Create an Azure Data Lake store

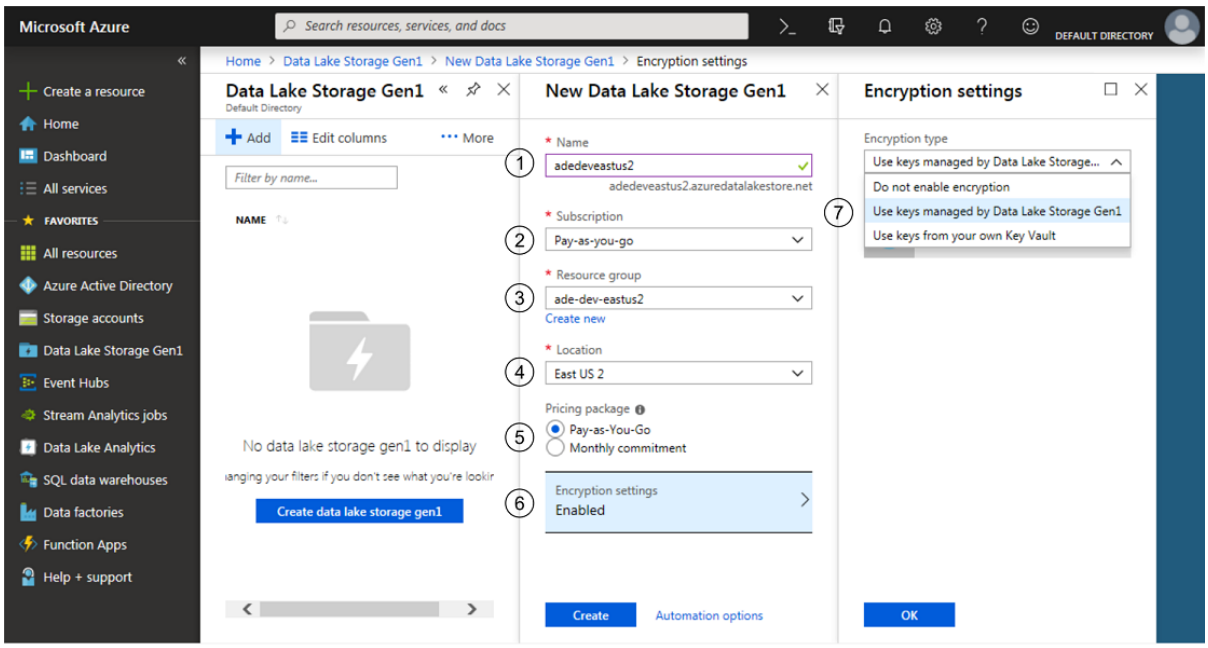
Setting up an ADL requires a few pieces of information common to all Azure services, including a subscription, name, location, and resource group.

- A subscription groups resources together for access control and billing.
- A resource group groups resources together for management.
- A location groups resources into a regional data center.

> **IMPORTANT** By default, all files in ADL are encrypted at rest. You should leave the management of the encryption keys to the service unless you have a system in place to manage them.

Here's how to create a new ADL.

1. In the Azure Portal, use the "Create a resource" menu to open a "New Data Lake Storage Gen1" blade, or use the "All services" menu and filter on Data Lake Storage Gen1 to show the Data Lake Storage Gen1 blade. Or you can go directly to the "New Data Lake Storage Gen1" blade at portal.azure.com/#create/Microsoft.AzureDataLakeStore.
2. Choose a name ("[XYZ]deveastus2"). The Data Lake store name must be lowercase alphanumeric, between 3-24 characters, and globally unique. Read more about Azure service naming conventions in chapter 3.
3. Choose a subscription. The default will be the oldest subscription, if you have access to more than one.
4. Choose a resource group. (See Appendix A for instructions if you haven't created one.)
5. Choose a location. ADLs are not available in all regions; choose one close to you. Keeping Azure resources which interact in the same region minimizes network latency. You may choose a region to match your user base, as some governments restrict movement of data outside their zone of control.
6. Choose a pricing package, or leave the default Pay-as-you-go. This minimizes costs for your tutorial usage. In a production system, reserving storage up front provides discounts. See azure.microsoft.com/pricing/details/data-lake-storage-gen1/ for more information.
7. Choose an encryption management scheme, or leave the default Enabled. To use a self-managed key for encryption, you will need to create an Azure Key Vault, and an encryption key.
8. Create the Data Lake store.



(1) Use lowercase letters and numbers.
(2) Use the same subscription for related resources.
(3) Use the same resource group for related resources.
(4) Keep related resources in the same location for best performance.
(5) Calculate the threshold for savings before committing.
(6) Use encryption.
(7) Unless you have established key management systems, let the service manage the keys.

Figure 4.1 Creating a Data Lake store

You can also create an ADL via Azure Powershell. Access Azure Powershell by visiting Azure Cloud Shell at shell.azure.com/, or clicking the ">_" header menu in the Azure portal.

Listing 4.1 Create new ADL

```
New-AzDataLakeStoreAccount -ResourceGroupName "ade-dev-eastus2"
<linearrow /> -Name "adedeveastus2" -Location "East US 2"
```

This PS script will return an error if a Data Lake store by that name exists, or the service is not available in the selected region. Keep resources within the same region. ADL is available in the fewest regions. It's a good idea to select one of these regions before creating the rest of your services in Azure, so that you keep all services in the same region. This lowers the latency of network communication between services.

You can also specify some other options during setup. You can add key/value pairs to Azure services, called "Tags", to help locate the service later. If you know your storage size, you can pre-purchase storage at a discounted rate.

Listing 4.2 Create new Data Lake store with options

```
New-AzDataLakeStoreAccount -ResourceGroupName "ade-dev-eastus2"
<linearrow /> -Name "adedeveastus2" -Location "East US 2"
<linearrow /> -Tag @{User="ADE";}   ❶
<linearrow /> -Tier Commitment1TB   ❷
```

❶ Create a tag called User with value ADE

❷ Prepurchase 1TB of storage each month and save 12% over basic rate

Add a Tag for management of resources. This is especially nice when browsing the "All resources" blade in the portal, because you can select from a list of all the tags you have provided. Use a consumption plan, until you've calculated your monthly storage needs.

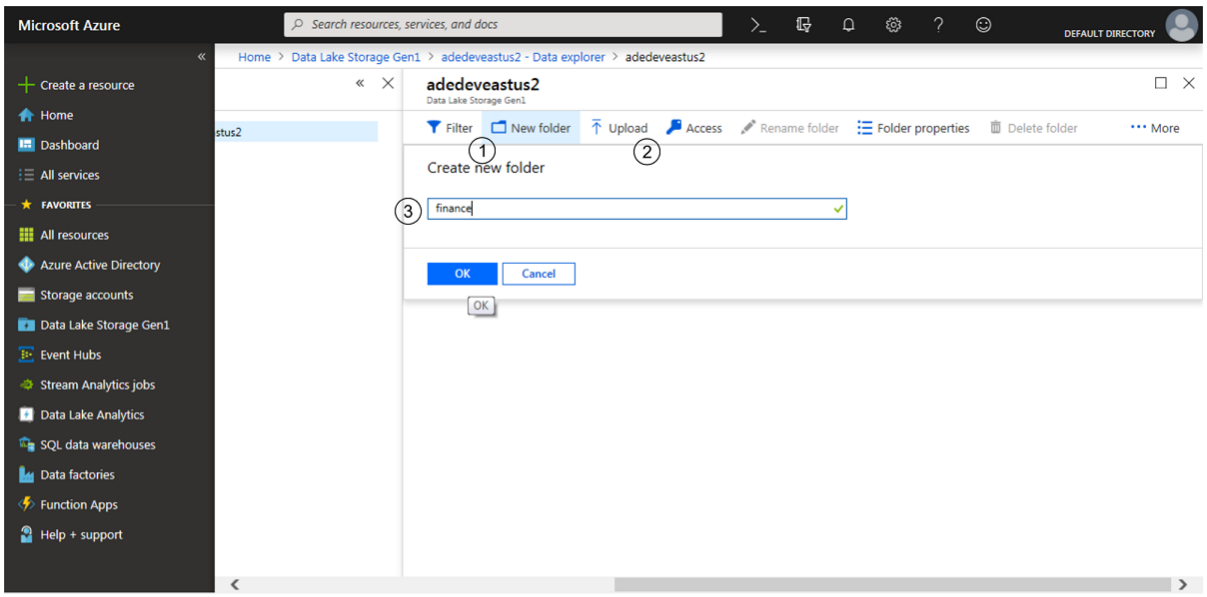
> **TIP** If you know in advance how much data you will store in the Data Lake, purchasing committed capacity in advance can save you money. Once your storage level passes a threshold close to the commitment capacity, you can reduce your spending. For example, if you are archiving more than 900 GB of log files, purchasing 1TB of capacity will cost less that the pay-as-you-go total. Overages are calculated at the standard rate of $0.039/GB.

There are fewer options to select when creating a ADL than a Storage account. The primary selection regards pricing. Do you have enough expected storage needs to prepurchase capacity? As you prepurchase capacity, the cost of ADL approaches that of Storage account Blobs, but doesn't match it.

4.1.3 Copy tools for Data Lake store

The ADL is ready to accept files from a number of tools. You can copy files directly into the root of the Data Lake. However, a better approach groups files into folder by security, functional, or source boundaries. In 3.5 you'll look at configuring the Data Lake to allow secure access to the files. For now, use the Azure portal to create a folder and upload the files.

1. In the "All services" blade, enter "Data Lake Storage Gen1" in the filter, and select the "Data Lake Storage Gen1" service type to see your ADLs. Click on the ADL you created in the previous section "[XYZ]deveastus2".
2. In the "Overview" blade of the ADL, click "Data explorer".
3. In the "Data explorer" blade, click "New folder".
4. Name the folder **iislogs**. Folder names can be any string of characters that are valid in a URL.
5. Click the folder **iislogs** to browse to it.
6. Click "Upload" to view the "Upload files" blade.
7. Click folder icon to open a File Select dialog.
8. Select your log files, and click "Open" to begin uploading them to the **iislogs** Data Lake folder.



(1) Once in the "Data Explorer" blade, you can create new folders.

(2) You can upload files. Browse into the target folder first.

(3) Name can be any string of characters that can be URLEncoded.

Figure 4.2 Creating a new folder in Data Lake store

Several Microsoft tools operate within Azure, copying files between services. Keeping the data transfer in Azure, rather than downloading and uploading files, minimizes network egress charges. Network transfers within an Azure data center would also be faster than across the Internet. ADLCopy is a command-line tool from Microsoft for copying files from Storage accounts to ADLs, and between ADLs. ADLA can perform the same functions as ADLCopy. (You can read more about ADLA in chapter 7.) Azure Data Factory (ADF) uses cloud scheduling and Azure runtimes, including ADLA, to copy data between services. (You can read about ADF in chapter 10.) You can even export files directly from SQL Data Warehouse to ADL. Chapter 3 discussed copying files into Storage accounts. Figure Figure 3.4 adds two more options for tools to copy files to Azure storage services.

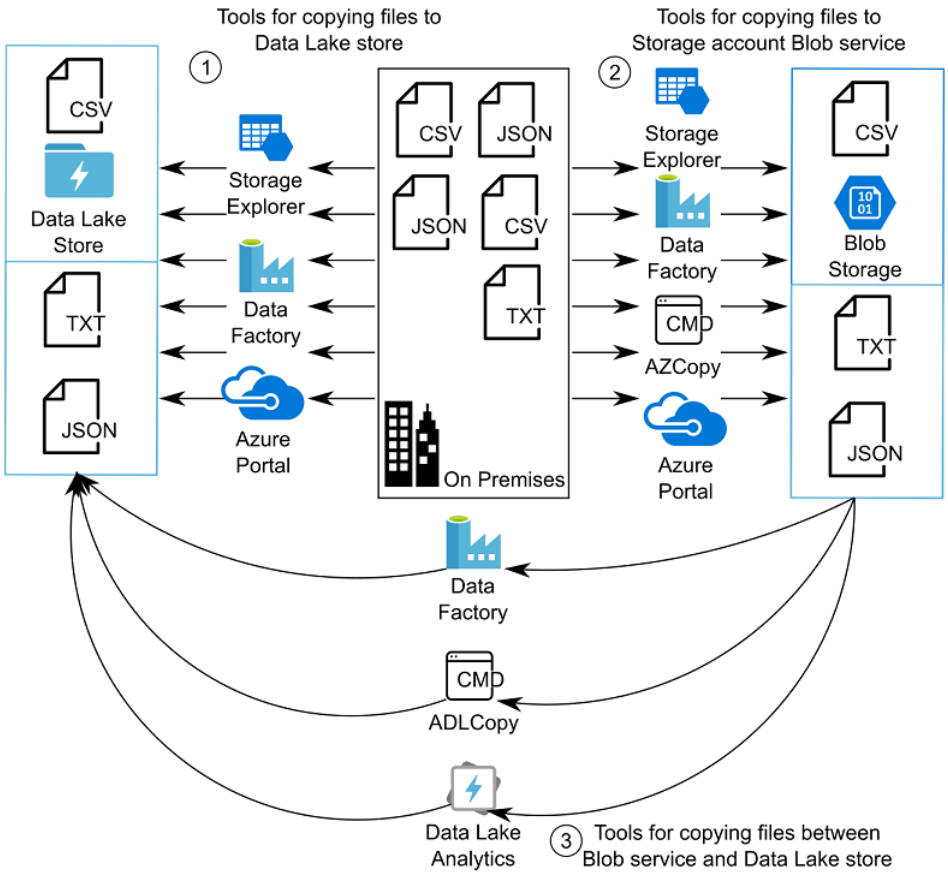


Figure 4.3 Tools for copying files between storage services

Each tool has a strong use case.

- The Azure Portal is available without an install.
- ADLCopy can be used for automated file copies without user interaction.
- File copies with Azure Data Factory can be included in multi-step workflows, and integrate with other Azure services.
- Storage Explorer provides an easy to use GUI and status tracking of actions.
- ADLA can retrieve data during processing jobs.

ADLCOPY TOOL

ADLCopy is a command-line tool from Microsoft for copying files between Storage accounts and ADLs, and between ADLs. It doesn't copy files from on-premises stores to Azure Data Lake store. Because the file copy occurs between storage systems in Azure, Azure resources are used to execute the copy. Internet bandwidth and single system constraints are removed. You can download ADLCopy at aka.ms/downloadadlcopy.

The copy can run in standalone (shared) mode or ADLA (dedicated) mode. With standalone mode, Azure executes the job using available shared resources for ADL. With ADLA mode, you

configure the transfer to use dedicated resources, and tweak the number of analytics units to balance cost and speed of the transfer. Dedicated resources ensure no throttling occurs during the transfer.

When using ADLCopy in standalone mode from a Storage account to an ADL, commands include 4 parameters:

- *Source* is the path to the files.
- *Dest* is the target of the file copy.
- *Sourcekey* is the root key or shared access signature key. (see chapter 3)
- *Pattern* is a regex pattern to match the files for copying. Pattern is optional, and not supplying a pattern will copy all files in the Source path.

The following listing shows the use of the ADLCopy tool to copy files from a Storage account to an ADL.

Listing 4.3 ADLCopy transfer standalone

```
"C:\Program Files (x86)\Microsoft SDKs\Azure\ADLCopy\adlcopy"
<linearrow  />/Source https://abc.blob.core.windows.net/project-abc/v1/v1.1
<linearrow  />/Dest adl://abc.azuredatalakestore.net/iislogs/v1/v1.1/   ❶
<linearrow  />/sourcekey ==StorageKey== /Pattern "ch*.csv"   ❷
```

❶ Replicating the folder structure

❷ Use file patterns for finer control

NOTE See Hierarchy structure revisited later in this chapter for a discussion of folder hierarchies and versioning in the Data Lake.

When using ADLCopy in ADLA mode from a Storage account to an ADL, commands include 6 parameters:

- *Source* is the path to the files.
- *Dest* is the target of the file copy.
- *Sourcekey* is the root key or shared access signature key. See chapter 3 for more details.
- *Pattern* is a regex pattern to match the files for copying. Pattern is optional, and not supplying a pattern will copy all files in the Source path.
- *Account* is the name of the ADLA to use for executing the copy job.
- *Units* specifies how many analytics units to use for the job. See chapter 7 for a discussion of ADLA analytics units.

The following listing shows the use of the ADLCopy tool to copy files from a Storage account to an ADL, using your existing ADLA to execute the job.

Listing 4.4 ADLCopy transfer with ADLA

```
"C:\Program Files (x86)\Microsoft SDKs\Azure\ADLCopy\adlcopy"
<linearrow  />/Source https://finance.blob.core.windows.net/datalakeload/p-abc/
<linearrow  />/Dest adl://abc.azuredatalakestore.net/staging/finance/p-abc-v1.2/   ❶
<linearrow  />/sourcekey ==StorageKey== /Pattern "tv*.csv"   ❷
/Account dedeveastus2 /Units 2   ❸
```

❶ Implement change in hierarchy by targeting new folder

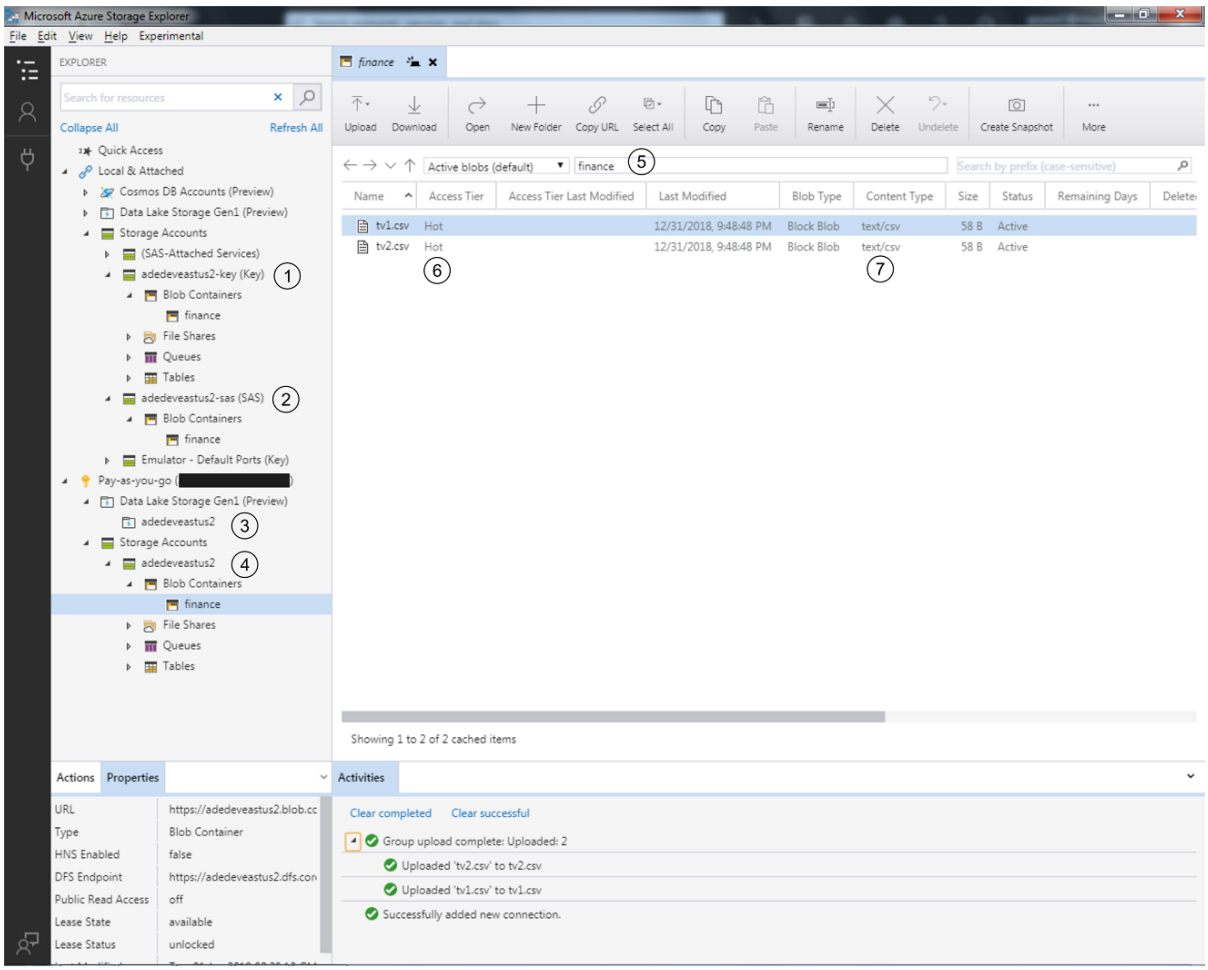
❷ Use file patterns for finer control

❸ Add ADLA account after file name pattern, use 2 parallel workers.

When using the ADLA account, the *Pattern* switch must be placed before the *Account* and *Units* switches. Unattended copy executions are possible. On first execution, ADLCopy prompts for Azure credentials. These are saved in %AppData%\ADLCopy\TokenCache.dat file. This file will then provide authentication for a scheduled execution of ADLCopy. You'll need to add the Storage account and ADL account as a data source in ADLA, if not already attached. You can read more about ADLA in chapter 7.

AZURE STORAGE EXPLORER TOOL

Azure Storage Explorer provides a desktop GUI interface for uploading files to multiple Azure services, including Storage accounts. Azure Storage Explorer can also connect to ADLs using AAD. MIME types are identified based on file extension. Figure Figure 3.5 shows Storage Explorer connecting with multiple types of authentication. You can use the drag-and-drop function to upload and download files, or use the individual function buttons in Storage Explorer. Download Azure Storage Explorer at azure.microsoft.com/features/storage-explorer/.



1. Use the account key to connect to a Storage account. All resources and services are accessible.
2. Use a shared access signature to connect to a Storage account. Only resources and services in the SAS are accessible.
3. Connect to Data Lake with your Azure Active Directory (AAD) user.
4. AAD user provides access to Storage account resources and services based on role. Nearly all roles grant read access.
5. Storage Explorer interprets the folder structure of the files in a Storage account Blob service.
6. Use Storage Explorer to change Access Tier: Hot, Cool, and Archive.
7. Store many file types with recognized MIME types.

Figure 4.4 Storage Explorer configured to connect to Storage accounts with access keys and SAS keys

> **NOTE** Data explorer, and other Azure services, doesn't create sub-folders automatically, but copies files only. Other tools, like Storage Explorer or ADLCopy, will copy files and folder structure. Creating folders yourself, and carefully planning the structure of your folder hierarchy, will help keep your Data Lake from turning into a Data Swamp!

Creating Storage accounts and ADLs are the basic skills needed to store data files in Azure. If you are the sole admin for your business, this may be sufficient. For most uses, user access configuration follows initial creation. Giving access to users expands the usefulness of the storage services. Let's look at methods for allowing secure access to the storage services.

4.2 Storage access

You created the Data Lake store, but the service is not useful for anyone else in its current state. For others to use it, you need to define, plan, and implement an *access scheme*.

4.2.1 Access schemes

An access scheme defines who is allowed to access resources and what they are allowed to access. Authentication encompasses validation of the identity of the entity making the access request. Authorization matches authorized entities with the actions they are allowed to perform. Authentication for ADL is handled by AAD.

AAD is Microsoft's cloud-based identity and access management service. It provides sign-on services and account management for Office 365, the Azure Portal, and other Internet applications. AAD can use on-premises directory synchronization for single sign-on for on-premise and cloud applications using the same account. On-premise users and groups are available for use in Azure with AAD directory synchronization. Authentication is handled at the user level; authorization can be set for users and groups.

Authorization for ADL actions is defined in two ways. For managing the ADL service, Role-based Access Controls (RBAC) allow or deny access to tasks like deleting the ADL, assigning roles to users, and purchasing reserved storage. For managing file and folder access in the ADL, Access Control Lists (ACLs) define granular access. The ACLs use the standard form Read (R), Write (W), and Execute (X) permissions. To see how these two authorization approaches work, you'll look at the folder hierarchy and security model for ADLs. Then you'll see how to plan and implement file access by working through an access scenario. By the end of the section, you'll be able to configure secure file access for ADLs.

4.2.2 Problem definition: Backup files from two departments to common cloud storage. Maintain separate security access.

Finance and Operations would like to have a joint file archive that can be used with ADLA. Separate Finance and Operations users and systems will be used to upload files from each department. Access for uploading files for one department will not allow reading files from the other department. You need to design and implement an appropriate access scheme in an ADL which satisfies these requirements.

LEAST PRIVILEGE

Up to this point, the ADLs you've created have been accessible only by the owner, you. Now you need to broaden access to include other users. The Principle of Least Privilege states that "a subject should be given only those privileges needed for it to complete its task" (Gegick). This scenario applies the principle by restricting access to department files and folders to members of the department. By working through this scenario, you'll learn some approaches to configure RBACs and ACLs on ADLs.

4.2.3 Configuring ADL access

How do you set up your new ADL to have a common file location but still restrict access to certain files? To implement the project Backup files from two departments, you need separate folders for Finance and Operations, with access and default ACLs specific to each department.

ROOT FOLDER ACLS

Folders get an access ACL and a default ACL. Files get an access ACL only. The access ACL determines access for the folder or file itself. The default ACL determines the access ACL for files created in the folder, and the default ACL for child folders. Access ACLs can be set using the "Access" blade of a file or folder in the "Data explorer" blade of an ADL, in the Azure Portal. Default ACLs can be set using the "Advanced" blade in the "Access" blade of a folder in the "Data explorer" blade of an ADL, in the Azure Portal. The "Advanced" blade also allows applying ACLs to child folders.

The folder IISLogs you created in 4.1.2 copied the ACLs from the "/" root folder of the Data Lake store. When the Data Lake is first created, the root folder has a unique access ACL and default ACL. The AAD account used to create the ADL is assigned to the root folder owner role. Since you are the creator, you are the root folder owner by default. A security group, with an all-zero GUID, is also assigned to the root folder owner role, to satisfy a requirement for having a security group as an owner. This null security group does not permit access, and should be replaced with a valid security group. This ensures access to the folders and files in case access via the ADL creator account is lost. You'll replace the owning group in the next section.

ROOT FOLDER OWNER

Assigning ownership of the root folder, and assigning access and default ACLs to the root folder, create the broad outlines of your access scheme. If you don't have an AAD user and group, see appendix A for a PowerShell script to create them.

Azure uses Azure Active Directory (AAD) extensively for user and service authentication. You should already be familiar with the Azure Portal, and using Active Directory for authentication and authorization to Azure services. Here is an Azure Powershell script for creating a new user, a new security group, and assigning the user to the group. The new user command requires a display name, a mail name, a principle name which is an email address, and a password. The

security group command requires a display name, and a mail name. The group membership command requires a combination of group and member identifiers, either user principle name or Id, and group display name or object or Id. You need to construct a UserPrincipalName using one of the AAD registered domains. For a personal Azure account, use your signup email without the @ and top-level domain, and append ".onmicrosoft.com".

Execute these lines in Powershell with the Azure Powershell module loaded.

Listing 4.5 New AAD user and group

```
$SecureStringPassword = Read-Host -Prompt "Enter password" -AsSecureString   ❶

$User = New-AzADUser -DisplayName "Tech User"
<linearrow /> -Password $SecureStringPassword -MailNickname "techuser"   ❷
<linearrow /> -UserPrincipalName "techuser@azuredomain.onmicrosoft.com"   ❸

$Group = New-AzADGroup -DisplayName "Technical Operations"
<linearrow /> -MailNickname "TechOps"

Add-AzADGroupMember -MemberObjectId $User.Id   ❹
<linearrow /> -TargetGroupObjectId $Group.Id   ❺
```

❶ Prompt for a password for the new user.

❷ Use the secure password.

❸ Build the principle name from MailNickname and your AAD registered domain.

❹ Get the Id from the variable $User, from the new user command.

❺ Get the Id from the variable $Group, from the new group command.

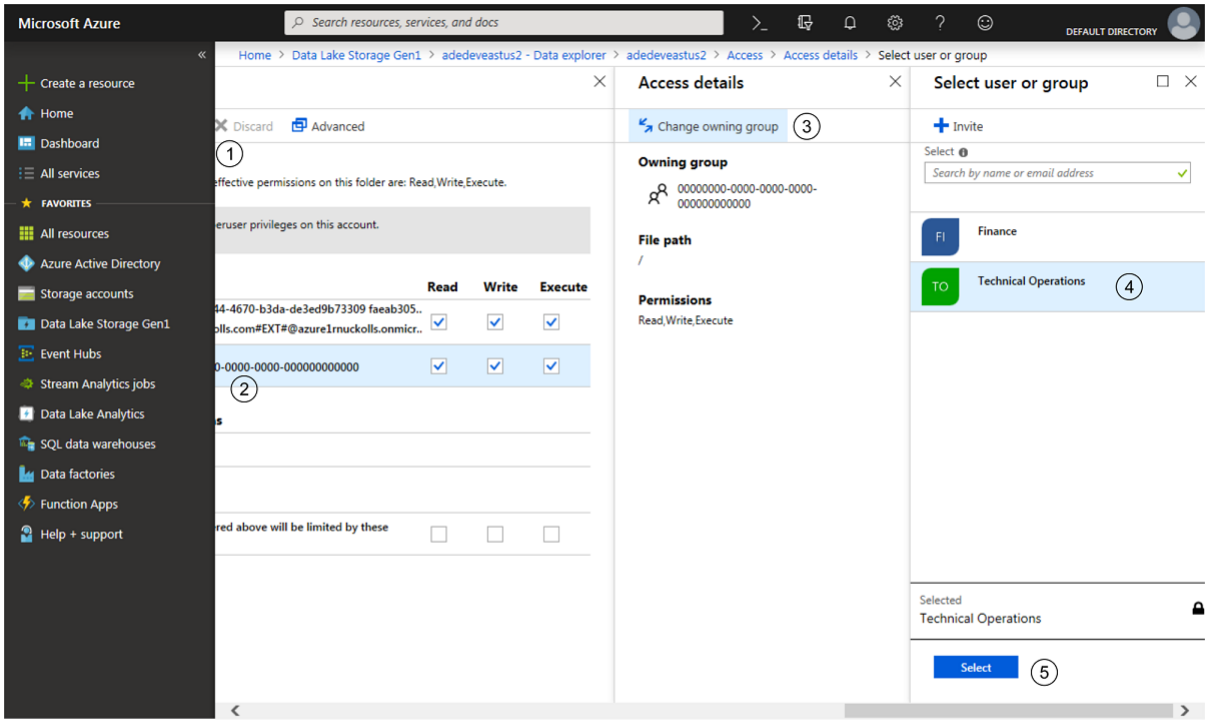
This PS script will return an error if a group by that name exists. The new user and security group have no access in this time, but allow authentication to Azure and the various services. Next, you'll give the group, and through it the user, access permissions in the ADL.

NOTE If you are using an Azure subscription without a corporate Active Directory, then your domain will be some variation of the email you used to sign up with Azure. You can find this value by going to the Azure Active Directory service "Overview" blade. The domain is listed above the header "Default Directory". The domain is also listed in the "Custom domain names" blade.

Now you've created a user and a security group in AAD using Powershell. You can use them when securing the ADL root directory. Set the owning group to the "TechOps" group using the Azure Portal.

1. In the Azure Portal, use the "All services" menu and filter on Data Lake Storage Gen1 to show the Data Lake Storage Gen1 blade.
2. Select your ADL to display the "Overview" blade.
3. In the "Overview" blade, click "Data explorer".

4. In the "Data explorer" blade, click "Access".
5. In the "Access" blade, verify that below the header "/ (Folder)" is displayed, indicating you have selected the root folder.
6. In the "Owners" section, click the group "00000000-0000-0000-0000-000000000000".
7. In the "Access details" blade, click "Change owning group". This opens an AAD search blade.
8. In the "Select user or group" blade, search for and select the TechOps" AAD group, and click Select.



1. Browse into root folder and click "Access".
2. Click the null security group.
3. Click "Change owning group.
4. Select a security group from Azure Active Directory.
5. Click "Select" to commit change.

Figure 4.5 Assign owning group to Data Lake store folder

You can also set the owning group on the Data Lake store root folder / with PowerShell. For account, use the name of the Data Lake store you wish to update. The forward-slash character following -Path option indicates the root folder. This change is to the security group owner type, passing in the ID of the "Technical Operations" AAD group. Execute this line in Powershell Core with the Azure Powershell module loaded.

```
Set-AzDataLakeStoreItemOwner -Account "adedeveastus2"
<linearrow /> -Path / -Type Group  ❶
<linearrow /> -Id (Get-AzADGroup -DisplayName "Technical Operations").Id  ❷
```

❶ Group owner instead of user owner

❷ Get the security group Id

"Get-AzADGroup" returns a security object, which has an Id property. Instead of inlining the group object lookup, you could include the GUID directly.

Next, set a fallback access ACL for non-owners on the root folder. This ACL has Read (R) and Execute (X) permissions. Without this ACL, non-owner AAD users will not be able to list the folder structure from the root folder with any of the available tools. Users with access to specific files can access them directly via URL, in the form adl://adedeveastus2.azuredatalakestore.net/file.csv You can find this path under "Properties" of the file in the Azure Portal "Data Explorer" blade. As an alternative to a general listing access, you could apply specific ACLs for AAD groups as they are added to the Data Lake service, and given access to one or more folders in the store. Try setting the fallback ACL using the Azure portal.

1. In the "Overview" blade of the Data Lake service, click **Data explorer**.
2. The "Data explorer" blade opens in the root folder.
3. Add a file and folder list ACL for everyone on the root folder by clicking **Access**, then checking Read, Execute boxes under "Everyone else".
4. Click **Save** to set the ACL on the root folder.

You can also set the ACL using PowerShell. The ACL entry command takes full words as value for its Permissions option. Values include None, Execute, Write, WriteExecute, Read, ReadExecute, ReadWrite, and All. Execute this line in Powershell Core with the Azure Powershell module loaded.

```
Set-AzDataLakeStoreItemAclEntry -AccountName "adedeveastus2"
<linearrow /> -Path / -AceType Other -Permissions ReadExecute   ❶
```

❶ Use AceType Other to set the "Everyone else" ACL

IMPORTANT Make sure that one of your first steps is assigning a valid security group to the Data Lake service owner role and as the root folder owning group when configuring security on the Data Lake. If your AAD account gets locked out or removed, users in the owning group can still access the Data Lake folders. This rationale applies to the Data Lake service owner role as well. The Data Lake service owner role can view all data in the store, but the root folder owning group cannot manage the Data Lake service, unless they have more than Reader role. You can choose different security groups for different roles. For instance, Technical Operations group can manage a Data Lake service as an owner, while the Analytics or Architects group can be the root folder owning group.

4.2.4 Hierarchy structure in Data Lake store

Storing lots of data in cloud storage enables users to apply analysis over the data. Lowering the implementation effort for new data collection often means accepting unprocessed or uncurated data sets, which then need transformation and data cleansing efforts to prepare them for consumption. Having a designated landing zone for the initial data files marks a clear distinction between the original and processed data files.

You'll look more closely at planning folder hierarchies in storage in the next section Storage folder structure and data drift.

NEW INBOUND FILES FOLDER

Now that you have the root folder in better shape, let's add a top level folder named Staging. The Staging folder is a target for the storing of unprocessed data in the cloud storage service. Finance and Operations will deposit files in folders under Staging. This Staging folder will inherit the owners assigned of the root folder on creation. You will also add an ACL which will allow users to list any files, and view the Staging folder itself. Without this fallback ACL, non-owner users will not be able to browse the folder hierarchy. Use the Azure Portal to create the folder and set the ACL.

1. Create the Staging folder by clicking New Folder, and providing the name Staging.
2. Click on the new Staging folder to browse the folder.
3. Add a file and folder list ACL for everyone on the Staging folder by clicking **Access**, then checking Read, Execute boxes under "Everyone else".
4. Click **Save** to set the ACL on the Staging folder.

This ACL will only be an access permission entry for the Staging folder for everyone else. Folders in Staging will belong to their respective groups, and general access won't be provided. You'll look more closely at this folder structure in the next section Storage folder structure and data drift.

FINANCE AND OPERATIONS FOLDERS

Here is an Azure Powershell script for creating a new Finance user, a new security group, and assigning the user to the group.

Execute these lines in Powershell with the Azure Powershell module loaded.

Listing 4.6 Finance AAD user and group

```
$SecureStringPassword = Read-Host -Prompt "Enter password" -AsSecureString   ❶

$User = New-AzADUser -DisplayName "Finance User"
<linearrow /> -Password $SecureStringPassword -MailNickname "financeuser"   ❷
<linearrow /> -UserPrincipalName "financeuser@azuredomain.onmicrosoft.com"   ❸

$Group = New-AzADGroup -DisplayName "Finance"
<linearrow /> -MailNickname "Finance"

Add-AzADGroupMember -MemberObjectId $User.Id   ❹
<linearrow /> -TargetGroupObjectId $Group.Id   ❺
```

❶ Prompt for a password for the new user.

❷ Use the secure password.

❸ Build the principle name from MailNickname and your AAD registered domain.

❹ Get the Id from the variable $User, from the new user command.

❺ Get the Id from the variable $Group, from the new group command.

Here is an Azure Powershell script for creating a new Operations user, a new security group, and assigning the user to the group.

Execute these lines in Powershell with the Azure Powershell module loaded.

Listing 4.7 Operations AAD user and group

```
$SecureStringPassword = Read-Host -Prompt "Enter password" -AsSecureString   ❶

$User = New-AzADUser -DisplayName "Operations User"
<linearrow /> -Password $SecureStringPassword -MailNickname "operationsuser"   ❷
<linearrow /> -UserPrincipalName "operationsuser@azuredomain.onmicrosoft.com"   ❸

$Group = New-AzADGroup -DisplayName "Operations"
<linearrow /> -MailNickname "Ops"

Add-AzADGroupMember -MemberObjectId $User.Id   ❹
<linearrow /> -TargetGroupObjectId $Group.Id   ❺
```

❶ Prompt for a password for the new user.

❷ Use the secure password.

❸ Build the principle name from MailNickname and your AAD registered domain.

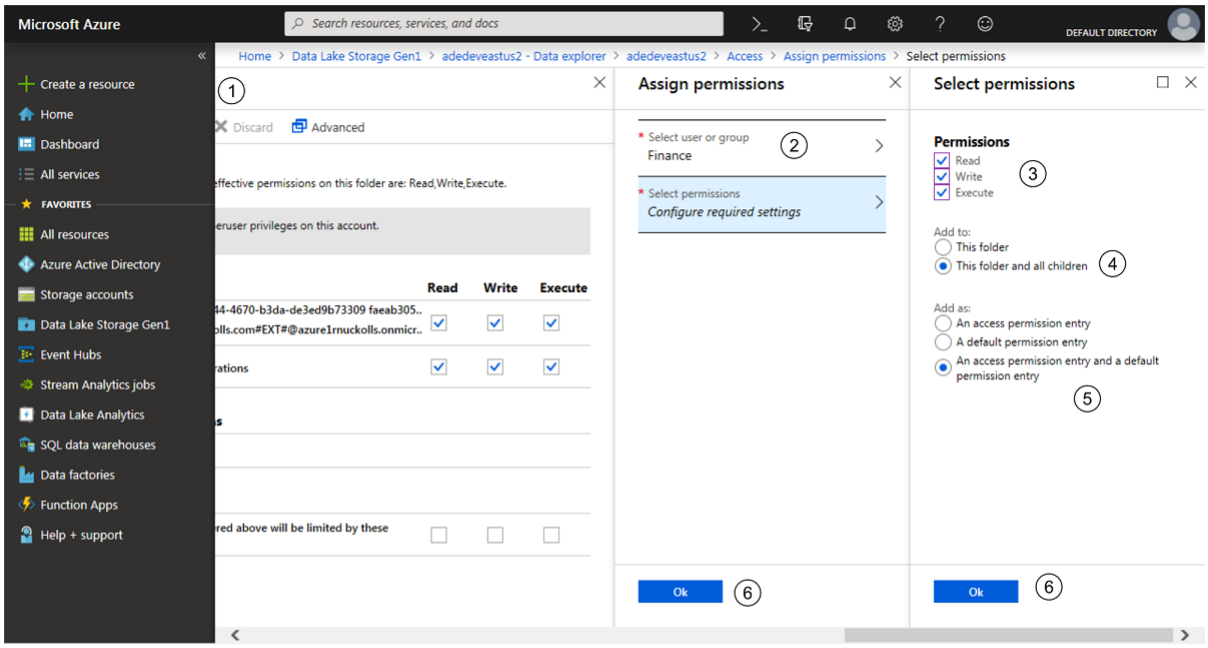
❹ Get the Id from the variable $User, from the new user command.

❺ Get the Id from the variable $Group, from the new group command.

Now that access to the Staging folder has been defined, create a "Finance" folder and "Operations" folder under Staging in the same way. For the "Finance" and "Operations" folders, assign a Read, Write, Execute (RWX) ACL to the Finance and Operations AAD groups respectively.

Figure 4.6 shows how to set the ACLs on a folder. Use the Azure Portal to create the folders and set the ACLs.

1. Create the department folder for Finance under Staging.
2. Click on the new "Finance" folder to browse the folder.
3. Click **Access** to show the Assign permissions blade, then click **Add** in the Assign permissions blade to configure the ACL.
4. In Select user or group, select the "Finance" group.
5. In Select permission, check Read, Write, Execute boxes under "Permissions.
6. Select "This folder and all children".
7. Select "An access permission entry and a default permission entry" under "Add as". This will set the permission on the folder, and as the inherited permission on any new files.
8. Click **Ok** to set the ACL on the "Finance" folder.
9. Repeat for "Operations".



1. Click "Add" on the Access blade. "Advanced" lets you modify child permissions.
2. Select a Azure Active Directory security group or user.
3. Choose Read and Execute for folder listing, or all three for full control.
4. You can cascade these permissions down to existing child files and folders.
5. Set permission on only existing files and folders (access), on new files and folders when created (default), or both.
6. Click "Ok" twice to commit the permissions.

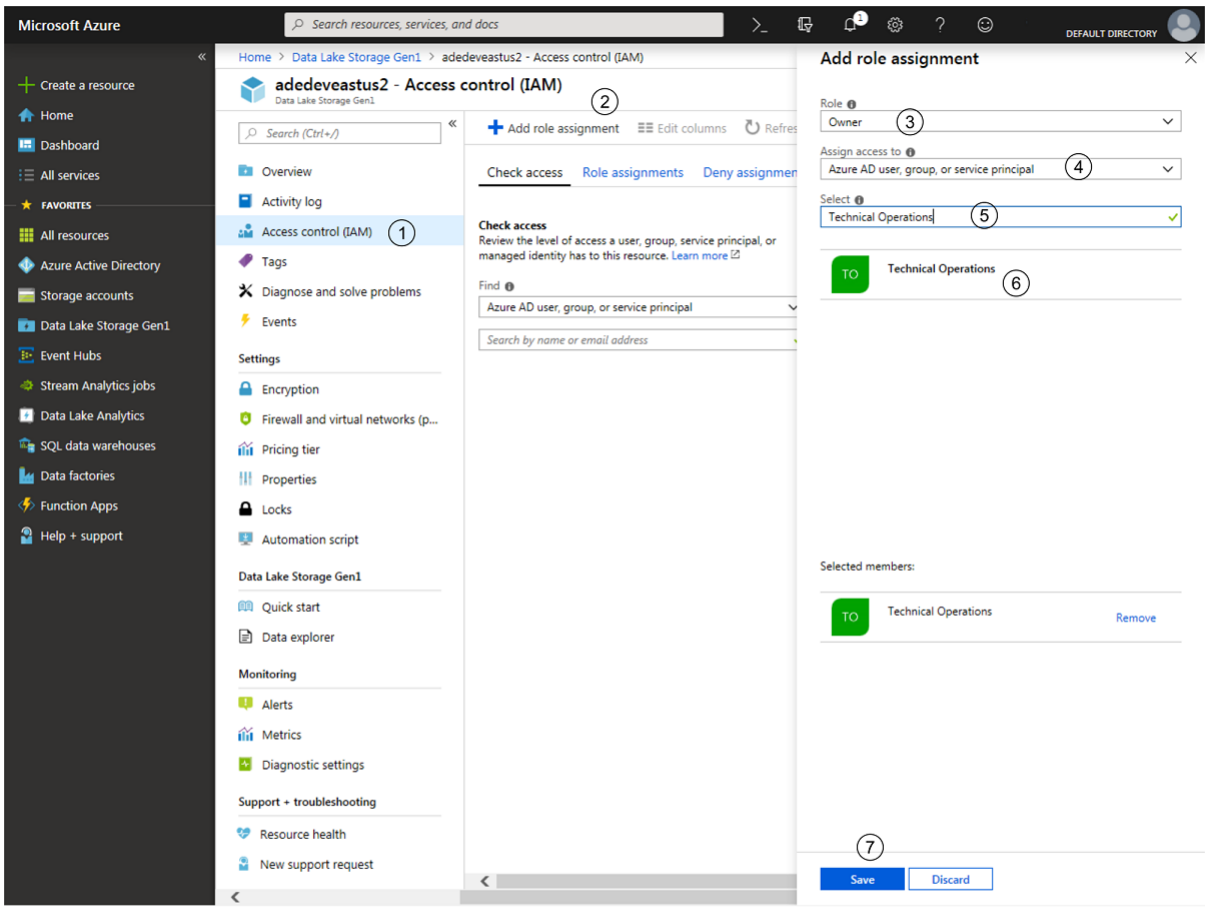
Figure 4.6 Assigning ACLs to Data Lake store folder

ADL SERVICE AUTHORIZATION

If your end users will only use command-line tools to copy files, then you can manage the ADL using ACL permissions only. If your end users use the Azure Portal, Azure PowerShell, or Azure Storage Explorer, then one more step is required before they have access. You need to give access to the ADL to any AAD user or group using the ADL via RBAC permissions. There are many built-in roles available. The Owner role has full control of the ADL. The Contributor role has full control of the ADL, except for assigning access via RBAC permissions. The Reader role allows read-only access to management data on the ADL, and access to interact with ADl storage via tools. You can read about the security controls on ADL at docs.microsoft.com/azure/data-lake-store/data-lake-store-security-overview. Every AAD user or group that needs to access the ADL must have at least the Reader role for the ADL. To complete the set up of the two departments folders, you need to assign the Reader role to the two AAD groups.

Use the Azure Portal to configure the roles.

1. Click **Access control (IAM)** in the Data Lake service blade to show the Access control blade.
2. Click **Add role assignment** button or the Add button within the "Add a role assignment" container to show the Add role assignment blade.
3. Select "Reader" for the Role.
4. Select the default "Azure AD user, group, or service principal" to Assign access to. Other options allow authorization by Azure services.
5. Click "Finance" group to select the group.
6. Click **Save** to add the role assignment.



1. Access control defines what you can do to the service , and is required for minimal access to the stored files.
2. Click "add role assignment to give new permissions.
3. Owners have full control, Contributors nearly as much, and Readers just read.
4. Use Azure AD groups for easiest management. Other options are Azure service-to-service authorization.
5. Start typing to filter the list.
6. Click on a security user or group to select.
7. Save your new role assignment.

Figure 4.7 Assigning AAD group to Data Lake service roles

The Data Lake service owner role gives access to all the content, as well as management functions for the Data Lake store. As an owner, you can assign other AAD users as owners too. These owners, and various other Data Lake service roles, are separate from the root, subfolder, and file owners.

4.3 Storage folder structure and data drift

In this section, you'll examine some of the challenges users face using the data sets stored in cloud storage, and review some approaches for data governance which will provide help to users. You'll also cover a methodology for clustering files of like schema, to mitigate the effects of data drift on subsequent analysis. To do this, you'll work through a scenario which involves coping with new fields in log files which are part of an existing process, which is structural drift. By the end of this section, you'll have a framework for structuring your storage which mitigates the impact of data drift and helps your users find the data they need.

4.3.1 Problem definition: IIS logging configuration is adding fields

In this scenario, log files for all the production IIS websites are shipped on a daily basis to cloud storage you manage. The files are stored in the IISLogs folder you set up in 4.1.2. The IIS logs currently store 15 fields. The next production release will add 4 more fields, for a total of 19. The filenames for the logfiles and their folders on the IIS servers will remain the same. You need to document this change so that users of the cloud storage can modify their analysis jobs. One way to document this change is by storing the files in the new format in a new folder.

4.3.2 Data drift

Data drift can be managed using folder structure and naming conventions. Segregation of differing files prevents breaking changes to existing analysis processes. Thoughtful naming conventions provide direction for finding the correct data sources and matching the import logic to the structure and schema. "Data drift exists in three forms: structural drift, semantic drift and infrastructure drift." (Pancha 2016) The set of fields contained in a data file can increase or decrease over time. This is structural drift. The content of each field can contain new values with the same meaning, or new meaning. This is semantic drift. And the systems generating, housing, or processing the data may change, leading to entirely different formats. This is infrastructure drift. Since computing systems change over time, data drift is a natural part of operating computer systems. Because later analysis of the data files must take into account these changes in the data itself, you should plan from the beginning ways to clearly identify the changes.

4.3.3 Hierarchy structure revisited

Earlier in this chapter you created folders at the root level of your storage to solve the immediate requirement without much structure or hierarchy beyond that necessitated by the underlying Azure service. Now you're going to look at an approach to structuring folders which provides a usage pattern for analysis. The first step in this approach was creating a Staging folder in the root folder of the Data Lake store. Now you'll take this folder construction further with the Data Lake store.

Documenting a storage area, especially a Data Lake, covers more than a set of notes to support it. Data files need attributes like source, type, quality, and date. Having these attributes helps users to find the data they need. You can provide these attributes at a basic level using a combination of folder structure and file naming conventions.

ZONES FRAMEWORK

Imagine splitting your Data Lake into multiple sections, or *zones*, based on the level of processing and/or transformation required. An initial zone would provide a target for data loading by external services and users. You created the Staging zone with the folder in the Data Lake store in the previous section. The next zone stores unprocessed or slightly processed files for long term access, without further modification. This is the Raw zone. The third zone allows storage of ad-hoc query output, as well as other files used when creating new data investigations. This is the Sandbox zone. The last zone stores production query output for business use. This is the Curated zone.

In a production ADL, analytics systems and automation systems handle data movement into and between zone. Sources of business data, like application logging, user behavior tracking, and IoT data, flow into the Staging zone. Azure services, like ADF and ADLA, copy and transform data before outputting files to the Raw zone. Analysts use tools like ADLA to generate data files in their Sandbox folder. Finally, analysts and data engineers use these same tools, like ADF and ADLA, to create final data sets in the Curated zone for end user queries. Figure 4.8 shows the layout of the zones framework with the flows of data between the zones.

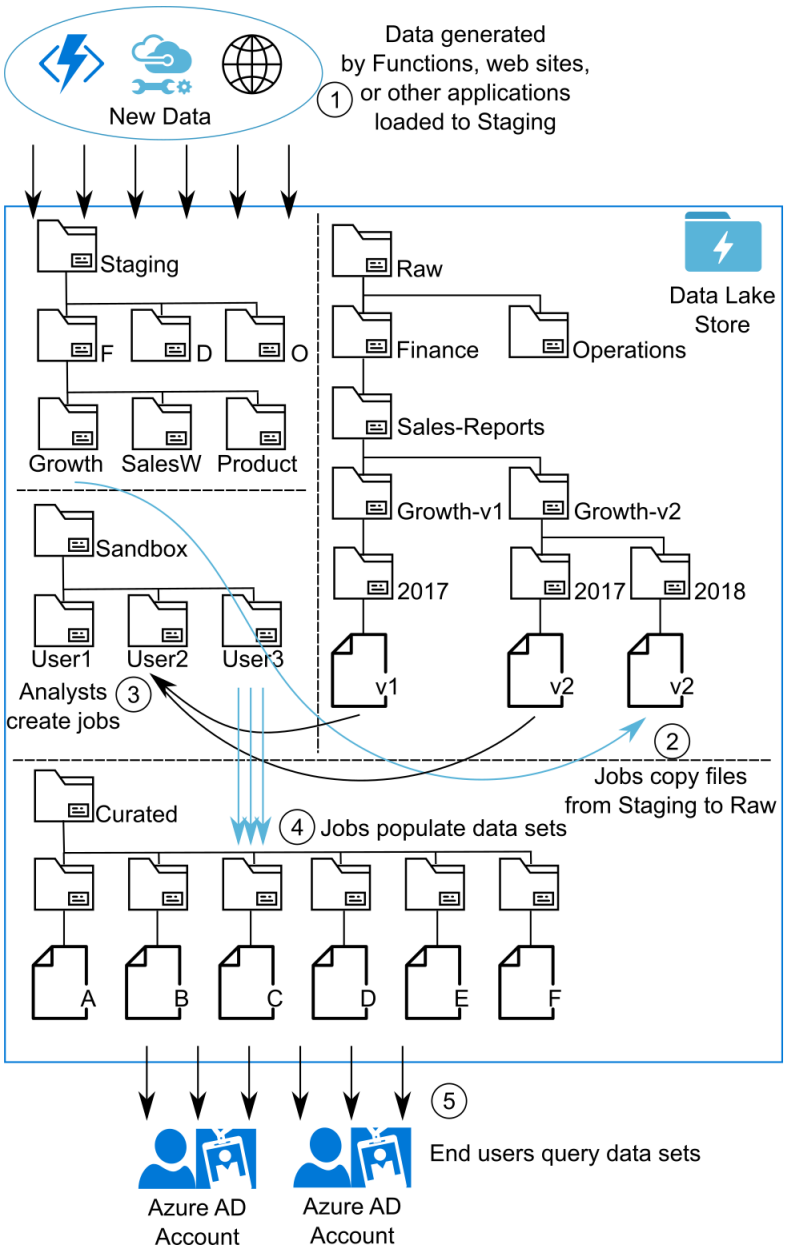


Figure 4.8 Folder structure with zones framework

NOTE You can read about creating and using ADLA jobs in chapter 7. You'll hear more about data movement in chapter 10 on Data Integration with ADF.

STAGING ZONE

Let's look at the Staging zone. This zone is for initial loading files in the Data Lake. Access to files and folders in the Staging zone should be limited to systems loading the data. Often these files need some type of processing before they can be used. Combining multiple small files into larger files for long term storage would happen with the Staging files. Distributed processing systems work most efficiently with fewer, larger files. Another example would be personally identifiable information (PII) cleansing. Files in the Staging zone should not be expected to remain long. Taking the design and structure from figure 4.8, here are a set of PowerShell commands to set up these folders for the zone framework. Execute these Azure PowerShell cmdlets using a PowerShell client with Azure PowerShell module loaded to set up the folders.

Listing 4.8 Set up Data Lake store Staging folder

```
New-AzDataLakeStoreItem -AccountName "adedeveastus2" -Path "/Staging" -Folder
New-AzDataLakeStoreItem -AccountName "adedeveastus2" -Path "/Staging/Finance" -Folder
New-AzDataLakeStoreItem -AccountName "adedeveastus2" -Path "/Staging/DevOps" -Folder
New-AzDataLakeStoreItem -AccountName "adedeveastus2" -Path "/Staging/Operations" -Folder
New-AzDataLakeStoreItem -AccountName "adedeveastus2" -Path "/Staging/Finance/Growth" -Folder
New-AzDataLakeStoreItem -AccountName "adedeveastus2" -Path "/Staging/Finance/SalesW" -Folder
New-AzDataLakeStoreItem -AccountName "adedeveastus2" -Path "/Staging/Finance/Product" -Folder
```

Remember to return and set ACL for access to folders below Staging, according to product or department ownership.

TIP ADLA performs best using splittable files between 250MB and 1GB. See docs.microsoft.com/azure/data-lake-store/data-lake-store-best-practices for more best practices.

RAW ZONE

Staging files are destined for the Raw folder. The Raw zone is where data files go to die. In the Raw zone, data files of all types and content wait for reading as part of an analytics job or other request. Files in the Raw zone should remain in their original state, without modification or updating. Access to files and folders in the Raw zone should be restricted to Read only access for data analysts. Collections of data files in the Raw zone can suffer from data drift. You'll examine the use of versioning to mitigate effect of data drift on analysis and processing jobs later in this section. Execute these Azure PowerShell cmdlets using a PowerShell client with Azure PowerShell module loaded to set up the folders.

Listing 4.9 Set up Data Lake store Raw folder

```
New-AzDataLakeStoreItem -AccountName "adedeveastus2" -Path "/Raw" -Folder
New-AzDataLakeStoreItem -AccountName "adedeveastus2"
                        -Path "/Raw/Finance" -Folder
New-AzDataLakeStoreItem -AccountName "adedeveastus2"
                        -Path "/Raw/Operations" -Folder
New-AzDataLakeStoreItem -AccountName "adedeveastus2"
                        -Path "/Raw/Finance/Sales-Reports" -Folder
New-AzDataLakeStoreItem -AccountName "adedeveastus2"
                        -Path "/Raw/Finance/Sales-Reports/Growth-v1" -Folder
New-AzDataLakeStoreItem -AccountName "adedeveastus2"
                        -Path "/Raw/Finance/Sales-Reports/Growth-v2" -Folder
New-AzDataLakeStoreItem -AccountName "adedeveastus2"
                        -Path "/Raw/Finance/Sales-Reports/Growth-v1/2017" -Folder
New-AzDataLakeStoreItem -AccountName "adedeveastus2"
                        -Path "/Raw/Finance/Sales-Reports/Growth-v2/2017" -Folder
New-AzDataLakeStoreItem -AccountName "adedeveastus2"
                        -Path "/Raw/Finance/Sales-Reports/Growth-v2/2018" -Folder
```

NOTE Many data exploration techniques rely on finding outliers. Cleaning and normalizing data at the Raw stage could remove valuable data. Consider storing even data rows which fail Staging processing in "Error" folders adjacent to the data files.

SANDBOX ZONE

The Sandbox zone gives data analysts an open area to process data files. It allows uploading of new data, and creating multiple versions of combined data files as new data products are developed. Processing routines can be developed using the Sandbox zone as a testing space. The data can be minimally or majorly processed in the Sandbox. Access to files and folders in the Sandbox zone should be unrestricted for each user. The Sandbox zone would not serve data to end users. Execute these Azure PowerShell cmdlets using a PowerShell client with Azure PowerShell module loaded to set up the folders.

Listing 4.10 Set up Data Lake store Sandbox folder

```
New-AzDataLakeStoreItem -AccountName "adedeveastus2" -Path "/Sandbox" -Folder
New-AzDataLakeStoreItem -AccountName "adedeveastus2" -Path "/Sandbox/User1" -Folder
New-AzDataLakeStoreItem -AccountName "adedeveastus2" -Path "/Sandbox/User2" -Folder
New-AzDataLakeStoreItem -AccountName "adedeveastus2" -Path "/Sandbox/User3" -Folder
```

You can skip setting up user folders in the Sandbox zone until you have user ready to use the system to do analysis. Remember to secure each user folder with appropriate security ACLs.

CURATED ZONE

The Curated zone holds output from analytics jobs run against data files in the Raw zone. The data contained has been processed, preparing it for use by end users. A common use case for files in the Curated zone would be exploration with visualization tools by business users. Access to files and folders in the Curated zone should be limited to Read only access for business users and tools, and write access for data analysts and jobs creating the data sets. Execute these Azure PowerShell cmdlets using a PowerShell client with Azure PowerShell module loaded to set up the folders.

Listing 4.11 Set up Data Lake store Curated folder

```
New-AzDataLakeStoreItem -AccountName "adedeveastus2" -Path "/Curated" -Folder
New-AzDataLakeStoreItem -AccountName "adedeveastus2" -Path "/Curated/FolderA" -Folder
New-AzDataLakeStoreItem -AccountName "adedeveastus2" -Path "/Curated/FolderB" -Folder
New-AzDataLakeStoreItem -AccountName "adedeveastus2" -Path "/Curated/FolderC" -Folder
```

These are the top-level zones that should be set up in the Data Lake store. These are community best practice, not a specific Microsoft recommendation. Figure 4.9 shows a hierarchy scheme using the zones approach.

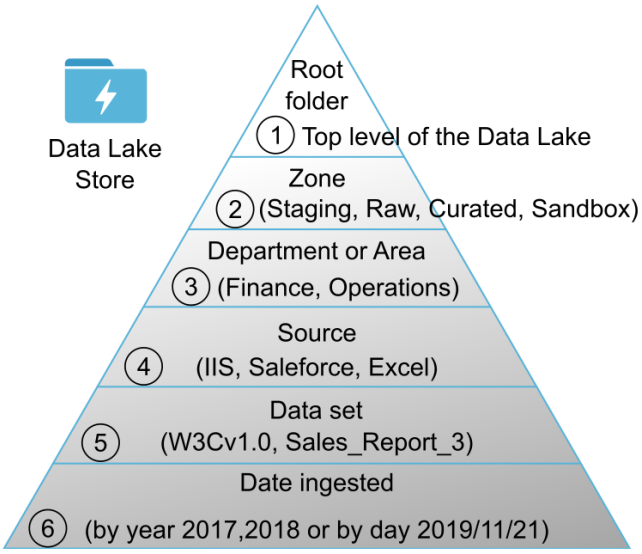


Figure 4.9 Azure Data Lake folder hierarchy

In each zone, data files are sorted by department, source, and type. Sorting often includes a date of ingestion, or loading, into the Data Lake store. The folder hierarchy of the Staging zone typically follows that of the Raw zone, but can vary depending on file aggregation or data drift controls. The Sandbox zone can be constructed using folders per user, rather than being broken up by department and source. Data sets in the Curated zone have a business case and frequently combine data from multiple sources. The Curated zone can be constructed using folders per business unit, project, or along security boundaries.

Later chapters will walk through the processes which move data between folders, create new data

files, and return data to the end user. When you put this folder hierarchy in place and enforce it with security controls, you will reduce the likelihood of your Data Lake becoming a Data Swamp.

REPLACING IILOGS FOLDER IN ZONES

Now that you have a framework for structuring our Data Lake store, let's review the state of the IISLogs folder in light of our scenario. The data files in IISLogs are TXT files in comma-separated format, with 15 columns. The column update will increase the column count to 19. This is structural drift within the files, and according to the scenario will not be obvious from the source folder or filename. This structural data drift could cause problems with analytics jobs, if a job attempts to read the new columns from files which don't contain the columns.

MITIGATING DATA DRIFT IN ZONES

Now consider a folder hierarchy using the zones framework. Operations doesn't deposit files directly in a top level IISLogs folder. Instead, they use the Staging zone folder, and use the IISLogs folder within. Files for analytics jobs no longer use the same IISLogs folder that Operations does. Analytics jobs read from folders in the Raw zone hierarchy. This hierarchy now includes:

- a folder level defining the originating department or area
- a folder level defining the source
- a folder level defining the data set
- a folder level defining a version of the file within

The folder naming convention matters less than the principle of segregation: an analytics job can read all the files in a single folder, and be guaranteed their schema match. This hierarchy adds description to the files, and versioning mitigates the effects of data drift. Figure 4.10 shows this folder structure laid out.

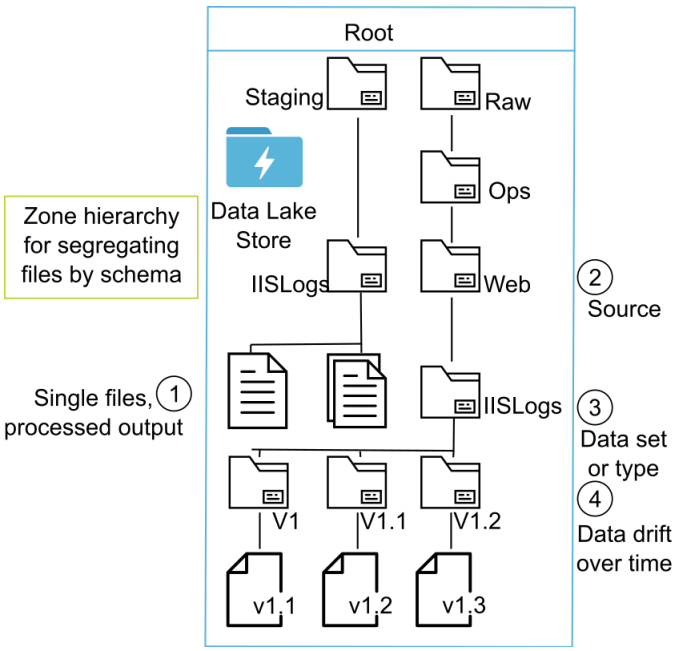


Figure 4.10 Folder structure with file versions

With this structure in place, Operations would not need to change their loading process, but would need to notify you of the change in schema. Otherwise, you may need to rely on failure notifications or other processes to detect the data drift. For your part, you will need to create the new folder for the new file version, and update any data movement and analytics jobs to add references to the new version.

The zones framework spreads out from the root and 4 zones to more numerous and more targeted folders. This organizes the files contained and conveys details about the files. Figure 4.11 describes the zones framework as a pyramid.

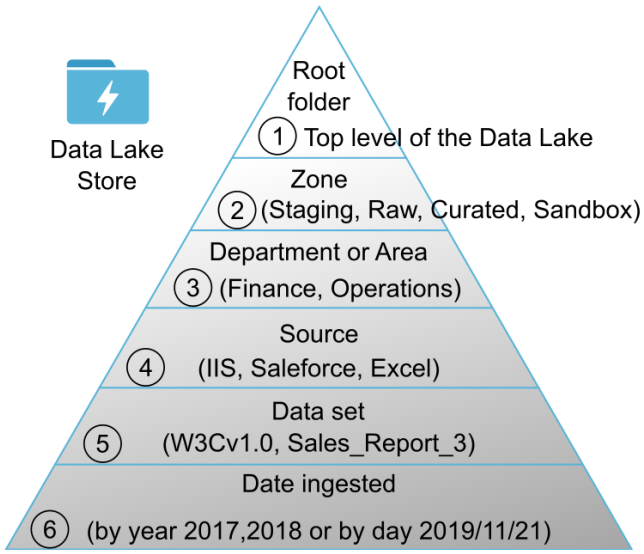


Figure 4.11 Azure Data Lake folder hierarchy

Because the likelihood of data drift increases with time, you should incorporate versioning into

your folder hierarchy plan. Versioning of the files allows analytics jobs to process files with the same schema in the same way. Versioning also allows analytics jobs to use different processes with differing schema.

With the zones framework, versioning by file folder works in any of the four top zones: Staging, Raw, Sandbox, or Curated. The zone structure provides flexibility around your implementation, especially at the lowest levels. Figure 4.11 shows a hierarchy scheme using an ingestion date as the lowest level. This works well in a slowly changing environment, with little data drift, and analysis bounded by date ranges. An ingestion date folder structure benefits from automated processing to create the folders and copy the files, especially when including month and day folders. Consider some other lower level folder variations.

- How granular are the data files? Will they be combined by week or month to improve efficiency? Try segregating by file version.
- How great is the volume? Will the files need to be divided into smaller files by day, hour, or minute? Try year, month, day folder structure.
- Does a single department generate a single format, like images or XML? Try segregated at the "Source" level by format, then by project set.
- How rapidly does the data drift? It’s easiest to modify the folder structure at the bottom than at the zone, department, or source level.

TIP Many data lakes collect data sources from third-parties. Consider adding a "Third_Party" or individual third-party folders at the "Department" level of the zones framework.

With the zones framework, you have a model for minimizing the impact of data drift in your Data Lake store. The framework also provides a method for managing the security structure of your Data Lake store. With these attributes in mind, you can create a Data Lake store which will serve your analytics system well.

4.4 Summary

In this chapter, you learned the following:

- ADL is a petabyte-scale storage service which provides a hierarchical folder structure over HDFS.
- AAD is used to secure files and folders in Azure Data Lake stores, which reduces management.
- Dividing the ADL into zones creates a structure necessary to control usage.
- Planning for data drift during creation of the ADL folders provides clear guidance for later changes.

A Set up of Azure resources through Powershell

This appendix covers

- Configuring Azure Powershell
- Adding an Azure Active Directory user and group
- Adding a Resource group
- Adding Azure services

This appendix covers setting up Azure resources for the examples in the book. These Powershell (PS) scripts can help you quickly get your Azure environment setup for use with chapter scripts. PS scripts in this book have been developed using Az Powershell version 1.5.0. See docs.microsoft.com/powershell/module/az.resources for a full command reference. You should already be familiar with the Azure Portal. This book uses the Portal to setup services, as this GUI makes discovering the many options available easy.

Some features in Azure cannot be configured through the Portal, and some that can be configured only through the Portal. Powershell is a powerful tool, and the examples in this book only scratch the surface of what's possible. *Learn Windows PowerShell in a Month of Lunches, Third Edition* by Jones and Hicks provides a much deeper view of the technology.

NOTE If you don't have the latest Azure Powershell module installed on your computing device, Azure offers Cloud Shell. You can access Cloud Shell from the Azure portal, or by connecting to shell.azure.com. First time Cloud Shell set up requires a Resource group and Storage account. You can create a new group and Storage account right from Cloud Shell, specifically for Cloud Shell use. Azure Cloud Shell is tied to a user account, so you will have access to all your subscriptions, but you will need to select a subscription to store the settings and local scripts used in command window.

A.1 Setup Azure Powershell

If you already have a working PS setup with Azure Powershell, or want to use Cloud Shell, you can skip to the next section.

Getting Azure PS installed on your machine may be complicated. Different versions of Windows com with different versions of PS. Azure Powershell requires PS 5.x and .NET 4.7.2 on Windows. Here are a few steps that can help.

- Run cmd.exe as Administrator.
- Run the following cmdlets to install the Azure PS package.

```
powershell.exe -ExecutionPolicy Unrestricted
```

This will start PS and allows the installation of packages.

```
Install-Module -Name Az -AllowClobber
```

Install-Module cmdlet installs PS modules. This line will install the Azure Powershell module, updating earlier cmdlets if present using the -AllowClobber switch.

```
Import-Module Az
```

This cmdlet loads Azure Powershell into memory for use.

```
Connect-AzAccount
```

This cmdlet generates a token for connecting the PS session to Azure. Follow the instructions on-screen, going to microsoft.com/devicelogin to enter your token.

You can modify a Windows shortcut for Powershell to pass parameter "-ExecutionPolicy Unrestricted", and run these PS script in a PS window. See docs.microsoft.com/powershell/azure/install-az-ps?view=azps-1.0.0 for more information.

Now that you have a working PS environment, you can use PS and the Azure Powershell

cmdlets to create and modify services in Azure. In the next section, you'll setup the basic resources needed for working with Azure services and completing the examples and exercises in this book. Setup these basic resources before moving on to creating any other Azure services.

A.2 Create a subscription

In order to follow along with the examples in the book, you'll need an Azure subscription. Signing up for a personal account and subscription takes an email address, a phone number, and a credit card. Visit azure.microsoft.com/free/ to start your free trial. All of the Azure services described in this book are available during the trial period.

A.3 Azure naming conventions

Every Azure service, also called a resource, must have a name. Consistently applying a considered naming convention helps users find services, and identify ownership and usage of services. Since some resource name require lowercase characters, and most resources are addressable via URL, make your names lowercase from the start. A working naming convention incorporates several aspects of managing cloud resources:

- Project, system, or owning organization
- Environment or deployment stage, such as Development, Staging, or Production
- Region, or lack thereof
- Supported function, such as web, tools, api, batch, daily or other description
- Multiple instances, using a numeric suffix

This book uses a naming convention with these elements where necessary.

Listing A.1 Naming convention

```
[PROJECT]-[ENVIRO]-[REGION]-[FUNCTION]-[SUBFUNCTION]-[NN]
```

For an Azure SQL Data Warehouse, this yields ade-dev-eastus2-sqldw-baseball. There will be one SQL Data Warehouse for development, in the East US 2 region, for "Azure Data Engineering" (ADE), supporting baseball statistics analysis. You can read about Microsoft's recommendations for resource naming at docs.microsoft.com/azure/architecture/best-practices/naming-conventions.

A.4 Setup common Azure resources using Powershell

In this section, you'll learn how to setup some of the common resources in Azure using Powershell. You'll need to have access to an instance of PS with the Azure Powershell module loaded to run these PS scripts. This can be a local PS install, or you can use Azure Cloud Shell shell.azure.com. You'll create a resource group which you can use for all the examples and exercises. You'll create an Azure Active Directory user and an Azure Active Directory security group, which you can use to test security on Azure resources with accounts different than your primary owner account. These basic resource types are useful throughout your Azure service use.

You can get the latest version of these scripts from the GitHub site for this book github.com/rnuckolls/azure_data_engineering/tree/master/appendixa/powershell.

A.4.1 Create a new resource group

Resource groups in Azure are organizing containers. Every Azure service has one. Every resource group has a region. The resource group anchors a service to a region, with the primary configuration data for the service stored in that region. This is especially true for some services, like Cosmos DB and Traffic Manager, which are global services and have infrastructure in every region. Deleting a resource group deletes all the services attached to it. This book uses "ade-dev-eastus2" as the resource group for any PS scripts which require it. You should choose a naming convention which make sense for your situation. This book uses the "East US 2" region for any PS scripts which require it, since all resources in the book are available in the region.

Listing A.2 New resource group

```
New-AzResourceGroup -Name "ade-dev-eastus2" -Location "East US 2"
```

Execute this line in Powershell with the Azure Modules loaded. This PS script will return an error if a group by that name exists. Otherwise it will create a new resource group. There are many regions, or locations, for hosting Azure resources across the globe, including the Americas, Europe, Asia Pacific, and the Middle East and Africa. You can see the current list of services by region at Microsoft's Azure website azure.microsoft.com/global-infrastructure/services/.

A.4.2 Create new Azure Active Directory user

Azure uses Azure Active Directory (AAD) extensively for user and service authentication. Setup, examples, and exercises in this book make use of AAD users and groups. You should already be familiar with the Azure Portal, and using Active Directory for authentication and authorization to Azure services.

Listing A.3 New AAD User

```
$SecureStringPassword = ConvertTo-SecureString -String "Password1!" -AsPlainText -Force
$User = New-AzADUser -DisplayName "Tech User" -UserPrincipalName "techuser@domain.onmicrosoft.com"
                     -Password $SecureStringPassword -MailNickname "techuser"
```

Execute these lines in Powershell with the Azure Powershell module loaded. This PS script will return an error if a group by that name exists. The first line creates a Password object. Don't use Password1! The second will create a new user "techuser". You need to construct a UserPrincipalName using one of the AAD registered domains.

> **NOTE** If you are using an Azure subscription without a corporate Active Directory, then your domain will be some variation of the email you used to sign up with Azure. You can find this value by going to the Azure Active Directory service "Overview" blade. The domain is listed above the header "Default Directory". The domain is also listed in the "Custom domain names" blade.

A.4.3 Create new Azure Active Directory group

Now create a new security group for "Technical Operations", and add "techuser" to this group. Execute these lines in Powershell with the Azure Powershell module loaded. This PS script will return an error if a group by that name exists.

Listing A.4 New AAD Group

```
$Group = New-AzADGroup -DisplayName "Technical Operations" -MailNickname "TechOps"
Add-AzADGroupMember -MemberObjectId $User.Id -TargetGroupObjectId $Group.Id
```

Now you've setup the basic resources in Azure using Powershell. You've learned why you need a resource group in Azure, and how to create one. You've created a user and a security group in AAD. Now you can use them when securing other resources in Azure. In the next section you'll create a storage account.

A.5 Setup Azure services using Powershell

A.5.1 Create new Storage account

Now you'll create a Storage account using Azure Powershell. The Storage account cmdlet needs four pieces of information.

1. Resource group created at New resource group section
2. Name, referencing the same values from the resource group
3. Replication SKU, for the default "Read access geo-redundant storage"
4. Location, the same region as the resource group

Execute the single line of code 3.1 in Azure Powershell. This PS script will return an error if a Storage account by that name exists.

Listing A.5 Create new Storage account

```
New-AzStorageAccount -ResourceGroupName "ade-dev-eastus2" -AccountName "adedeveastus2" `   ❶
-SkuName Standard_RAGRS -Location "East US 2" `   ❷
-EnableHttpsTrafficOnly 1 -Kind "StorageV2"   ❸
```

❶ Account name must be alphanumeric

❷ Choose RAGRS for maximum redundancy, LRS for minimal redundancy

❸ Allowing only HTTPS traffic increases security

There are multiple values for the Replication SKU, which we cover in Chapter 3. The default value includes the most redundancy. Other values are less expensive, but have less redundancy.

There are multiple values for the Kind command.

- "Storage", which includes Blobs, Tables, Queues, Files and Disks services
- "StorageV2", the default, which includes Blobs, Tables, Queues, Files and Disks services and adds Hot/Cold/Archive tiered storage
- BlobStorage, which only support the Blobs service

Now you've learned how to create a Storage account using Azure Powershell.

A.5.2 Create new Data Lake store

Creating a Data Lake store works the same as most Azure resources. You need to choose the New- Azure Powershell cmdlet for the resource itself, then provide the basic properties of name, resource group, and location. The Data Lake store cmdlet takes these three pieces of information.

1. Resource group created at New resource group section
2. Name, referencing the same values from the resource group, alphanumeric only
3. Location, the same region as the resource group

Execute the single line of code 4.1 in Azure Powershell.

Listing A.6 Create new Data Lake store

```
New-AzDataLakeStoreAccount -ResourceGroupName "ade-dev-eastus2" -Name "adedeveastus2"
                           -Location "East US 2"
```

This PS script will return an error if a Data Lake store by that name exists, or the service is not available in the selected region. Keep resources within the same region. Data Lake store has the fewest available regions. It's a good idea to select one of these regions before creating the rest of your resource in Azure, so that you keep all resources in the same region. This lowers the latency of network communication between resources.

You can also specify some other options during setup. You can add key/value pairs, called Tags, to Azure resources to help locate them later. If you know your storage size, you can pre-purchase storage at a discounted rate using the -Tier parameter.

Listing A.7 Create new Data Lake store with options

```
New-AzDataLakeStoreAccount -ResourceGroupName "ade-dev-eastus2" -Name "adedeveastus2"
                           -Location "East US 2" -Tag @{User="ADE";}
                           -Tier Commitment1TB -DefaultGroup
                                (Get-AzADGroup -DisplayName "Technical Operations").Id
```

Default security group will set the owning group for your Root folder and other folders in the Data Lake store. Add a Tag for management of resources. This is especially nice when browsing the "All resources" blade in the portal, because you can select from a list of all the tags you have provided. Use a consumption plan, until you calculated your monthly storage needs. After you find your usage, sign up for recurring billing of the committed storage.

A.6 Summary

In this chapter, you learned:

- Powershell is a powerful tool for creating and configuring Azure services, using the Azure Powershell module.
- You can create basic resources like user accounts and Resource Groups in Azure using Powershell.
- Setting up and configuring Azure Storage can be done with Azure Powershell.
- Setting up and configuring Azure Data Lake storage can be done with Azure Powershell.