

# Table of Contents

[Azure Architecture Center](#)

[Data Architecture Guide](#)

[Traditional RDBMS](#)

[Concepts](#)

[Scenarios](#)

[Technology choices](#)

[Big Data and NoSQL](#)

[Concepts](#)

[Scenarios](#)

[Technology choices](#)

[Cross-cutting concerns](#)

[Data transfer](#)

[Extending on-premises data solutions to the cloud](#)

[Securing data solutions](#)

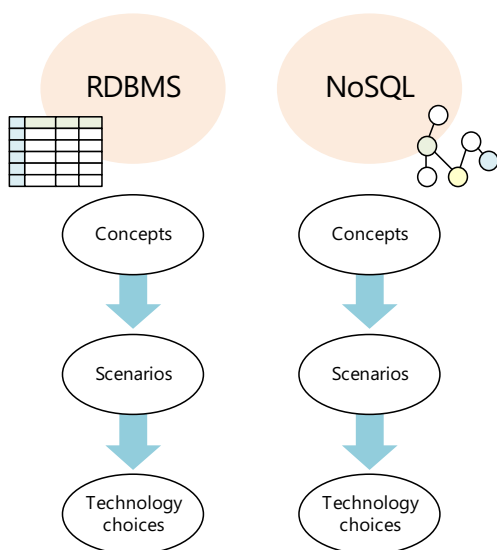
This guide presents a structured approach for designing data-centric solutions on Microsoft Azure. It is based on proven practices derived from customer engagements.

## Introduction

The cloud is changing the way applications are designed, including how data is processed and stored. Instead of a single general-purpose database that handles all of a solution's data, *polyglot persistence* solutions use multiple, specialized data stores, each optimized to provide specific capabilities. The perspective on data in the solution changes as a result. There are no longer multiple layers of business logic that read and write to a single data layer. Instead, solutions are designed around a *data pipeline* that describes how data flows through a solution, where it is processed, where it is stored, and how it is consumed by the next component in the pipeline.

## How this guide is structured

This guide is structured around a basic pivot: The distinction between *relational* data and *non-relational* data.



Relational data is generally stored in a traditional RDBMS or a data warehouse. It has a pre-defined schema ("schema on write") with a set of constraints to maintain referential integrity. Most relational databases use Structured Query Language (SQL) for querying. Solutions that use relational databases include online transaction processing (OLTP) and online analytical processing (OLAP).

Non-relational data is any data that does not use the [relational model](#) found in traditional RDBMS systems. This may include key-value data, JSON data, graph data, time series data, and other data types. The term *NoSQL* refers to databases that are designed to hold various types of non-relational data. However, the term is not entirely accurate, because many non-relational data stores support SQL compatible queries. Non-relational data and NoSQL databases often come up in discussions of *big data* solutions. 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.

Within each of these two main categories, the Data Architecture Guide contains the following sections:

- **Concepts.** Overview articles that introduce the main concepts you need to understand when working with this type of data.
- **Scenarios.** A representative set of data scenarios, including a discussion of the relevant Azure services and the appropriate architecture for the scenario.
- **Technology choices.** Detailed comparisons of various data technologies available on Azure, including open source options. Within each category, we describe the key selection criteria and a capability matrix, to help you choose the right technology for your scenario.

This guide is not intended to teach you data science or database theory — you can find entire books on those subjects. Instead, the goal is to help you select the right data architecture or data pipeline for your scenario, and then select the Azure services and

technologies that best fit your requirements. If you already have an architecture in mind, you can skip directly to the technology choices.

## Traditional RDBMS

### Concepts

- [Relational data](#)
- [Transactional data](#)
- [Semantic modeling](#)

### Scenarios

- [Online analytical processing \(OLAP\)](#)
- [Online transaction processing \(OLTP\)](#)
- [Data warehousing and data marts](#)
- [ETL](#)

## Big data and NoSQL

### Concepts

- [Non-relational data stores](#)
- [Working with CSV and JSON files](#)
- [Big data architectures](#)
- [Advanced analytics](#)
- [Machine learning at scale](#)

### Scenarios

- [Batch processing](#)
- [Real time processing](#)
- [Free-form text search](#)
- [Interactive data exploration](#)
- [Natural language processing](#)
- [Time series solutions](#)

## Cross-cutting concerns

- [Data transfer](#)
- [Extending on-premises data solutions to the cloud](#)
- [Securing data solutions](#)

# Relational data

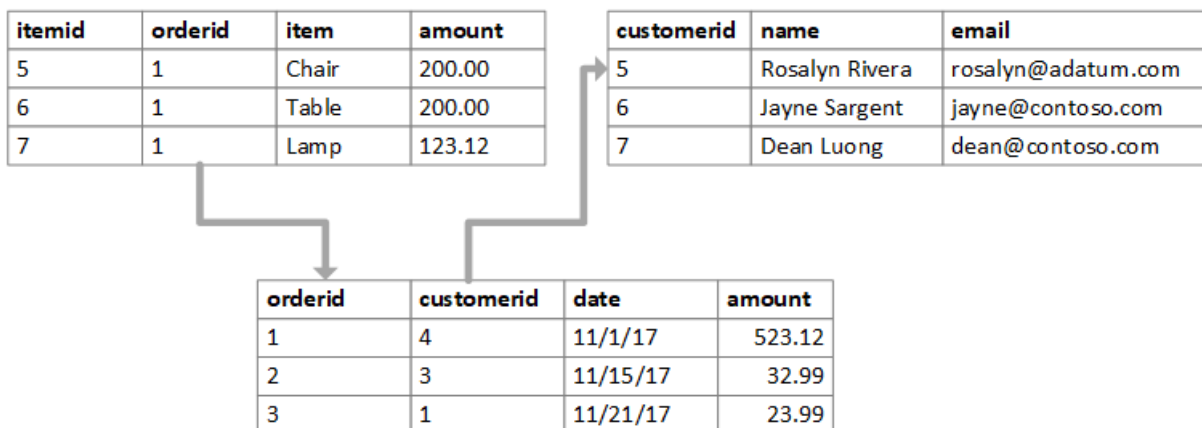
2/13/2018 • 2 min to read • [Edit Online](#)

Relational data is data modeled using the relational model. In this model, data is expressed as tuples. A *tuple* is a set of attribute/value pairs. For example, a tuple might be (itemid = 5, orderid = 1, item = "Chair", amount = 200.00). A set of tuples that all share the same attributes is called a *relation*.

Relations are naturally represented as tables, where each tuple is exposed as a row in the table. However, rows have an explicit ordering, unlike tuples. The database schema defines the columns (headings) of each table. Each column is defined with a name and a data type for all values stored in that column across all rows in the table.

itemid	orderid	item	amount
5	1	Chair	200.00
6	1	Table	200.00
7	1	Lamp	123.12

A data store that organizes data using the relational model is referred to as a relational database. Primary keys uniquely identify rows within a table. Foreign key fields are used in one table to refer to a row in another table by referencing the primary key of the other table. Foreign keys are used to maintain referential integrity, ensuring that the referenced rows are not altered or deleted while the referencing row depends on them.



Relational databases support various types of constraints that help to ensure data integrity:

- Unique constraints ensure that all values in a column are unique.
- Foreign key constraints enforce a link between the data in two tables. A foreign key references the primary key or another unique key from another table. A foreign key constraint enforces referential integrity, disallowing changes that cause invalid foreign key values.
- Check constraints, also known as entity integrity constraints, limit the values that can be stored within a single column, or in relationship to values in other columns of the same row.

Most relational databases use the Structured Query Language (SQL) language that enables a declarative approach to querying. The query describes the desired result, but not the steps to execute the query. The engine then decides the best way to execute the query. This differs from a procedural approach, where the query program specifies the processing steps explicitly. However, relational databases can store executable code routines in the form of stored procedures and functions, which enables a mixture of declarative and procedural approaches.

To improve query performance, relational databases use *indexes*. Primary indexes, which are used by the primary key, define the order of the data as it sits on disk. Secondary indexes provide an alternative combination of fields,

so the desired rows can be queried efficiently, without having to re-sort the entire data on disk.

Because relational databases enforce referential integrity, scaling a relational database can become challenging. That's because any query or insert operation might touch any number of tables. You can scale out a relational database by *sharding* the data, but this requires careful design of the schema. For more information, see [Sharding pattern](#).

If data is non-relational or has requirements that are not suited to a relational database, consider a [Non-relational or NoSQL](#) data store.

# Transactional data

3/4/2018 • 1 min to read • [Edit Online](#)

Transactional data is information that tracks the interactions related to an organization's activities. These interactions are typically business transactions, such as payments received from customers, payments made to suppliers, products moving through inventory, orders taken, or services delivered. Transactional events, which represent the transactions themselves, typically contain a time dimension, some numerical values, and references to other data.

Transactions typically need to be *atomic* and *consistent*. Atomicity means that an entire transaction always succeeds or fails as one unit of work, and is never left in a half-completed state. If a transaction cannot be completed, the database system must roll back any steps that were already done as part of that transaction. In a traditional RDBMS, this rollback happens automatically if a transaction cannot be completed. Consistency means that transactions always leave the data in a valid state. (These are very informal descriptions of atomicity and consistency. There are more formal definitions of these properties, such as [ACID](#).)

Transactional databases can support strong consistency for transactions using various locking strategies, such as pessimistic locking, to ensure that all data is strongly consistent within the context of the enterprise, for all users and processes.

The most common deployment architecture that uses transactional data is the data store tier in a 3-tier architecture. A 3-tier architecture typically consists of a presentation tier, business logic tier, and data store tier. A related deployment architecture is the [N-tier](#) architecture, which may have multiple middle-tiers handling business logic.



## Typical traits of transactional data

Transactional data tends to have the following traits:

REQUIREMENT	DESCRIPTION
Normalization	Highly normalized
Schema	Schema on write, strongly enforced
Consistency	Strong consistency, ACID guarantees
Integrity	High integrity
Uses transactions	Yes
Locking strategy	Pessimistic or optimistic
Updateable	Yes

REQUIREMENT	DESCRIPTION
Appendable	Yes
Workload	Heavy writes, moderate reads
Indexing	Primary and secondary indexes
Datum size	Small to medium sized
Model	Relational
Data shape	Tabular
Query flexibility	Highly flexible
Scale	Small (MBs) to Large (a few TBs)

## See Also

[Online Transaction Processing](#)

# Semantic modeling

2/27/2018 • 2 min to read • [Edit Online](#)

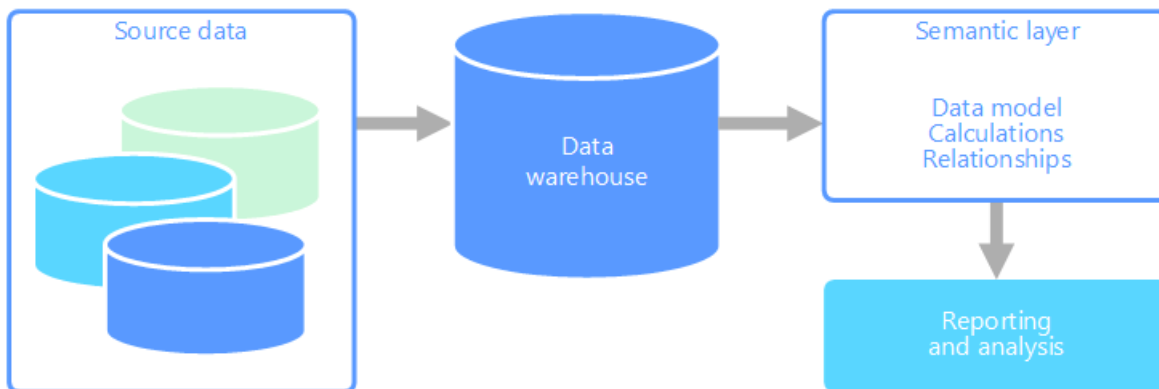
A semantic data model is a conceptual model that describes the meaning of the data elements it contains. Organizations often have their own terms for things, sometimes with synonyms, or even different meanings for the same term. For example, an inventory database might track a piece of equipment with an asset ID and a serial number, but a sales database might refer to the serial number as the asset ID. There is no simple way to relate these values without a model that describes the relationship.

Semantic modeling provides a level of abstraction over the database schema, so that users don't need to know the underlying data structures. This makes it easier for end users to query data without performing aggregates and joins over the underlying schema. Also, usually columns are renamed to more user-friendly names, so that the context and meaning of the data are more obvious.

Semantic modeling is predominately used for read-heavy scenarios, such as analytics and business intelligence (OLAP), as opposed to more write-heavy transactional data processing (OLTP). This is mostly due to the nature of a typical semantic layer:

- Aggregation behaviors are set so that reporting tools display them properly.
- Business logic and calculations are defined.
- Time-oriented calculations are included.
- Data is often integrated from multiple sources.

Traditionally, the semantic layer is placed over a data warehouse for these reasons.



There are two primary types of semantic models:

- **Tabular.** Uses relational modeling constructs (model, tables, columns). Internally, metadata is inherited from OLAP modeling constructs (cubes, dimensions, measures). Code and script use OLAP metadata.
- **Multidimensional.** Uses traditional OLAP modeling constructs (cubes, dimensions, measures).

Relevant Azure service:

- [Azure Analysis Services](#)

## Example use case

An organization has data stored in a large database. It wants to make this data available to business users and customers to create their own reports and do some analysis. One option is just to give those users direct access to the database. However, there are several drawbacks to doing this, including managing security and controlling access. Also, the design of the database, including the names of tables and columns, may be hard for a user to



understand. Users would need to know which tables to query, how those tables should be joined, and other business logic that must be applied to get the correct results. Users would also need to know a query language like SQL even to get started. Typically this leads to multiple users reporting the same metrics but with different results.

Another option is to encapsulate all of the information that users need into a semantic model. The semantic model can be more easily queried by users with a reporting tool of their choice. The data provided by the semantic model is pulled from a data warehouse, ensuring that all users see a single version of the truth. The semantic model also provides friendly table and column names, relationships between tables, descriptions, calculations, and row-level security.

## Typical traits of semantic modeling

Semantic modeling and analytical processing tends to have the following traits:

REQUIREMENT	DESCRIPTION
Schema	Schema on write, strongly enforced
Uses Transactions	No
Locking Strategy	None
Updateable	No (typically requires recomputing cube)
Appendable	No (typically requires recomputing cube)
Workload	Heavy reads, read-only
Indexing	Multidimensional indexing
Datum size	Small to medium sized
Model	Multidimensional
Data shape:	Cube or star/snowflake schema
Query flexibility	Highly flexible
Scale:	Large (10s-100s GBs)

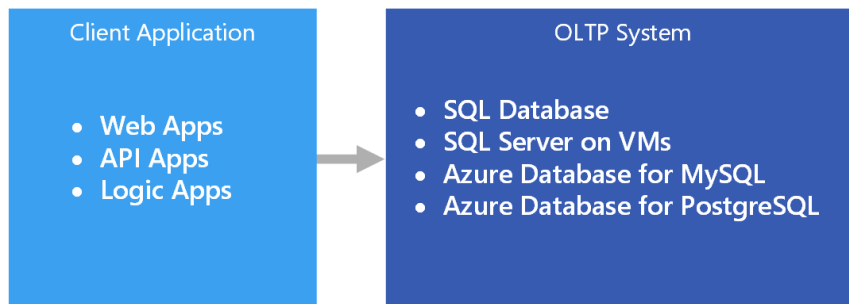
## See also

- [Data warehousing](#)
- [Online analytical processing \(OLAP\)](#)

# Online transaction processing (OLTP)

2/13/2018 • 2 min to read • [Edit Online](#)

The management of [transactional data](#) using computer systems is referred to as Online Transaction Processing (OLTP). OLTP systems record business interactions as they occur in the day-to-day operation of the organization, and support querying of this data to make inferences.



## When to use this solution

Choose OLTP when you need to efficiently process and store business transactions and immediately make them available to client applications in a consistent way. Use this architecture when any tangible delay in processing would have a negative impact on the day-to-day operations of the business.

OLTP systems are designed to efficiently process and store transactions, as well as query transactional data. The goal of efficiently processing and storing individual transactions by an OLTP system is partly accomplished by data normalization — that is, breaking the data up into smaller chunks that are less redundant. This supports efficiency because it enables the OLTP system to process large numbers of transactions independently, and avoids extra processing needed to maintain data integrity in the presence of redundant data.

## Challenges

Implementing and using an OLTP system can create a few challenges:

- OLTP systems are not always good for handling aggregates over large amounts of data, although there are exceptions, such as a well-planned SQL Server-based solution. Analytics against the data, that rely on aggregate calculations over millions of individual transactions, are very resource intensive for an OLTP system. They can be slow to execute and can cause a slow-down by blocking other transactions in the database.
- When conducting analytics and reporting on data that is highly normalized, the queries tend to be complex, because most queries need to de-normalize the data by using joins. Also, naming conventions for database objects in OLTP systems tend to be terse and succinct. The increased normalization coupled with terse naming conventions makes OLTP systems difficult for business users to query, without the help of a DBA or data developer.
- Storing the history of transactions indefinitely and storing too much data in any one table can lead to slow query performance, depending on the number of transactions stored. The common solution is to maintain a relevant window of time (such as the current fiscal year) in the OLTP system and offload historical data to other systems, such as a data mart or [data warehouse](#).

## OLTP in Azure

Applications such as websites hosted in [App Service Web Apps](#), REST APIs running in App Service, or mobile or desktop applications communicate with the OLTP system, typically via a REST API intermediary.

In practice, most workloads are not purely OLTP. There tends to be an [analytical component](#) as well. In addition, there is an increasing demand for real-time reporting, such as running reports against the operational system. This is also referred to as HTAP (Hybrid Transactional and Analytical Processing). For more information, see [Online Analytical Processing \(OLAP\) data stores](#).

## Technology choices

Data storage:

- [Azure SQL Database](#)
- [SQL Server in an Azure VM](#)
- [Azure Database for MySQL](#)
- [Azure Database for PostgreSQL](#)

For more information, see [Choosing an OLTP data store](#)

Data sources:

- [App service](#)
- [Mobile Apps](#)

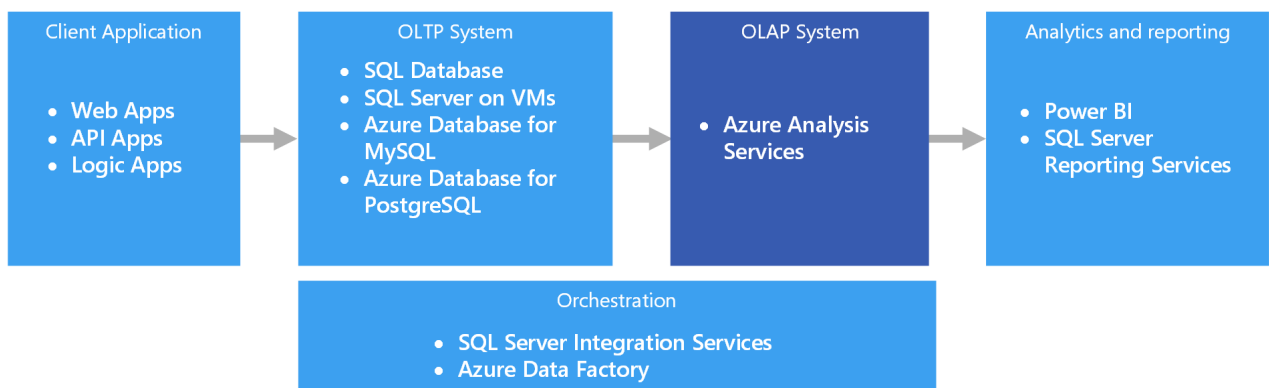
# Online analytical processing (OLAP)

2/13/2018 • 2 min to read • [Edit Online](#)

Online analytical processing (OLAP) is a technology that organizes large business databases and supports complex analysis. It can be used to perform complex analytical queries without negatively affecting transactional systems.

The databases that a business uses to store all its transactions and records are called [online transaction processing \(OLTP\)](#) databases. These databases usually have records that are entered one at a time. Often they contain a great deal of information that is valuable to the organization. The databases that are used for OLTP, however, were not designed for analysis. Therefore, retrieving answers from these databases is costly in terms of time and effort.

OLAP systems were designed to help extract this business intelligence information from the data in a highly performant way. This is because OLAP databases are optimized for heavy read, low write workloads.



## When to use this solution

Consider OLAP in the following scenarios:

- You need to execute complex analytical and ad hoc queries rapidly, without negatively affecting your OLTP systems.
- You want to provide business users with a simple way to generate reports from your data
- You want to provide a number of aggregations that will allow users to get fast, consistent results.

OLAP is especially useful for applying aggregate calculations over large amounts of data. OLAP systems are optimized for read-heavy scenarios, such as analytics and business intelligence. OLAP allows users to segment multi-dimensional data into slices that can be viewed in two dimensions (such as a pivot table) or filter the data by specific values. This process is sometimes called "slicing and dicing" the data, and can be done regardless of whether the data is partitioned across several data sources. This helps users to find trends, spot patterns, and explore the data without having to know the details of traditional data analysis.

[Semantic models](#) can help business users abstract relationship complexities and make it easier to analyze data quickly.

## Challenges

For all the benefits OLAP systems provide, they do produce a few challenges:

- Whereas data in OLTP systems is constantly updated through transactions flowing in from various sources, OLAP data stores are typically refreshed at a much slower intervals, depending on business needs. This means OLAP systems are better suited for strategic business decisions, rather than immediate responses to changes. Also, some level of data cleansing and orchestration needs to be planned to keep the OLAP data stores up-to-

date.

- Unlike traditional, normalized, relational tables found in OLTP systems, OLAP data models tend to be multidimensional. This makes it difficult or impossible to directly map to entity-relationship or object-oriented models, where each attribute is mapped to one column. Instead, OLAP systems typically use a star or snowflake schema in place of traditional normalization.

## OLAP in Azure

In Azure, data held in OLTP systems such as Azure SQL Database is copied into the OLAP system, such as [Azure Analysis Services](#). Data exploration and visualization tools like [Power BI](#), Excel, and third-party options connect to Analysis Services servers and provide users with highly interactive and visually rich insights into the modeled data. The flow of data from OLTP data to OLAP is typically orchestrated using SQL Server Integration Services, which can be executed using [Azure Data Factory](#).

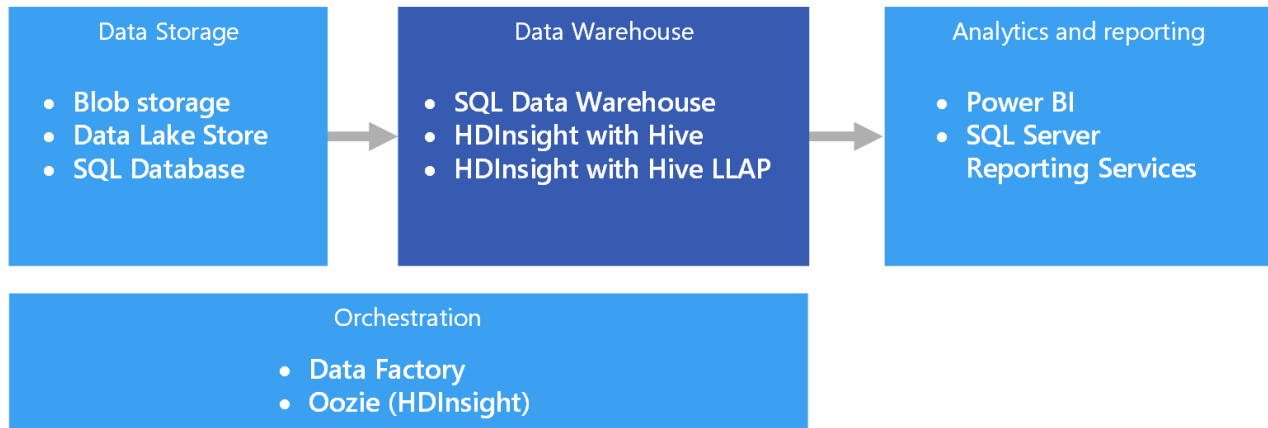
## Technology choices

- [Online Analytical Processing \(OLAP\) data stores](#)

# Data warehousing and data marts

2/13/2018 • 4 min to read • [Edit Online](#)

A data warehouse is a central, organizational, relational repository of integrated data from one or more disparate sources, across many or all subject areas. Data warehouses store current and historical data and are used for reporting and analysis of the data in different ways.



To move data into a data warehouse, it is extracted on a periodic basis from various sources that contain important business information. As the data is moved, it can be formatted, cleaned, validated, summarized, and reorganized. Alternately, the data can be stored in the lowest level of detail, with aggregated views provided in the warehouse for reporting. In either case, the data warehouse becomes a permanent storage space for data used for reporting, analysis, and forming important business decisions using business intelligence (BI) tools.

## Data marts and operational data stores

Managing data at scale is complex, and it is becoming less common to have a single data warehouse that represents all data across the entire enterprise. Instead, organizations create smaller, more focused data warehouses, called *data marts*, that expose the desired data for analytics purposes. An orchestration process populates the data marts from data maintained in an operational data store. The operational data store acts as an intermediary between the source transactional system and the data mart. Data managed by the operational data store is a cleaned version of the data present in the source transactional system, and is typically a subset of the historical data that is maintained by the data warehouse or data mart.

## When to use this solution

Choose a data warehouse when you need to turn massive amounts of data from operational systems into a format that is easy to understand, current, and accurate. Data warehouses do not need to follow the same terse data structure you may be using in your operational/OLTP databases. You can use column names that make sense to business users and analysts, restructure the schema to simplify data relationships, and consolidate several tables into one. These steps help guide users who need to create ad hoc reports, or create reports and analyze the data in BI systems, without the help of a database administrator (DBA) or data developer.

Consider using a data warehouse when you need to keep historical data separate from the source transaction systems for performance reasons. Data warehouses make it easy to access historical data from multiple locations, by providing a centralized location using common formats, common keys, common data models, and common access methods.

Data warehouses are optimized for read access, resulting in faster report generation compared to running reports against the source transaction system. In addition, data warehouses provide the following benefits:

- All historical data from multiple sources can be stored and accessed from a data warehouse as the single source of truth.
- You can improve data quality by cleaning up data as it is imported into the data warehouse, providing more accurate data as well as providing consistent codes and descriptions.
- Reporting tools do not compete with the transactional source systems for query processing cycles. A data warehouse allows the transactional system to focus predominantly on handling writes, while the data warehouse satisfies the majority of read requests.
- A data warehouse can help consolidate data from different software.
- Data mining tools can help you find hidden patterns using automatic methodologies against data stored in your warehouse.
- Data warehouses make it easier to provide secure access to authorized users, while restricting access to others. There is no need to grant business users access to the source data, thereby removing a potential attack vector against one or more production transaction systems.
- Data warehouses make it easier to create business intelligence solutions on top of the data, such as [OLAP cubes](#).

## Challenges

Properly configuring a data warehouse to fit the needs of your business can bring some of the following challenges:

- Committing the time required to properly model your business concepts. This is an important step, as data warehouses are information driven, where concept mapping drives the rest of the project. This involves standardizing business-related terms and common formats (such as currency and dates), and restructuring the schema in a way that makes sense to business users but still ensures accuracy of data aggregates and relationships.
- Planning and setting up your data orchestration. Consideration include how to copy data from the source transactional system to the data warehouse, and when to move historical data out of your operational data stores and into the warehouse.
- Maintaining or improving data quality by cleaning the data as it is imported into the warehouse.

## Data warehousing in Azure

In Azure, you may have one or more sources of data, whether from customer transactions, or from various business applications used by various departments. This data is traditionally stored in one or more [OLTP](#) databases. The data could be persisted in other storage mediums such as network shares, Azure Storage Blobs, or a data lake. The data could also be stored by the data warehouse itself or in a relational database such as Azure SQL Database. The purpose of the analytical data store layer is to satisfy queries issued by analytics and reporting tools against the data warehouse or data mart. In Azure, this analytical store capability can be met with Azure SQL Data Warehouse, or with Azure HDInsight using Hive or Interactive Query. In addition, you will need some level of orchestration to periodically move or copy data from data storage to the data warehouse, which can be done using Azure Data Factory or Oozie on Azure HDInsight.

Related services:

- [Azure SQL Database](#)
- [SQL Server in a VM](#)
- [Azure Data Warehouse](#)
- [Apache Hive on HDInsight](#)
- [Interactive Query \(Hive LLAP\) on HDInsight](#)

## Technology choices

- [Data warehouses](#)
- [Pipeline orchestration](#)



# Extract, transform, and load (ETL)

2/13/2018 • 5 min to read • [Edit Online](#)

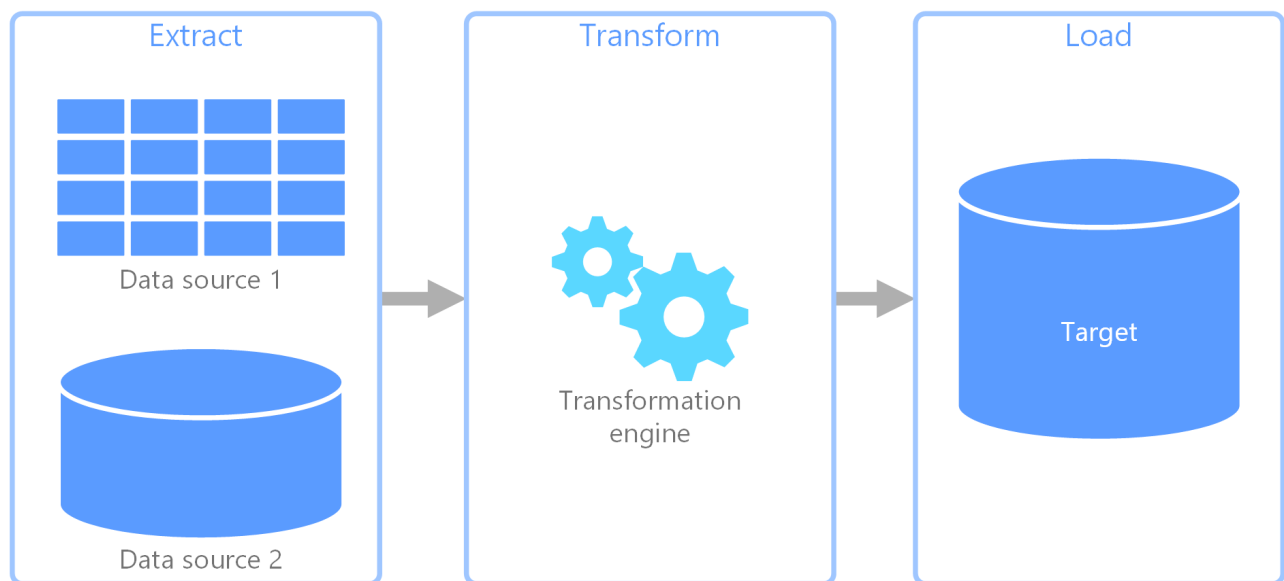
A common problem that organizations face is how to gathering data from multiple sources, in multiple formats, and move it to one or more data stores. The destination may not be the same type of data store as the source, and often the format is different, or the data needs to be shaped or cleaned before loading it into its final destination.

Various tools, services, and processes have been developed over the years to help address these challenges. No matter the process used, there is a common need to coordinate the work and apply some level of data transformation within the data pipeline. The following sections highlight the common methods used to perform these tasks.

## Extract, transform, and load (ETL)

Extract, transform, and load (ETL) is a data pipeline used to collect data from various sources, transform the data according to business rules, and load it into a destination data store. The transformation work in ETL takes place in a specialized engine, and often involves using staging tables to temporarily hold data as it is being transformed and ultimately loaded to its destination.

The data transformation that takes place usually involves various operations, such as filtering, sorting, aggregating, joining data, cleaning data, deduplicating, and validating data.



Often, the three ETL phases are run in parallel to save time. For example, while data is being extracted, a transformation process could be working on data already received and prepare it for loading, and a loading process can begin working on the prepared data, rather than waiting for the entire extraction process to complete.

Relevant Azure service:

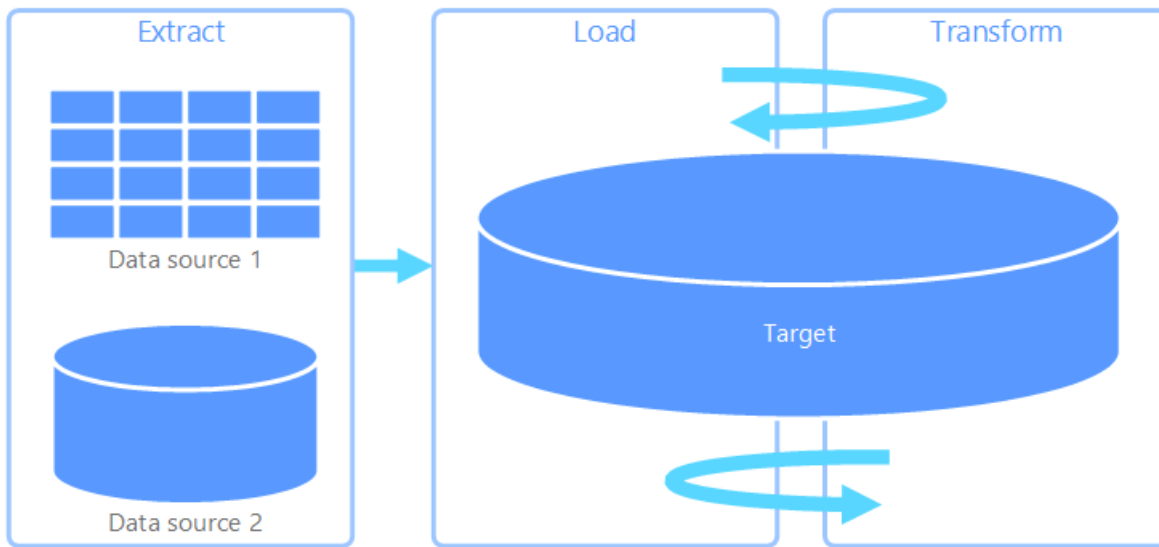
- [Azure Data Factory v2](#)

Other tools:

- [SQL Server Integration Services \(SSIS\)](#)

## Extract, load, and transform (ELT)

Extract, load, and transform (ELT) differs from ETL solely in where the transformation takes place. In the ELT pipeline, the transformation occurs in the target data store. Instead of using a separate transformation engine, the processing capabilities of the target data store are used to transform data. This simplifies the architecture by removing the transformation engine from the pipeline. Another benefit to this approach is that scaling the target data store also scales the ELT pipeline performance. However, ELT only works well when the target system is powerful enough to transform the data efficiently.



Typical use cases for ELT fall within the big data realm. For example, you might start by extracting all of the source data to flat files in scalable storage such as Hadoop distributed file system (HDFS) or Azure Data Lake Store. Technologies such as Spark, Hive, or PolyBase can then be used to query the source data. The key point with ELT is that the data store used to perform the transformation is the same data store where the data is ultimately consumed. This data store reads directly from the scalable storage, instead of loading the data into its own proprietary storage. This approach skips the data copy step present in ETL, which can be a time consuming operation for large data sets.

In practice, the target data store is a [data warehouse](#) using either a Hadoop cluster (using Hive or Spark) or a SQL Data Warehouse. In general, a schema is overlaid on the flat file data at query time and stored as a table, enabling the data to be queried like any other table in the data store. These are referred to as external tables because the data does not reside in storage managed by the data store itself, but on some external scalable storage.

The data store only manages the schema of the data and applies the schema on read. For example, a Hadoop cluster using Hive would describe a Hive table where the data source is effectively a path to a set of files in HDFS. In SQL Data Warehouse, PolyBase can achieve the same result — creating a table against data stored externally to the database itself. Once the source data is loaded, the data present in the external tables can be processed using the capabilities of the data store. In big data scenarios, this means the data store must be capable of massively parallel processing (MPP), which breaks the data into smaller chunks and distributes processing of the chunks across multiple machines in parallel.

The final phase of the ELT pipeline is typically to transform the source data into a final format that is more efficient for the types of queries that need to be supported. For example, the data may be partitioned. Also, ELT might use optimized storage formats like Parquet, which stores row-oriented data in a columnar fashion and provides optimized indexing.

Relevant Azure service:

- [Azure SQL Data Warehouse](#)
- [HDInsight with Hive](#)
- [Azure Data Factory v2](#)
- [Oozie on HDInsight](#)

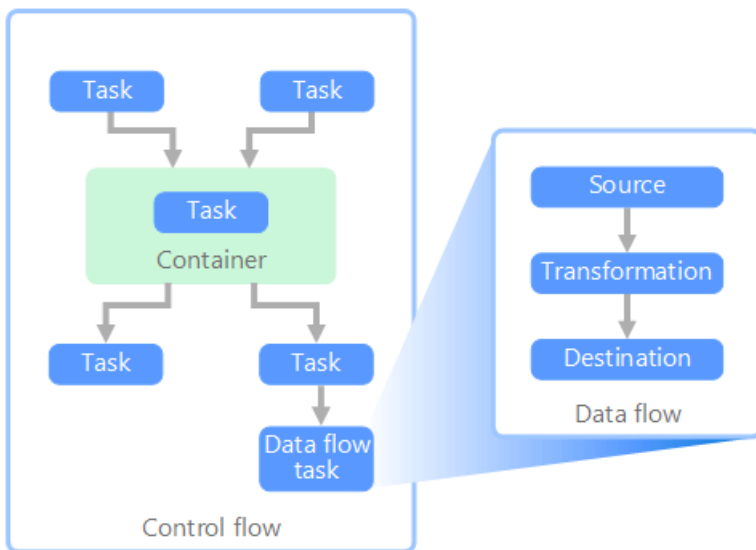
Other tools:

- [SQL Server Integration Services \(SSIS\)](#)

## Data flow and control flow

In the context of data pipelines, the control flow ensures orderly processing of a set of tasks. To enforce the correct processing order of these tasks, precedence constraints are used. You can think of these constraints as connectors in a workflow diagram, as shown in the image below. Each task has an outcome, such as success, failure, or completion. Any subsequent task does not initiate processing until its predecessor has completed with one of these outcomes.

Control flows execute data flows as a task. In a data flow task, data is extracted from a source, transformed, or loaded into a data store. The output of one data flow task can be the input to the next data flow task, and data flows can run in parallel. Unlike control flows, you cannot add constraints between tasks in a data flow. You can, however, add a data viewer to observe the data as it is processed by each task.



In the diagram above, there are several tasks within the control flow, one of which is a data flow task. One of the tasks is nested within a container. Containers can be used to provide structure to tasks, providing a unit of work. One such example is for repeating elements within a collection, such as files in a folder or database statements.

Relevant Azure service:

- [Azure Data Factory v2](#)

Other tools:

- [SQL Server Integration Services \(SSIS\)](#)

## Technology choices

- [Online Transaction Processing \(OLTP\) data stores](#)
- [Online Analytical Processing \(OLAP\) data stores](#)
- [Data warehouses](#)
- [Pipeline orchestration](#)

# Choosing a data warehouse in Azure

2/13/2018 • 7 min to read • [Edit Online](#)

A data warehouse is a central, organizational, relational repository of integrated data from one or more disparate sources. This topic compares options for data warehouses in Azure.

## NOTE

For more information about when to use a data warehouse, see [Data warehousing and data marts](#).

## What are your options when choosing a data warehouse?

There are several options for implementing a data warehouse in Azure, depending on your needs. The following lists are broken into two categories, [symmetric multiprocessing](#) (SMP) and [massively parallel processing](#) (MPP).

SMP:

- [Azure SQL Database](#)
- [SQL Server in a virtual machine](#)

MPP:

- [Azure Data Warehouse](#)
- [Apache Hive on HDInsight](#)
- [Interactive Query \(Hive LLAP\) on HDInsight](#)

As a general rule, SMP-based warehouses are best suited for small to medium data sets (up to 4-100 TB), while MPP is often used for big data. The delineation between small/medium and big data partly has to do with your organization's definition and supporting infrastructure. (See [Choosing an OLTP data store](#).)

Beyond data sizes, the type of workload pattern is likely to be a greater determining factor. For example, complex queries may be too slow for an SMP solution, and require an MPP solution instead. MPP-based systems are likely to impose a performance penalty with small data sizes, due to the way jobs are distributed and consolidated across nodes. If your data sizes already exceed 1 TB and are expected to continually grow, consider selecting an MPP solution. However, if your data sizes are less than this, but your workloads are exceeding the available resources of your SMP solution, then MPP may be your best option as well.

The data accessed or stored by your data warehouse could come from a number of data sources, including a data lake, such as [Azure Data Lake Store](#). For a video session that compares the different strengths of MPP services that can use Azure Data Lake, see [Azure Data Lake and Azure Data Warehouse: Applying Modern Practices to Your App](#).

SMP systems are characterized by a single instance of a relational database management system sharing all resources (CPU/Memory/Disk). You can scale up an SMP system. For SQL Server running on a VM, you can scale up the VM size. For Azure SQL Database, you can scale up by selecting a different service tier.

MPP systems can be scaled out by adding more compute nodes (which have their own CPU, memory and I/O subsystems). There are physical limitations to scaling up a server, at which point scaling out is more desirable, depending on the workload. However, MPP solutions require a different skillset, due to variances in querying, modeling, partitioning of data, and other factors unique to parallel processing.

When deciding which SMP solution to use, see [A closer look at Azure SQL Database and SQL Server on Azure VMs](#).

Azure SQL Data Warehouse can also be used for small and medium datasets, where the workload is compute and memory intensive. Read more about SQL Data Warehouse patterns and common scenarios:

- [SQL Data Warehouse Patterns and Anti-Patterns](#)
- [SQL Data Warehouse Loading Patterns and Strategies](#)
- [Migrating Data to Azure SQL Data Warehouse](#)
- [Common ISV Application Patterns Using Azure SQL Data Warehouse](#)

## Key selection criteria

To narrow the choices, start by answering these questions:

- Do you want a managed service rather than managing your own servers?
- Are you working with extremely large data sets or highly complex, long-running queries? If yes, consider an MPP option.
- For a large data set, is the data source structured or unstructured? Unstructured data may need to be processed in a big data environment such as Spark on HDInsight, Azure Databricks, Hive LLAP on HDInsight, or Azure Data Lake Analytics. All of these can serve as ELT (Extract, Load, Transform) and ETL (Extract, Transform, Load) engines. They can output the processed data into structured data, making it easier to load into SQL Data Warehouse or one of the other options. For structured data, SQL Data Warehouse has a performance tier called Optimized for Compute, for compute-intensive workloads requiring ultra-high performance.
- Do you want to separate your historical data from your current, operational data? If so, select one of the options where [orchestration](#) is required. These are standalone warehouses optimized for heavy read access, and are best suited as a separate historical data store.
- Do you need to integrate data from several sources, beyond your OLTP data store? If so, consider options that easily integrate multiple data sources.
- Do you have a multi-tenancy requirement? If so, SQL Data Warehouse is not ideal for this requirement. For more information, see [SQL Data Warehouse Patterns and Anti-Patterns](#).
- Do you prefer a relational data store? If so, narrow your options to those with a relational data store, but also note that you can use a tool like PolyBase to query non-relational data stores if needed. If you decide to use PolyBase, however, run performance tests against your unstructured data sets for your workload.
- Do you have real-time reporting requirements? If you require rapid query response times on high volumes of singleton inserts, narrow your options to those that can support real-time reporting.
- Do you need to support a large number of concurrent users and connections? The ability to support a number of concurrent users/connections depends on several factors.
  - For Azure SQL Database, refer to the [documented resource limits](#) based on your service tier.
  - SQL Server allows a maximum of 32,767 user connections. When running on a VM, performance will depend on the VM size and other factors.
  - SQL Data Warehouse has limits on concurrent queries and concurrent connections. For more information, see [Concurrency and workload management in SQL Data Warehouse](#). Consider using complementary services, such as [Azure Analysis Services](#), to overcome limits in SQL Data Warehouse.
- What sort of workload do you have? In general, MPP-based warehouse solutions are best suited for analytical, batch-oriented workloads. If your workloads are transactional by nature, with many small read/write operations or multiple row-by-row operations, consider using one of the SMP options. One

exception to this guideline is when using stream processing on an HDInsight cluster, such as Spark Streaming, and storing the data within a Hive table.

## Capability Matrix

The following tables summarize the key differences in capabilities.

### General capabilities

	AZURE SQL DATABASE	SQL SERVER (VM)	SQL DATA WAREHOUSE	APACHE HIVE ON HDINSIGHT	HIVE LLAP ON HDINSIGHT
Is managed service	Yes	No	Yes	Yes <sup>1</sup>	Yes <sup>1</sup>
Requires data orchestration (holds copy of data/historical data)	No	No	Yes	Yes	Yes
Easily integrate multiple data sources	No	No	Yes	Yes	Yes
Supports pausing compute	No	No	Yes	No <sup>2</sup>	No <sup>2</sup>
Relational data store	Yes	Yes	Yes	No	No
Real-time reporting	Yes	Yes	No	No	Yes
Flexible backup restore points	Yes	Yes	No <sup>3</sup>	Yes <sup>4</sup>	Yes <sup>4</sup>
SMP/MPP	SMP	SMP	MPP	MPP	MPP

[1] Manual configuration and scaling.

[2] HDInsight clusters can be deleted when not needed, and then re-created. Attach an external data store to your cluster so your data is retained when you delete your cluster. You can use Azure Data Factory to automate your cluster's lifecycle by creating an on-demand HDInsight cluster to process your workload, then delete it once the processing is complete.

[3] With SQL Data Warehouse, you can restore a database to any available restore point within the last seven days. Snapshots start every four to eight hours and are available for seven days. When a snapshot is older than seven days, it expires and its restore point is no longer available.

[4] Consider using an [external Hive metastore](#) that can be backed up and restored as needed. Standard backup and restore options that apply to Blob Storage or Data Lake Store can be used for the data, or third party HDInsight backup and restore solutions, such as [Imanis Data](#) can be used for greater flexibility and ease of use.

### Scalability capabilities

	<b>AZURE SQL DATABASE</b>	<b>SQL SERVER (VM)</b>	<b>SQL DATA WAREHOUSE</b>	<b>APACHE HIVE ON HDINSIGHT</b>	<b>HIVE LLAP ON HDINSIGHT</b>
Redundant regional servers for high availability	Yes	Yes	Yes	No	No
Supports query scale out (distributed queries)	No	No	Yes	Yes	Yes
Dynamic scalability (scale up)	Yes	No	Yes <sup>1</sup>	No	No
Supports in-memory caching of data	Yes	Yes	No	Yes	Yes

[1] SQL Data Warehouse allows you to scale up or down by adjusting the number of data warehouse units (DWUs). See [Manage compute power in Azure SQL Data Warehouse](#).

### Security capabilities

	<b>AZURE SQL DATABASE</b>	<b>SQL SERVER IN A VIRTUAL MACHINE</b>	<b>SQL DATA WAREHOUSE</b>	<b>APACHE HIVE ON HDINSIGHT</b>	<b>HIVE LLAP ON HDINSIGHT</b>
Authentication	SQL / Azure Active Directory (Azure AD)	SQL / Azure AD / Active Directory	SQL / Azure AD	local / Azure AD <sup>1</sup>	local / Azure AD <sup>1</sup>
Authorization	Yes	Yes	Yes	Yes	Yes <sup>1</sup>
Auditing	Yes	Yes	Yes	Yes	Yes <sup>1</sup>
Data encryption at rest	Yes <sup>2</sup>	Yes <sup>2</sup>	Yes <sup>2</sup>	Yes <sup>2</sup>	Yes <sup>1</sup>
Row-level security	Yes	Yes	Yes	No	Yes <sup>1</sup>
Supports firewalls	Yes	Yes	Yes	Yes	Yes <sup>3</sup>
Dynamic data masking	Yes	Yes	Yes	No	Yes <sup>1</sup>

[1] Requires using a [domain-joined HDInsight cluster](#).

[2] Requires using Transparent Data Encryption (TDE) to encrypt and decrypt your data at rest.

[3] Supported when [used within an Azure Virtual Network](#).

Read more about securing your data warehouse:

- [Securing your SQL Database](#)
- [Secure a database in SQL Data Warehouse](#)

- [Extend Azure HDInsight using an Azure Virtual Network](#)
- [Enterprise-level Hadoop security with domain-joined HDInsight clusters](#)



# Choosing an OLAP data store in Azure

2/13/2018 • 3 min to read • [Edit Online](#)

Online analytical processing (OLAP) is a technology that organizes large business databases and supports complex analysis. This topic compares the options for OLAP solutions in Azure.

## NOTE

For more information about when to use an OLAP data store, see [Online analytical processing](#).

## What are your options when choosing an OLAP data store?

In Azure, all of the following data stores will meet the core requirements for OLAP:

- [SQL Server with Columnstore indexes](#)
- [Azure Analysis Services](#)
- [SQL Server Analysis Services \(SSAS\)](#)

SQL Server Analysis Services (SSAS) offers OLAP and data mining functionality for business intelligence applications. You can either install SSAS on local servers, or host within a virtual machine in Azure. Azure Analysis Services is a fully managed service that provides the same major features as SSAS. Azure Analysis Services supports connecting to [various data sources](#) in the cloud and on-premises in your organization.

Clustered Columnstore indexes are available in SQL Server 2014 and above, as well as Azure SQL Database, and are ideal for OLAP workloads. However, beginning with SQL Server 2016 (including Azure SQL Database), you can take advantage of hybrid transactional/analytics processing (HTAP) through the use of updateable nonclustered columnstore indexes. HTAP enables you to perform OLTP and OLAP processing on the same platform, which removes the need to store multiple copies of your data, and eliminates the need for distinct OLTP and OLAP systems. For more information, see [Get started with Columnstore for real-time operational analytics](#).

## Key selection criteria

To narrow the choices, start by answering these questions:

- Do you want a managed service rather than managing your own servers?
- Do you require secure authentication using Azure Active Directory (Azure AD)?
- Do you want to conduct real-time analytics? If so, narrow your options to those that support real-time analytics.

*Real-time analytics* in this context applies to a single data source, such as an enterprise resource planning (ERP) application, that will run both an operational and an analytics workload. If you need to integrate data from multiple sources, or require extreme analytics performance by using pre-aggregated data such as cubes, you might still require a separate data warehouse.

- Do you need to use pre-aggregated data, for example to provide semantic models that make analytics more business user friendly? If yes, choose an option that supports multidimensional cubes or tabular semantic models.

Providing aggregates can help users consistently calculate data aggregates. Pre-aggregated data can also provide a large performance boost when dealing with several columns across many rows. Data can be pre-

aggregated in multidimensional cubes or tabular semantic models.

- Do you need to integrate data from several sources, beyond your OLTP data store? If so, consider options that easily integrate multiple data sources.

## Capability matrix

The following tables summarize the key differences in capabilities.

### General capabilities

	AZURE ANALYSIS SERVICES	SQL SERVER ANALYSIS SERVICES	SQL SERVER WITH COLUMNSTORE INDEXES	AZURE SQL DATABASE WITH COLUMNSTORE INDEXES
Is managed service	Yes	No	No	Yes
Supports multidimensional cubes	No	Yes	No	No
Supports tabular semantic models	Yes	Yes	No	No
Easily integrate multiple data sources	Yes	Yes	No <sup>1</sup>	No <sup>1</sup>
Supports real-time analytics	No	No	Yes	Yes
Requires process to copy data from source(s)	Yes	Yes	No	No
Azure AD integration	Yes	No	No <sup>2</sup>	Yes

[1] Although SQL Server and Azure SQL Database cannot be used to query from and integrate multiple external data sources, you can still build a pipeline that does this for you using [SSIS](#) or [Azure Data Factory](#). SQL Server hosted in an Azure VM has additional options, such as linked servers and [PolyBase](#). For more information, see [Pipeline orchestration, control flow, and data movement](#).

[2] Connecting to SQL Server running on an Azure Virtual Machine is not supported using an Azure AD account. Use a domain Active Directory account instead.

### Scalability Capabilities

	AZURE ANALYSIS SERVICES	SQL SERVER ANALYSIS SERVICES	SQL SERVER WITH COLUMNSTORE INDEXES	AZURE SQL DATABASE WITH COLUMNSTORE INDEXES
Redundant regional servers for high availability	Yes	No	Yes	Yes
Supports query scale out	Yes	No	Yes	No

	<b>AZURE ANALYSIS SERVICES</b>	<b>SQL SERVER ANALYSIS SERVICES</b>	<b>SQL SERVER WITH COLUMNSTORE INDEXES</b>	<b>AZURE SQL DATABASE WITH COLUMNSTORE INDEXES</b>
Dynamic scalability (scale up)	Yes	No	Yes	No

# Choosing an OLTP data store in Azure

2/13/2018 • 3 min to read • [Edit Online](#)

Online transaction processing (OLTP) is the management of transactional data and transaction processing. This topic compares options for OLTP solutions in Azure.

## NOTE

For more information about when to use an OLTP data store, see [Online transaction processing](#).

## What are your options when choosing an OLTP data store?

In Azure, all of the following data stores will meet the core requirements for OLTP and the management of transaction data:

- [Azure SQL Database](#)
- [SQL Server in an Azure virtual machine](#)
- [Azure Database for MySQL](#)
- [Azure Database for PostgreSQL](#)

## Key selection criteria

To narrow the choices, start by answering these questions:

- Do you want a managed service rather than managing your own servers?
- Does your solution have specific dependencies for Microsoft SQL Server, MySQL or PostgreSQL compatibility? Your application may limit the data stores you can choose based on the drivers it supports for communicating with the data store, or the assumptions it makes about which database is used.
- Are your write throughput requirements particularly high? If yes, choose an option that provides in-memory tables.
- Is your solution multi-tenant? If so, consider options that support capacity pools, where multiple database instances draw from an elastic pool of resources, instead of fixed resources per database. This can help you better distribute capacity across all database instances, and can make your solution more cost effective.
- Does your data need to be readable with low latency in multiple regions? If yes, choose an option that supports readable secondary replicas.
- Does your database need to be highly available across geo-graphic regions? If yes, choose an option that supports geographic replication. Also consider the options that support automatic failover from the primary replica to a secondary replica.
- Does your database have specific security needs? If yes, examine the options that provide capabilities like row level security, data masking, and transparent data encryption.

## Capability matrix

The following tables summarize the key differences in capabilities.

### General capabilities

	AZURE SQL DATABASE	SQL SERVER IN AN AZURE VIRTUAL MACHINE	AZURE DATABASE FOR MYSQL	AZURE DATABASE FOR POSTGRESQL
Is Managed Service	Yes	No	Yes	Yes
Runs on Platform	N/A	Windows, Linux, Docker	N/A	N/A
Programmability <sup>1</sup>	T-SQL, .NET, R	T-SQL, .NET, R, Python	T-SQL, .NET, R, Python	SQL

[1] Not including client driver support, which allows many programming languages to connect to and use the OLTP data store.

### Scalability capabilities

	AZURE SQL DATABASE	SQL SERVER IN AN AZURE VIRTUAL MACHINE	AZURE DATABASE FOR MYSQL	AZURE DATABASE FOR POSTGRESQL
Maximum database instance size	4 TB	256 TB	1 TB	1 TB
Supports capacity pools	Yes	Yes	No	No
Supports clusters scale out	No	Yes	No	No
Dynamic scalability (scale up)	Yes	No	Yes	Yes

### Analytic workload capabilities

	AZURE SQL DATABASE	SQL SERVER IN AN AZURE VIRTUAL MACHINE	AZURE DATABASE FOR MYSQL	AZURE DATABASE FOR POSTGRESQL
Temporal tables	Yes	Yes	No	No
In-memory (memory-optimized) tables	Yes	Yes	No	No
Columnstore support	Yes	Yes	No	No
Adaptive query processing	Yes	Yes	No	No

### Availability capabilities

	AZURE SQL DATABASE	SQL SERVER IN AN AZURE VIRTUAL MACHINE	AZURE DATABASE FOR MYSQL	AZURE DATABASE FOR POSTGRESQL
Readable secondaries	Yes	Yes	No	No

	AZURE SQL DATABASE	SQL SERVER IN AN AZURE VIRTUAL MACHINE	AZURE DATABASE FOR MYSQL	AZURE DATABASE FOR POSTGRESQL
Geographic replication	Yes	Yes	No	No
Automatic failover to secondary	Yes	No	No	No
Point-in-time restore	Yes	Yes	Yes	Yes

## Security capabilities

	AZURE SQL DATABASE	SQL SERVER IN AN AZURE VIRTUAL MACHINE	AZURE DATABASE FOR MYSQL	AZURE DATABASE FOR POSTGRESQL
Row level security	Yes	Yes	Yes	Yes
Data masking	Yes	Yes	No	No
Transparent data encryption	Yes	Yes	Yes	Yes
Restrict access to specific IP addresses	Yes	Yes	Yes	Yes
Restrict access to allow VNET access only	Yes	Yes	No	No
Azure Active Directory authentication	Yes	Yes	No	No
Active Directory authentication	No	Yes	No	No
Multi-factor authentication	Yes	Yes	No	No
Supports <a href="#">Always Encrypted</a>	Yes	Yes	Yes	No
Private IP	No	Yes	Yes	No

# Non-relational data and NoSQL

2/13/2018 • 12 min to read • [Edit Online](#)

A *non-relational database* is a database that does not use the tabular schema of rows and columns found in most traditional database systems. Instead, non-relational databases use a storage model that is optimized for the specific requirements of the type of data being stored. For example, data may be stored as simple key/value pairs, as JSON documents, or as a graph consisting of edges and vertices.

What all of these data stores have in common is that they don't use a [relational model](#). Also, they tend to be more specific in the type of data they support and how data can be queried. For example, time series data stores are optimized for queries over time-based sequences of data, while graph data stores are optimized for exploring weighted relationships between entities. Neither format would generalize well to the task of managing transactional data.

The term *NoSQL* refers to data stores that do not use SQL for queries, and instead use other programming languages and constructs to query the data. In practice, "NoSQL" means "non-relational database," even though many of these databases do support SQL-compatible queries. However, the underlying query execution strategy is usually very different from the way a traditional RDBMS would execute the same SQL query.

The following sections describe the major categories of non-relational or NoSQL database.

## Document data stores

A document data store manages a set of named string fields and object data values in an entity referred to as a *document*. These data stores typically store data in the form of JSON documents. Each field value could be a scalar item, such as a number, or a compound element, such as a list or a parent-child collection. The data in the fields of a document can be encoded in a variety of ways, including XML, YAML, JSON, BSON, or even stored as plain text. The fields within documents are exposed to the storage management system, enabling an application to query and filter data by using the values in these fields.

Typically, a document contains the entire data for an entity. What items constitute an entity are application specific. For example, an entity could contain the details of a customer, an order, or a combination of both. A single document might contain information that would be spread across several relational tables in a relational database management system (RDBMS). A document store does not require that all documents have the same structure. This free-form approach provides a great deal of flexibility. For example, applications can store different data in documents in response to a change in business requirements.

Key	Document
1001	<pre>{   "CustomerID": 99,   "OrderItems": [     { "ProductID": 2010,       "Quantity": 2,       "Cost": 520     },     { "ProductID": 4365,       "Quantity": 1,       "Cost": 18     }   ],   "OrderDate": "04/01/2017" }</pre>
1002	<pre>{   "CustomerID": 220,   "OrderItems": [     { "ProductID": 1285,       "Quantity": 1,       "Cost": 120     }   ],   "OrderDate": "05/08/2017" }</pre>

The application can retrieve documents by using the document key. This is a unique identifier for the document, which is often hashed, to help distribute data evenly. Some document databases create the document key automatically. Others enable you to specify an attribute of the document to use as the key. The application can also query documents based on the value of one or more fields. Some document databases support indexing to facilitate fast lookup of documents based on one or more indexed fields.

Many document databases support in-place updates, enabling an application to modify the values of specific fields in a document without rewriting the entire document. Read and write operations over multiple fields in a single document are usually atomic.

Relevant Azure service:

- [Azure Cosmos DB](#)

## Columnar data stores

A columnar or column-family data store organizes data into columns and rows. In its simplest form, a column-family data store can appear very similar to a relational database, at least conceptually. The real power of a column-family database lies in its denormalized approach to structuring sparse data, which stems from the column-oriented approach to storing data.

You can think of a column-family data store as holding tabular data with rows and columns, but the columns are divided into groups known as column families. Each column family holds a set of columns that are logically related and are typically retrieved or manipulated as a unit. Other data that is accessed separately can be stored in separate column families. Within a column family, new columns can be added dynamically, and rows can be sparse (that is, a row doesn't need to have a value for every column).

The following diagram shows an example with two column families, `Identity` and `Contact Info`. The data for a single entity has the same row key in each column family. This structure, where the rows for any given object in a column family can vary dynamically, is an important benefit of the column-family approach, making this form of data store highly suited for storing data with varying schemas.

CustomerID	Column Family: Identity	CustomerID	Column Family: Contact Info
001	First name: Mu Bae Last name: Min	001	Phone number: 555-0100 Email: someone@example.com
002	First name: Francisco Last name: Vila Nova Suffix: Jr.	002	Email: vilanova@contoso.com
003	First name: Lena Last name: Adamczyk Title: Dr.	003	Phone number: 555-0120

Unlike a key/value store or a document database, most column-family databases physically store data in key order, rather than by computing a hash. The row key is considered the primary index and enables key-based access via a specific key or a range of keys. Some implementations allow you to create secondary indexes over specific columns in a column family. Secondary indexes let you retrieve data by columns value, rather than row key.

On disk, all of the columns within a column family are stored together in the same file, with a certain number of rows in each file. With large data sets, this approach creates a performance benefit by reducing the amount of data that needs to be read from disk when only a few columns are queried together at a time.

Read and write operations for a row are usually atomic within a single column family, although some implementations provide atomicity across the entire row, spanning multiple column families.

Relevant Azure service:



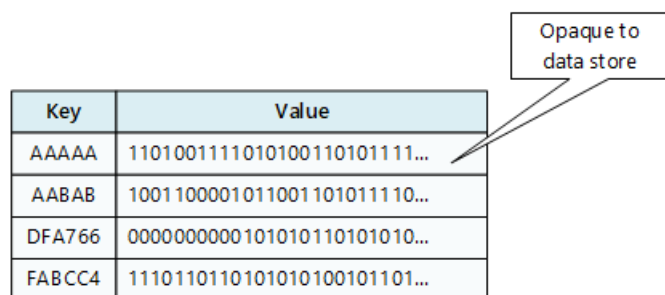
- [HBase in HDInsight](#)

## Key/value data stores

A key/value store is essentially a large hash table. You associate each data value with a unique key, and the key/value store uses this key to store the data by using an appropriate hashing function. The hashing function is selected to provide an even distribution of hashed keys across the data storage.

Most key/value stores only support simple query, insert, and delete operations. To modify a value (either partially or completely), an application must overwrite the existing data for the entire value. In most implementations, reading or writing a single value is an atomic operation. If the value is large, writing may take some time.

An application can store arbitrary data as a set of values, although some key/value stores impose limits on the maximum size of values. The stored values are opaque to the storage system software. Any schema information must be provided and interpreted by the application. Essentially, values are blobs and the key/value store simply retrieves or stores the value by key.



The diagram shows a table with two columns: 'Key' and 'Value'. The 'Value' column contains binary strings. A callout box labeled 'Opaque to data store' points to the first value in the table.

Key	Value
AAAAA	110100111101010011010111...
AABAB	1001100001011001101011110...
DFA766	0000000000101010110101010...
FABCC4	1110110110101010100101101...

Key/value stores are highly optimized for applications performing simple lookups using the value of the key, or by a range of keys, but are less suitable for systems that need to query data across different tables of keys/values, such as joining data across multiple tables.

Key/value stores are also not optimized for scenarios where querying or filtering by non-key values is important, rather than performing lookups based only on keys. For example, with a relational database, you can find a record by using a WHERE clause to filter the non-key columns, but key/values stores usually do not have this type of lookup capability for values, or if they do it requires a slow scan of all values.

A single key/value store can be extremely scalable, as the data store can easily distribute data across multiple nodes on separate machines.

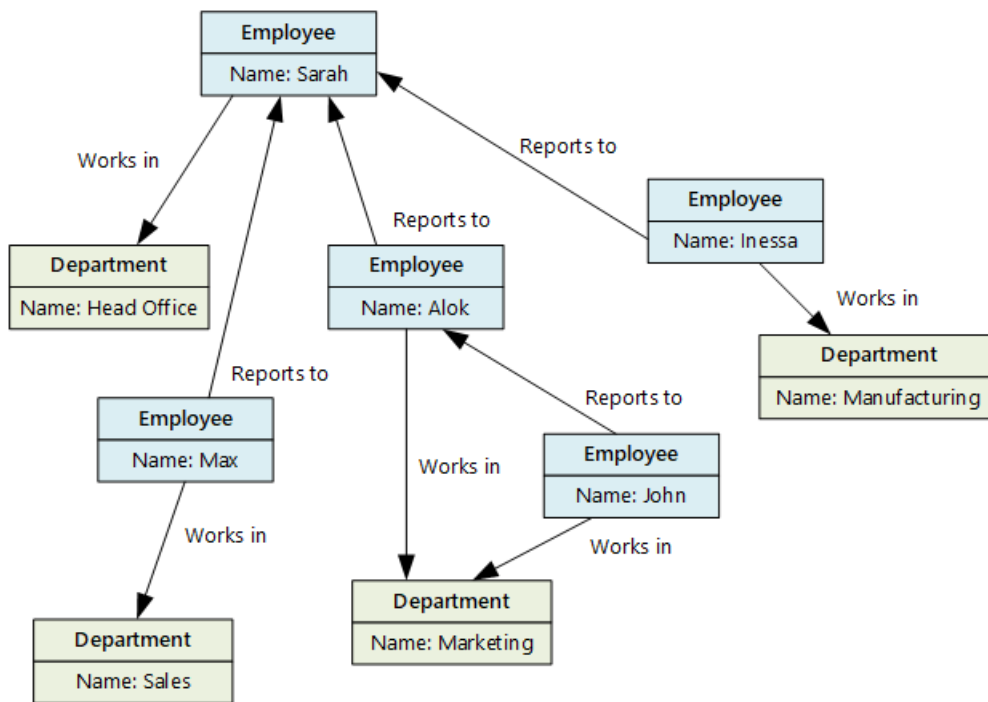
Relevant Azure services:

- [Azure Cosmos DB Table API](#)
- [Azure Redis Cache](#)
- [Azure Table Storage](#)

## Graph data stores

A graph data store manages two types of information, nodes and edges. Nodes represent entities, and edges specify the relationships between these entities. Both nodes and edges can have properties that provide information about that node or edge, similar to columns in a table. Edges can also have a direction indicating the nature of the relationship.

The purpose of a graph data store is to allow an application to efficiently perform queries that traverse the network of nodes and edges, and to analyze the relationships between entities. The following diagram shows an organization's personnel data structured as a graph. The entities are employees and departments, and the edges indicate reporting relationships and the department in which employees work. In this graph, the arrows on the edges show the direction of the relationships.



This structure makes it straightforward to perform queries such as "Find all employees who report directly or indirectly to Sarah" or "Who works in the same department as John?" For large graphs with lots of entities and relationships, you can perform very complex analyses very quickly. Many graph databases provide a query language that you can use to traverse a network of relationships efficiently.

Relevant Azure service:

- [Azure Cosmos DB Graph API](#)

## Time series data stores

Time series data is a set of values organized by time, and a time series data store is optimized for this type of data. Time series data stores must support a very high number of writes, as they typically collect large amounts of data in real time from a large number of sources. Time series data stores are optimized for storing telemetry data. Scenarios include IoT sensors or application/system counters. Updates are rare, and deletes are often done as bulk operations.

timestamp	deviceid	value
2017-01-05T08:00:00.123	1	90.0
2017-01-05T08:00:01.225	2	75.0
2017-01-05T08:01:01.525	2	78.0

Although the records written to a time series database are generally small, there are often a large number of records, and total data size can grow rapidly. Time series data stores also handle out-of-order and late-arriving data, automatic indexing of data points, and optimizations for queries described in terms of windows of time. This last feature enables queries to run across millions of data points and multiple data streams quickly, in order to support time series visualizations, which is a common way that time series data is consumed.

For more information, see [Time series solutions](#)

Relevant Azure service:

- [Azure Time Series Insights](#)

- [OpenTSDB with HBase on HDInsight](#)

## Object data stores

Object data stores are optimized for storing and retrieving large binary objects or blobs such as images, text files, video and audio streams, large application data objects and documents, and virtual machine disk images. An object consists of the stored data, some metadata, and a unique ID for accessing the object. Object stores are designed to support files that are individually very large, as well provide large amounts of total storage to manage all files.

path	blob	meta data
/delays/2017/06/01/flights.csv	0XAABBCCDDEEF...	{created: 2017-06-02}
/delays/2017/06/02/flights.csv	0XAADDCCDDEEF...	{created: 2017-06-03}
/delays/2017/06/03/flights.csv	0XAEBBDEDDEEF...	{created: 2017-06-03}

Some object data stores replicate a given blob across multiple server nodes, which enables fast parallel reads. This in turn enables the scale-out querying of data contained in large files, because multiple processes, typically running on different servers, can each query the large data file simultaneously.

One special case of object data stores is the network file share. Using file shares enables files to be accessed across a network using standard networking protocols like server message block (SMB). Given appropriate security and concurrent access control mechanisms, sharing data in this way can enable distributed services to provide highly scalable data access for basic, low level operations such as simple read and write requests.

Relevant Azure service:

- [Azure Blob Storage](#)
- [Azure Data Lake Store](#)
- [Azure File Storage](#)

## External index data stores

External index data stores provide the ability to search for information held in other data stores and services. An external index acts as a secondary index for any data store, and can be used to index massive volumes of data and provide near real-time access to these indexes.

For example, you might have text files stored in a file system. Finding a file by its file path is quick, but searching based on the contents of the file would require a scan of all of the files, which is slow. An external index lets you create secondary search indexes and then quickly find the path to the files that match your criteria. Another example application of an external index is with key/value stores that only index by the key. You can build a secondary index based on the values in the data, and quickly look up the key that uniquely identifies each matched item.

id	search-document
233358	{"name": "Pacific Crest National Scenic Trail", "county": "San Diego", "elevation":1294, "location": {"type": "Point", "coordinates": [-120.802102,49.00021]}}
801970	{"name": "Lewis and Clark National Historic Trail", "county": "Richland", "elevation":584, "location": {"type": "Point", "coordinates": [-104.8546903,48.1264084]}}
1144102	{"name": "Intake Trail", "county": "Umatilla", "elevation":1076, "location": {"type": "Point", "coordinates": [-118.0468873,45.9981939]}}

The indexes are created by running an indexing process. This can be performed using a pull model, triggered by the data store, or using a push model, initiated by application code. Indexes can be multidimensional and may support free-text searches across large volumes of text data.

External index data stores are often used to support full text and web based search. In these cases, searching can be exact or fuzzy. A fuzzy search finds documents that match a set of terms and calculates how closely they match. Some external indexes also support linguistic analysis that can return matches based on synonyms, genre expansions (for example, matching "dogs" to "pets"), and stemming (for example, searching for "run" also matches "ran" and "running").

Relevant Azure service:

- [Azure Search](#)

## Typical requirements

Non-relational data stores often use a different storage architecture from that used by relational databases. Specifically, they tend towards having no fixed schema. Also, they tend not to support transactions, or else restrict the scope of transactions, and they generally don't include secondary indexes for scalability reasons.

The following compares the requirements for each of the non-relational data stores:

REQUIREMENT	DOCUMENT DATA	COLUMN-FAMILY DATA	KEY/VALUE DATA	GRAPH DATA
Normalization	Denormalized	Denormalized	Denormalized	Normalized
Schema	Schema on read	Column families defined on write, column schema on read	Schema on read	Schema on read
Consistency (across concurrent transactions)	Tunable consistency, document-level guarantees	Column-family-level guarantees	Key-level guarantees	Graph-level guarantees
Atomicity (transaction scope)	Collection	Table	Table	Graph
Locking Strategy	Optimistic (lock free)	Pessimistic (row locks)	Optimistic (ETag)	
Access pattern	Random access	Aggregates on tall/wide data	Random access	Random access

REQUIREMENT	DOCUMENT DATA	COLUMN-FAMILY DATA	KEY/VALUE DATA	GRAPH DATA
Indexing	Primary and secondary indexes	Primary and secondary indexes	Primary index only	Primary and secondary indexes
Data shape	Document	Tabular with column families containing columns	Key and value	Graph containing edges and vertices
Sparse	Yes	Yes	Yes	No
Wide (lots of columns/attributes)	Yes	Yes	No	No
Datum size	Small (KBs) to medium (low MBs)	Medium (MBs) to Large (low GBs)	Small (KBs)	Small (KBs)
Overall Maximum Scale	Very Large (PBs)	Very Large (PBs)	Very Large (PBs)	Large (TBs)

REQUIREMENT	TIME SERIES DATA	OBJECT DATA	EXTERNAL INDEX DATA
Normalization	Normalized	Denormalized	Denormalized
Schema	Schema on read	Schema on read	Schema on write
Consistency (across concurrent transactions)	N/A	N/A	N/A
Atomicity (transaction scope)	N/A	Object	N/A
Locking Strategy	N/A	Pessimistic (blob locks)	N/A
Access pattern	Random access and aggregation	Sequential access	Random access
Indexing	Primary and secondary indexes	Primary index only	N/A
Data shape	Tabular	Blob and metadata	Document
Sparse	No	N/A	No
Wide (lots of columns/attributes)	No	Yes	Yes
Datum size	Small (KBs)	Large (GBs) to Very Large (TBs)	Small (KBs)
Overall Maximum Scale	Large (low TBs)	Very Large (PBs)	Large (low TBs)

# Working with CSV and JSON files for data solutions

2/13/2018 • 4 min to read • [Edit Online](#)

CSV and JSON are likely the most common formats used for ingesting, exchanging, and storing unstructured or semi-structured data.

## About CSV format

CSV (comma-separated values) files are commonly used to exchange tabular data between systems in plain text. They typically contain a header row that provides column names for the data, but are otherwise considered semi-structured. This is due to the fact that CSVs cannot naturally represent hierarchical or relational data. Data relationships are typically handled with multiple CSV files, where foreign keys are stored in columns of one or more files, but the relationships between those files are not expressed by the format itself. Files in CSV format may use other delimiters besides commas, such as tabs or spaces.

Despite their limitations, CSV files are a popular choice for data exchange, because they are supported by a wide range of business, consumer, and scientific applications. For example, database and spreadsheet programs can import and export CSV files. Similarly, most batch and stream data processing engines, such as Spark and Hadoop, natively support serializing and deserializing CSV-formatted files and offer ways to apply a schema on read. This makes it easier to work with the data, by offering options to query against it and store the information in a more efficient data format for faster processing.

## About JSON format

JSON (JavaScript Object Notation) data is represented as key-value pairs in a semi-structured format. JSON is often compared to XML, as both are capable of storing data in hierarchical format, with child data represented inline with its parent. Both are self-describing and human readable, but JSON documents tend to be much smaller, leading to their popular use in online data exchange, especially with the advent of REST-based web services.

JSON-formatted files have several benefits over CSV:

- JSON maintains hierarchical structures, making it easier to hold related data in a single document and represent complex relationships.
- Most programming languages provide native support for deserializing JSON into objects, or provide lightweight JSON serialization libraries.
- JSON supports lists of objects, helping to avoid messy translations of lists into a relational data model.
- JSON is a commonly used file format for NoSQL databases, such as MongoDB, Couchbase, and Azure Cosmos DB.

Since a lot of data coming across the wire is already in JSON format, most web-based programming languages support working with JSON natively, or through the use of external libraries to serialize and deserialize JSON data. This universal support for JSON has led to its use in logical formats through data structure representation, exchange formats for hot data, and data storage for cold data.

Many batch and stream data processing engines natively support JSON serialization and deserialization. Though the data contained within JSON documents may ultimately be stored in a more performance-optimized formats, such as Parquet or Avro, it serves as the raw data for source of truth, which is critical for reprocessing the data as needed.

## When to use CSV or JSON formats

CSVs are more commonly used for exporting and importing data, or processing it for analytics and machine learning. JSON-formatted files have the same benefits, but are more common in hot data exchange solutions. JSON documents are often sent by web and mobile devices performing online transactions, by IoT (internet of things) devices for one-way or bidirectional communication, or by client applications communicating with SaaS and PaaS services or serverless architectures.

CSV and JSON file formats both make it easy to exchange data between dissimilar systems or devices. Their semi-structured formats allow flexibility in transferring almost any type of data, and universal support for these formats make them simple to work with. Both can be used as the raw source of truth in cases where the processed data is stored in binary formats for more efficient querying.

## Working with CSV and JSON data in Azure

Azure provides several solutions for working with CSV and JSON files, depending on your needs. The primary landing place for these files is either Azure Storage or Azure Data Lake Store. Most Azure services that work with these and other text-based files integrate with either object storage service. In some situations, however, you may opt to directly import the data into Azure SQL or some other data store. SQL Server has native support for storing and working with JSON documents, which makes it easy to [import and process those types of files](#). You can use a utility like SQL Bulk Import to easily [import CSV files](#).

Depending on the scenario, you may perform [batch processing](#) or [real-time processing](#) of the data.

## Challenges

There are some challenges to consider when working with these formats:

- Without any restraints on the data model, CSV and JSON files are prone to data corruption ("garbage in, garbage out"). For instance, there's no notion of a date/time object in either file, so the file format does not prevent you from inserting "ABC123" in a date field, for example.
- Using CSV and JSON files as your cold storage solution does not scale well when working with big data. In most cases, they cannot be split into partitions for parallel processing, and cannot be compressed as well as binary formats. This often leads to processing and storing this data into read-optimized formats such as Parquet and ORC (optimized row columnar), which also provide indexes and inline statistics about the data contained.
- You may need to apply a schema on the semi-structured data to make it easier to query and analyze. Typically, this requires storing the data in another form that complies with your environment's data storage needs, such as within a database.

# Data lakes

2/27/2018 • 2 min to read • [Edit Online](#)

A data lake is a storage repository that holds a large amount of data in its native, raw format. Data lake stores are optimized for scaling to terabytes and petabytes of data. The data typically comes from multiple heterogeneous sources, and may be structured, semi-structured, or unstructured. The idea with a data lake is to store everything in its original, untransformed state. This approach differs from a traditional [data warehouse](#), which transforms and processes the data at the time of ingestion.

Advantages of a data lake:

- Data is never thrown away, because the data is stored in its raw format. This is especially useful in a big data environment, when you may not know in advance what insights are available from the data.
- Users can explore the data and create their own queries.
- May be faster than traditional ETL tools.
- More flexible than a data warehouse, because it can store unstructured and semi-structured data.

A complete data lake solution consists of both storage and processing. Data lake storage is designed for fault-tolerance, infinite scalability, and high-throughput ingestion of data with varying shapes and sizes. Data lake processing involves one or more processing engines built with these goals in mind, and can operate on data stored in a data lake at scale.

## When to use a data lake

Typical uses for a data lake include [data exploration](#), data analytics, and machine learning.

A data lake can also act as the data source for a data warehouse. With this approach, the raw data is ingested into the data lake and then transformed into a structured queryable format. Typically this transformation uses an [ELT](#) (extract-load-transform) pipeline, where the data is ingested and transformed in place. Source data that is already relational may go directly into the data warehouse, using an ETL process, skipping the data lake.

Data lake stores are often used in event streaming or IoT scenarios, because they can persist large amounts of relational and nonrelational data without transformation or schema definition. They are built to handle high volumes of small writes at low latency, and are optimized for massive throughput.

## Challenges

- Lack of a schema or descriptive metadata can make the data hard to consume or query.
- Lack of semantic consistency across the data can make it challenging to perform analysis on the data, unless users are highly skilled at data analytics.
- It can be hard to guarantee the quality of the data going into the data lake.
- Without proper governance, access control and privacy issues can be problems. What information is going into the data lake, who can access that data, and for what uses?
- A data lake may not be the best way to integrate data that is already relational.
- By itself, a data lake does not provide integrated or holistic views across the organization.
- A data lake may become a dumping ground for data that is never actually analyzed or mined for insights.

## Relevant Azure services

- [Data Lake Store](#) is a hyper-scale, Hadoop-compatible repository.



- [Data Lake Analytics](#) is an on-demand analytics job service to simplify big data analytics.

# Big data architectures

2/27/2018 • 10 min to read • [Edit Online](#)

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. The threshold at which organizations enter into the big data realm differs, depending on the capabilities of the users and their tools. For some, it can mean hundreds of gigabytes of data, while for others it means hundreds of terabytes. As tools for working with big data sets advance, so does the meaning of big data. More and more, this term relates to the value you can extract from your data sets through advanced analytics, rather than strictly the size of the data, although in these cases they tend to be quite large.

Over the years, the data landscape has changed. What you can do, or are expected to do, with data has changed. The cost of storage has fallen dramatically, while the means by which data is collected keeps growing. Some data arrives at a rapid pace, constantly demanding to be collected and observed. Other data arrives more slowly, but in very large chunks, often in the form of decades of historical data. You might be facing an advanced analytics problem, or one that requires machine learning. These are challenges that big data architectures seek to solve.

Big data solutions typically involve one or more of the following types of workload:

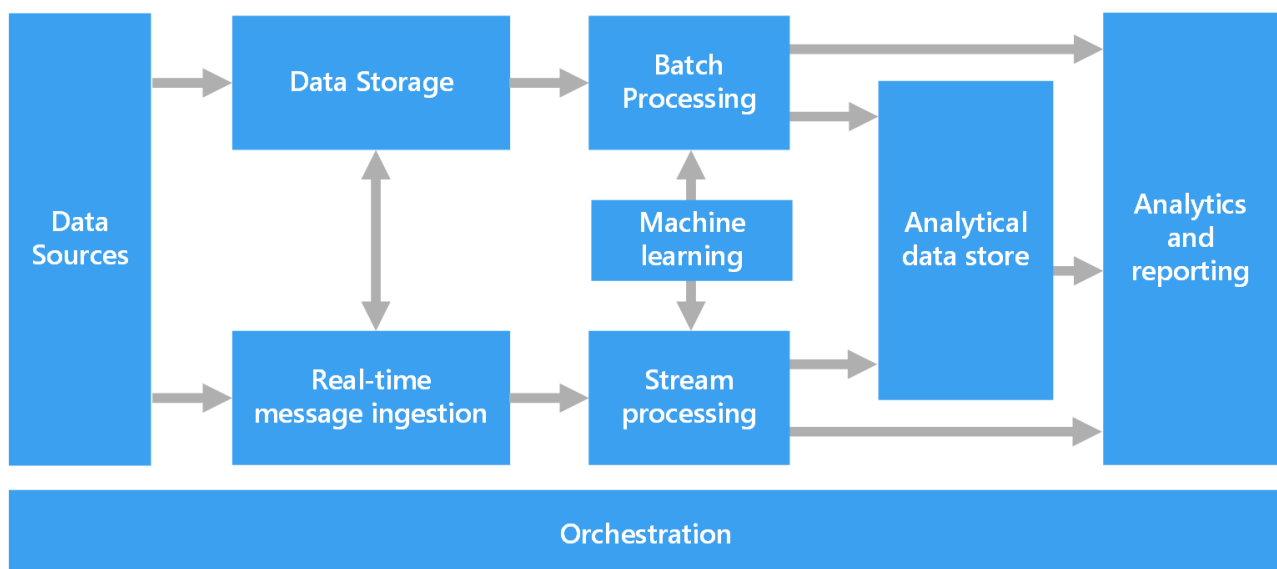
- Batch processing of big data sources at rest.
- Real-time processing of big data in motion.
- Interactive exploration of big data.
- Predictive analytics and machine learning.

Consider big data architectures when you need to:

- Store and process data in volumes too large for a traditional database.
- Transform unstructured data for analysis and reporting.
- Capture, process, and analyze unbounded streams of data in real time, or with low latency.

## Components of a big data architecture

The following diagram shows the logical components that fit into a big data architecture. Individual solutions may not contain every item in this diagram.



Most big data architectures include some or all of the following components:

- **Data sources.** All big data solutions start with one or more data sources. Examples include:
  - Application data stores, such as relational databases.
  - Static files produced by applications, such as web server log files.
  - Real-time data sources, such as IoT devices.
- **Data storage.** Data for batch processing operations is typically stored in a distributed file store that can hold high volumes of large files in various formats. This kind of store is often called a *data lake*. Options for implementing this storage include Azure Data Lake Store or blob containers in Azure Storage.
- **Batch processing.** Because the data sets are so large, often a big data solution must process data files using long-running batch jobs to filter, aggregate, and otherwise prepare the data for analysis. Usually these jobs involve reading source files, processing them, and writing the output to new files. Options include running U-SQL jobs in Azure Data Lake Analytics, using Hive, Pig, or custom Map/Reduce jobs in an HDInsight Hadoop cluster, or using Java, Scala, or Python programs in an HDInsight Spark cluster.
- **Real-time message ingestion.** If the solution includes real-time sources, the architecture must include a way to capture and store real-time messages for stream processing. This might be a simple data store, where incoming messages are dropped into a folder for processing. However, many solutions need a message ingestion store to act as a buffer for messages, and to support scale-out processing, reliable delivery, and other message queuing semantics. This portion of a streaming architecture is often referred to as stream buffering. Options include Azure Event Hubs, Azure IoT Hub, and Kafka.
- **Stream processing.** After capturing real-time messages, the solution must process them by filtering, aggregating, and otherwise preparing the data for analysis. The processed stream data is then written to an output sink. Azure Stream Analytics provides a managed stream processing service based on perpetually running SQL queries that operate on unbounded streams. You can also use open source Apache streaming technologies like Storm and Spark Streaming in an HDInsight cluster.
- **Analytical data store.** Many big data solutions prepare data for analysis and then serve the processed data in a structured format that can be queried using analytical tools. The analytical data store used to serve these queries can be a Kimball-style relational data warehouse, as seen in most traditional business intelligence (BI) solutions. Alternatively, the data could be presented through a low-latency NoSQL technology such as HBase, or an interactive Hive database that provides a metadata abstraction over data files in the distributed data store. Azure SQL Data Warehouse provides a managed service for large-scale, cloud-based data warehousing. HDInsight supports Interactive Hive, HBase, and Spark SQL, which can also be used to serve data for analysis.
- **Analysis and reporting.** The goal of most big data solutions is to provide insights into the data through analysis and reporting. To empower users to analyze the data, the architecture may include a data modeling layer, such as a multidimensional OLAP cube or tabular data model in Azure Analysis Services. It might also support self-service BI, using the modeling and visualization technologies in Microsoft Power BI or Microsoft Excel. Analysis and reporting can also take the form of interactive data exploration by data scientists or data analysts. For these scenarios, many Azure services support analytical notebooks, such as Jupyter, enabling these users to leverage their existing skills with Python or R. For large-scale data exploration, you can use Microsoft R Server, either standalone or with Spark.
- **Orchestration.** Most big data solutions consist of repeated data processing operations, encapsulated in workflows, that transform source data, move data between multiple sources and sinks, load the processed data into an analytical data store, or push the results straight to a report or dashboard. To automate these workflows, you can use an orchestration technology such Azure Data Factory or Apache Oozie and Sqoop.

## Lambda architecture

When working with very large data sets, it can take a long time to run the sort of queries that clients need. These queries can't be performed in real time, and often require algorithms such as [MapReduce](#) that operate in parallel

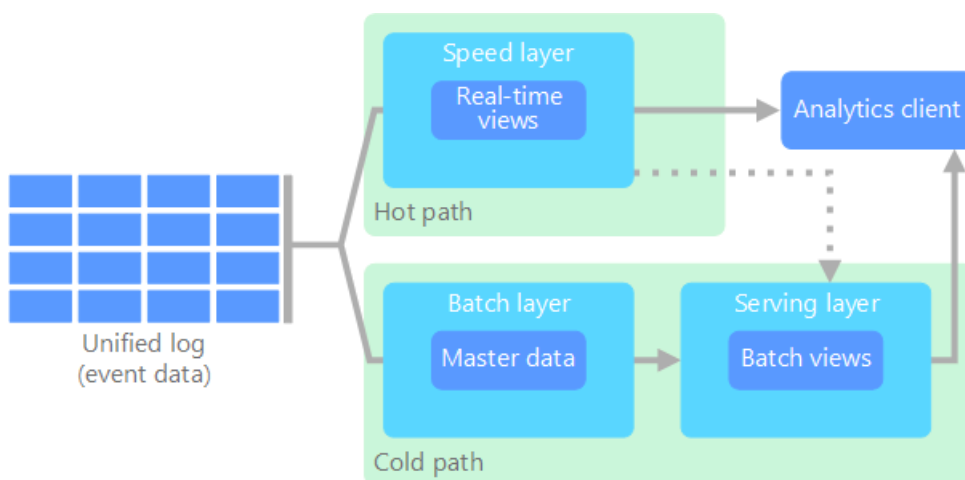
across the entire data set. The results are then stored separately from the raw data and used for querying.

One drawback to this approach is that it introduces latency — if processing takes a few hours, a query may return results that are several hours old. Ideally, you would like to get some results in real time (perhaps with some loss of accuracy), and combine these results with the results from the batch analytics.

The **lambda architecture**, first proposed by Nathan Marz, addresses this problem by creating two paths for data flow. All data coming into the system goes through these two paths:

- A **batch layer** (cold path) stores all of the incoming data in its raw form and performs batch processing on the data. The result of this processing is stored as a **batch view**.
- A **speed layer** (hot path) analyzes data in real time. This layer is designed for low latency, at the expense of accuracy.

The batch layer feeds into a **serving layer** that indexes the batch view for efficient querying. The speed layer updates the serving layer with incremental updates based on the most recent data.



Data that flows into the hot path is constrained by latency requirements imposed by the speed layer, so that it can be processed as quickly as possible. Often, this requires a tradeoff of some level of accuracy in favor of data that is ready as quickly as possible. For example, consider an IoT scenario where a large number of temperature sensors are sending telemetry data. The speed layer may be used to process a sliding time window of the incoming data.

Data flowing into the cold path, on the other hand, is not subject to the same low latency requirements. This allows for high accuracy computation across large data sets, which can be very time intensive.

Eventually, the hot and cold paths converge at the analytics client application. If the client needs to display timely, yet potentially less accurate data in real time, it will acquire its result from the hot path. Otherwise, it will select results from the cold path to display less timely but more accurate data. In other words, the hot path has data for a relatively small window of time, after which the results can be updated with more accurate data from the cold path.

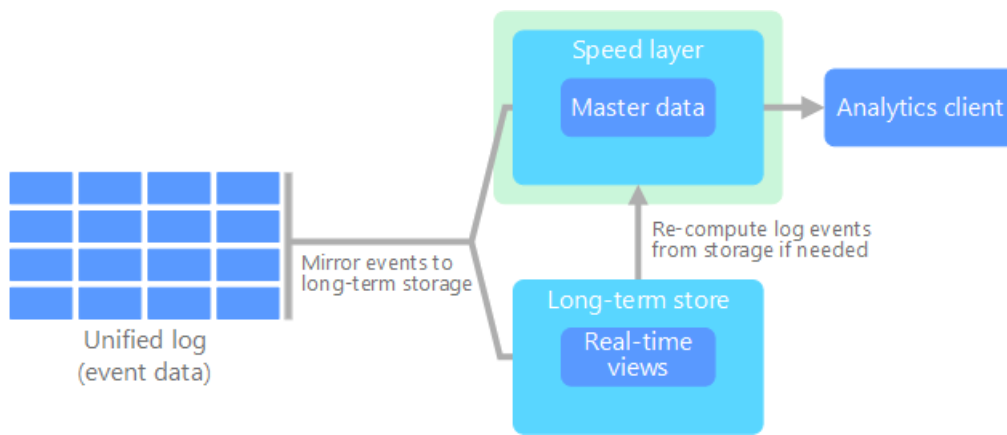
The raw data stored at the batch layer is immutable. Incoming data is always appended to the existing data, and the previous data is never overwritten. Any changes to the value of a particular datum are stored as a new timestamped event record. This allows for recomputation at any point in time across the history of the data collected. The ability to recompute the batch view from the original raw data is important, because it allows for new views to be created as the system evolves.

## Kappa architecture

A drawback to the lambda architecture is its complexity. Processing logic appears in two different places — the cold and hot paths — using different frameworks. This leads to duplicate computation logic and the complexity of managing the architecture for both paths.

The **kappa architecture** was proposed by Jay Kreps as an alternative to the lambda architecture. It has the same

basic goals as the lambda architecture, but with an important distinction: All data flows through a single path, using a stream processing system.



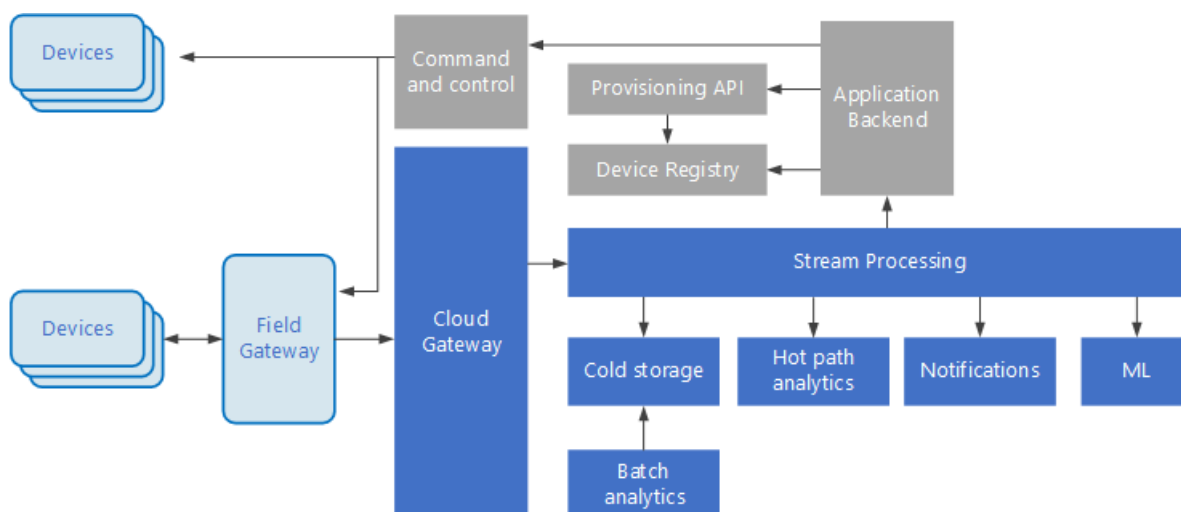
There are some similarities to the lambda architecture's batch layer, in that the event data is immutable and all of it is collected, instead of a subset. The data is ingested as a stream of events into a distributed and fault tolerant unified log. These events are ordered, and the current state of an event is changed only by a new event being appended. Similar to a lambda architecture's speed layer, all event processing is performed on the input stream and persisted as a real-time view.

If you need to recompute the entire data set (equivalent to what the batch layer does in lambda), you simply replay the stream, typically using parallelism to complete the computation in a timely fashion.

## Internet of Things (IoT)

From a practical viewpoint, Internet of Things (IoT) represents any device that is connected to the Internet. This includes your PC, mobile phone, smart watch, smart thermostat, smart refrigerator, connected automobile, heart monitoring implants, and anything else that connects to the Internet and sends or receives data. The number of connected devices grows every day, as does the amount of data collected from them. Often this data is being collected in highly constrained, sometimes high-latency environments. In other cases, data is sent from low-latency environments by thousands or millions of devices, requiring the ability to rapidly ingest the data and process accordingly. Therefore, proper planning is required to handle these constraints and unique requirements.

Event-driven architectures are central to IoT solutions. The following diagram shows a possible logical architecture for IoT. The diagram emphasizes the event-streaming components of the architecture.



The **cloud gateway** ingests device events at the cloud boundary, using a reliable, low latency messaging system.

Devices might send events directly to the cloud gateway, or through a **field gateway**. A field gateway is a specialized device or software, usually collocated with the devices, that receives events and forwards them to the cloud gateway. The field gateway might also preprocess the raw device events, performing functions such as

filtering, aggregation, or protocol transformation.

After ingestion, events go through one or more **stream processors** that can route the data (for example, to storage) or perform analytics and other processing.

The following are some common types of processing. (This list is certainly not exhaustive.)

- Writing event data to cold storage, for archiving or batch analytics.
- Hot path analytics, analyzing the event stream in (near) real time, to detect anomalies, recognize patterns over rolling time windows, or trigger alerts when a specific condition occurs in the stream.
- Handling special types of nontelemetry messages from devices, such as notifications and alarms.
- Machine learning.

The boxes that are shaded gray show components of an IoT system that are not directly related to event streaming, but are included here for completeness.

- The **device registry** is a database of the provisioned devices, including the device IDs and usually device metadata, such as location.
- The **provisioning API** is a common external interface for provisioning and registering new devices.
- Some IoT solutions allow **command and control messages** to be sent to devices.

Relevant Azure services:

- [Azure IoT Hub](#)
- [Azure Event Hubs](#)
- [Azure Stream Analytics](#)

Learn more about IoT on Azure by reading the [Azure IoT reference architecture](#).

# Advanced analytics

2/27/2018 • 6 min to read • [Edit Online](#)

Advanced analytics goes beyond the historical reporting and data aggregation of traditional business intelligence (BI), and uses mathematical, probabilistic, and statistical modeling techniques to enable predictive processing and automated decision making.

Advanced analytics solutions typically involve the following workloads:

- Interactive data exploration and visualization
- Machine Learning model training
- Real-time or batch predictive processing

Most advanced analytics architectures include some or all of the following components:

- **Data storage.** Advanced analytics solutions require data to train machine learning models. Data scientists typically need to explore the data to identify its predictive features and the statistical relationships between them and the values they predict (known as a label). The predicted label can be a quantitative value, like the financial value of something in the future or the duration of a flight delay in minutes. Or it might represent a categorical class, like "true" or "false," "flight delay" or "no flight delay," or categories like "low risk," "medium risk," or "high risk."
- **Batch processing.** To train a machine learning model, you typically need to process a large volume of training data. Training the model can take some time (on the order of minutes to hours). This training can be performed using scripts written in languages such as Python or R, and can be scaled out to reduce training time using distributed processing platforms like Apache Spark hosted in HDInsight or a Docker container.
- **Real-time message ingestion.** In production, many advanced analytics feed real-time data streams to a predictive model that has been published as a web service. The incoming data stream is typically captured in some form of queue and a stream processing engine pulls the data from this queue and applies the prediction to the input data in near real time.
- **Stream processing.** Once you have a trained model, prediction (or scoring) is typically a very fast operation (on the order of milliseconds) for a given set of features. After capturing real-time messages, the relevant feature values can be passed to the predictive service to generate a predicted label.
- **Analytical data store.** In some cases, the predicted label values are written to the analytical data store for reporting and future analysis.
- **Analysis and reporting.** As the name suggests, advanced analytics solutions usually produce some sort of report or analytical feed that includes predicted data values. Often, predicted label values are used to populate real-time dashboards.
- **Orchestration.** Although the initial data exploration and modeling is performed interactively by data scientists, many advanced analytics solutions periodically re-train models with new data — continually refining the accuracy of the models. This retraining can be automated using an orchestrated workflow.

## Machine learning

Machine learning is a mathematical modeling technique used to train a predictive model. The general principle is to apply a statistical algorithm to a large dataset of historical data to uncover relationships between the fields it contains.

Machine learning modeling is usually performed by data scientists, who need to thoroughly explore and prepare the data before training a model. This exploration and preparation typically involves a great deal of interactive data analysis and visualization — usually using languages such as Python and R in interactive tools and environments that are specifically designed for this task.

In some cases, you may be able to use [pretrained models](#) that come with training data obtained and developed by Microsoft. The advantage of pretrained models is that you can score and classify new content right away, even if you don't have the necessary training data, the resources to manage large datasets or to train complex models.

There are two broad categories of machine learning:

- **Supervised learning.** Supervised learning is the most common approach taken by machine learning. In a supervised learning model, the source data consists of a set of *feature* data fields that have a mathematical relationship with one or more *label* data fields. During the training phase of the machine learning process, the data set includes both features and known labels, and an algorithm is applied to fit a function that operates on the features to calculate the corresponding label predictions. Typically, a subset of the training dataset is held back and used to validate the performance of the trained model. Once the model has been trained, it can be deployed into production, and used to predict unknown values.
- **Unsupervised learning.** In an unsupervised learning model, the training data does not include known label values. Instead, the algorithm makes its predictions based on its first exposure to the data. The most common form of unsupervised learning is *clustering*, where the algorithm determines the best way to split the data into a specified number of clusters based on statistical similarities in the features. In clustering, the predicted outcome is the cluster number to which the input features belong. While they can sometimes be used directly to generate useful predictions, such as using clustering to identify groups of users in a database of customers, unsupervised learning approaches are more often used to identify which data is most useful to provide to a supervised learning algorithm in training a model.

Relevant Azure services:

- [Azure Machine Learning](#)
- [Machine Learning Server \(R Server\) on HDInsight](#)

## Deep learning

Machine learning models based on mathematical techniques like linear or logistic regression have been available for some time. More recently, the use of *deep learning* techniques based on neural networks has increased. This is driven partly by the availability of highly scalable processing systems that reduce how long it takes to train complex models. Also, the increased prevalence of big data makes it easier to train deep learning models in a variety of domains.

When designing a cloud architecture for advanced analytics, you should consider the need for large-scale processing of deep learning models. These can be provided through distributed processing platforms like Apache Spark and the latest generation of virtual machines that include access to GPU hardware.

Relevant Azure services:

- [Deep Learning Virtual Machine](#)
- [Apache Spark on HDInsight](#)

## Artificial intelligence

Artificial intelligence (AI) refers to scenarios where a machine mimics the cognitive functions associated with human minds, such as learning and problem solving. Because AI leverages machine learning algorithms, it is viewed as an umbrella term. Most AI solutions rely on a combination of predictive services, often implemented as web services, and natural language interfaces, such as chatbots that interact via text or speech, that are presented



by AI apps running on mobile devices or other clients. In some cases, the machine learning model is embedded with the AI app.

## Model deployment

The predictive services that support AI applications may leverage custom machine learning models, or off-the-shelf cognitive services that provide access to pretrained models. The process of deploying custom models into production is known as operationalization, where the same AI models that are trained and tested within the processing environment are serialized and made available to external applications and services for batch or self-service predictions. To use the predictive capability of the model, it is deserialized and loaded using the same machine learning library that contains the algorithm that was used to train the model in the first place. This library provides predictive functions (often called score or predict) that take the model and features as input and return the prediction. This logic is then wrapped in a function that an application can call directly or can be exposed as a web service.

Relevant Azure services:

- [Azure Machine Learning](#)
- [Machine Learning Server \(R Server\) on HDInsight](#)

## See also

- [Choosing a cognitive services technology](#)
- [Choosing a machine learning technology](#)

# Machine learning at scale

2/13/2018 • 3 min to read • [Edit Online](#)

Machine learning (ML) is a technique used to train predictive models based on mathematical algorithms. Machine learning analyzes the relationships between data fields to predict unknown values.

Creating and deploying a machine learning model is an iterative process:

- Data scientists explore the source data to determine relationships between *features* and predicted *labels*.
- The data scientists train and validate models based on appropriate algorithms to find the optimal model for prediction.
- The optimal model is deployed into production, as a web service or some other encapsulated function.
- As new data is collected, the model is periodically retrained to improve its effectiveness.

Machine learning at scale addresses two different scalability concerns. The first is training a model against large data sets that require the scale-out capabilities of a cluster to train. The second centers is operationalizing the learned model in a way that can scale to meet the demands of the applications that consume it. Typically this is accomplished by deploying the predictive capabilities as a web service that can then be scaled out.

Machine learning at scale has the benefit that it can produce powerful, predictive capabilities because better models typically result from more data. Once a model is trained, it can be deployed as a stateless, highly-performant, scale-out web service.

## Model preparation and training

During the model preparation and training phase, data scientists explore the data interactively using languages like Python and R to:

- Extract samples from high volume data stores.
- Find and treat outliers, duplicates, and missing values to clean the data.
- Determine correlations and relationships in the data through statistical analysis and visualization.
- Generate new calculated features that improve the predictiveness of statistical relationships.
- Train ML models based on predictive algorithms.
- Validate trained models using data that was withheld during training.

To support this interactive analysis and modeling phase, the data platform must enable data scientists to explore data using a variety of tools. Additionally, the training of a complex machine learning model can require a lot of intensive processing of high volumes of data, so sufficient resources for scaling out the model training is essential.

## Model deployment and consumption

When a model is ready to be deployed, it can be encapsulated as a web service and deployed in the cloud, to an edge device, or within an enterprise ML execution environment. This deployment process is referred to as operationalization.

## Challenges

Machine learning at scale produces a few challenges:

- You typically need a lot of data to train a model, especially for deep learning models.
- You need to prepare these big data sets before you can even begin training your model.

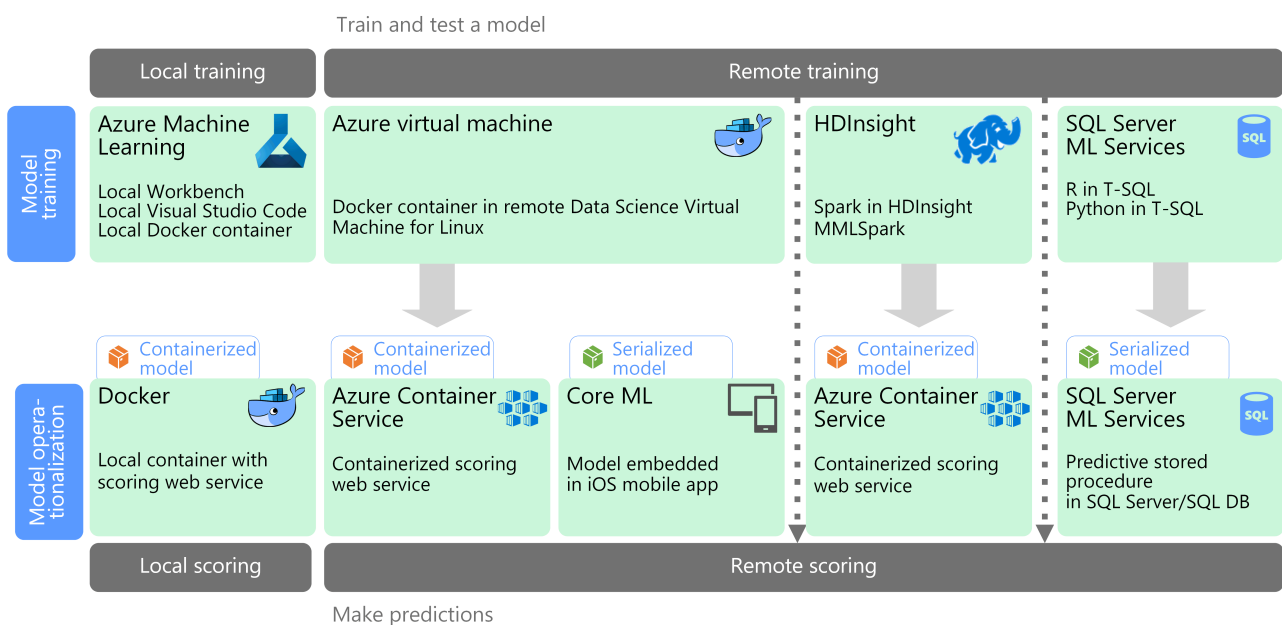
- The model training phase must access the big data stores. It's common to perform the model training using the same big data cluster, such as Spark, that is used for data preparation.
- For scenarios such as deep learning, not only will you need a cluster that can provide you scale out on CPUs, but your cluster will need to consist of GPU-enabled nodes.

## Machine learning at scale in Azure

Before deciding which ML services to use in training and operationalization, consider whether you need to train a model at all, or if a prebuilt model can meet your requirements. In many cases, using a prebuilt model is just a matter of calling a web service or using an ML library to load an existing model. Some options include:

- Use the web services provided by Microsoft Cognitive Services.
- Use the pretrained neural network models provided by Cognitive Toolkit.
- Embed the serialized models provided by Core ML for an iOS apps.

If a prebuilt model does not fit your data or your scenario, options in Azure include Azure Machine Learning, HDInsight with Spark MLlib and MMLSpark, Cognitive Toolkit, and SQL Machine Learning Services. If you decide to use a custom model, you must design a pipeline that includes model training and operationalization.



For a list of technology choices for ML in Azure, see the following topics:

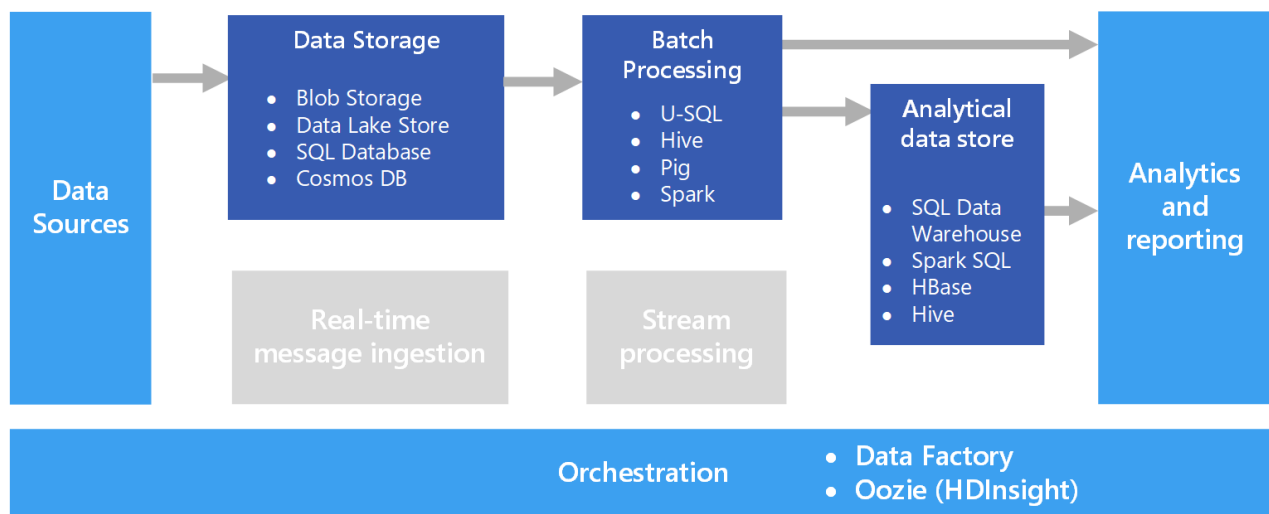
- [Choosing a cognitive services technology](#)
- [Choosing a machine learning technology](#)
- [Choosing a natural language processing technology](#)

# Batch processing

3/4/2018 • 6 min to read • [Edit Online](#)

A common big data scenario is batch processing of data at rest. In this scenario, the source data is loaded into data storage, either by the source application itself or by an orchestration workflow. The data is then processed in-place by a parallelized job, which can also be initiated by the orchestration workflow. The processing may include multiple iterative steps before the transformed results are loaded into an analytical data store, which can be queried by analytics and reporting components.

For example, the logs from a web server might be copied to a folder and then processed overnight to generate daily reports of web activity.



## When to use this solution

Batch processing is used in a variety of scenarios, from simple data transformations to a more complete ETL (extract-transform-load) pipeline. In a big data context, batch processing may operate over very large data sets, where the computation takes significant time. (For example, see [Lambda architecture](#).) Batch processing typically leads to further interactive exploration, provides the modeling-ready data for machine learning, or writes the data to a data store that is optimized for analytics and visualization.

One example of batch processing is transforming a large set of flat, semi-structured CSV or JSON files into a schematized and structured format that is ready for further querying. Typically the data is converted from the raw formats used for ingestion (such as CSV) into binary formats that are more performant for querying because they store data in a columnar format, and often provide indexes and inline statistics about the data.

## Challenges

- **Data format and encoding.** Some of the most difficult issues to debug happen when files use an unexpected format or encoding. For example, source files might use a mix of UTF-16 and UTF-8 encoding, or contain unexpected delimiters (space versus tab), or include unexpected characters. Another common example is text fields that contain tabs, spaces, or commas that are interpreted as delimiters. Data loading and parsing logic must be flexible enough to detect and handle these issues.
- **Orchestrating time slices.** Often source data is placed in a folder hierarchy that reflects processing windows, organized by year, month, day, hour, and so on. In some cases, data may arrive late. For example, suppose that a web server fails, and the logs for March 7th don't end up in the folder for processing until March 9th. Are they just ignored because they're too late? Can the downstream processing logic handle

out-of-order records?

## Architecture

A batch processing architecture has the following logical components, shown in the diagram above.

- **Data storage.** Typically a distributed file store that can serve as a repository for high volumes of large files in various formats. Generically, this kind of store is often referred to as a data lake.
- **Batch processing.** The high-volume nature of big data often means that solutions must process data files using long-running batch jobs to filter, aggregate, and otherwise prepare the data for analysis. Usually these jobs involve reading source files, processing them, and writing the output to new files.
- **Analytical data store.** Many big data solutions are designed to prepare data for analysis and then serve the processed data in a structured format that can be queried using analytical tools.
- **Analysis and reporting.** The goal of most big data solutions is to provide insights into the data through analysis and reporting.
- **Orchestration.** With batch processing, typically some orchestration is required to migrate or copy the data into your data storage, batch processing, analytical data store, and reporting layers.

## Technology choices

The following technologies are recommended choices for batch processing solutions in Azure.

### Data storage

- **Azure Storage Blob Containers.** Many existing Azure business processes already make use of Azure blob storage, making this a good choice for a big data store.
- **Azure Data Lake Store.** Azure Data Lake Store offers virtually unlimited storage for any size of file, and extensive security options, making it a good choice for extremely large-scale big data solutions that require a centralized store for data in heterogeneous formats.

For more information, see [Data storage](#).

### Batch processing

- **U-SQL.** U-SQL is the query processing language used by Azure Data Lake Analytics. It combines the declarative nature of SQL with the procedural extensibility of C#, and takes advantage of parallelism to enable efficient processing of data at massive scale.
- **Hive.** Hive is a SQL-like language that is supported in most Hadoop distributions, including HDInsight. It can be used to process data from any HDFS-compatible store, including Azure blob storage and Azure Data Lake Store.
- **Pig.** Pig is a declarative big data processing language used in many Hadoop distributions, including HDInsight. It is particularly useful for processing data that is unstructured or semi-structured.
- **Spark.** The Spark engine supports batch processing programs written in a range of languages, including Java, Scala, and Python. Spark uses a distributed architecture to process data in parallel across multiple worker nodes.

For more information, see [Batch processing](#).

### Analytical data store

- **SQL Data Warehouse.** Azure SQL Data Warehouse is a managed service based on SQL Server database technologies and optimized to support large-scale data warehousing workloads.
- **Spark SQL.** Spark SQL is an API built on Spark that supports the creation of dataframes and tables that can be queried using SQL syntax.
- **HBase.** HBase is a low-latency NoSQL store that offers a high-performance, flexible option for querying

structured and semi-structured data.

- **Hive.** In addition to being useful for batch processing, Hive offers a database architecture that is conceptually similar to that of a typical relational database management system. Improvements in Hive query performance through innovations like the Tez engine and Stinger initiative mean that Hive tables can be used effectively as sources for analytical queries in some scenarios.

For more information, see [Analytical data stores](#).

### Analytics and reporting

- **Azure Analysis Services.** Many big data solutions emulate traditional enterprise business intelligence architectures by including a centralized online analytical processing (OLAP) data model (often referred to as a cube) on which reports, dashboards, and interactive “slice and dice” analysis can be based. Azure Analysis Services supports the creation of multidimensional and tabular models to meet this need.
- **Power BI.** Power BI enables data analysts to create interactive data visualizations based on data models in an OLAP model or directly from an analytical data store.
- **Microsoft Excel.** Microsoft Excel is one of the most widely used software applications in the world, and offers a wealth of data analysis and visualization capabilities. Data analysts can use Excel to build document data models from analytical data stores, or to retrieve data from OLAP data models into interactive PivotTables and charts.

For more information, see [Analytics and reporting](#).

### Orchestration

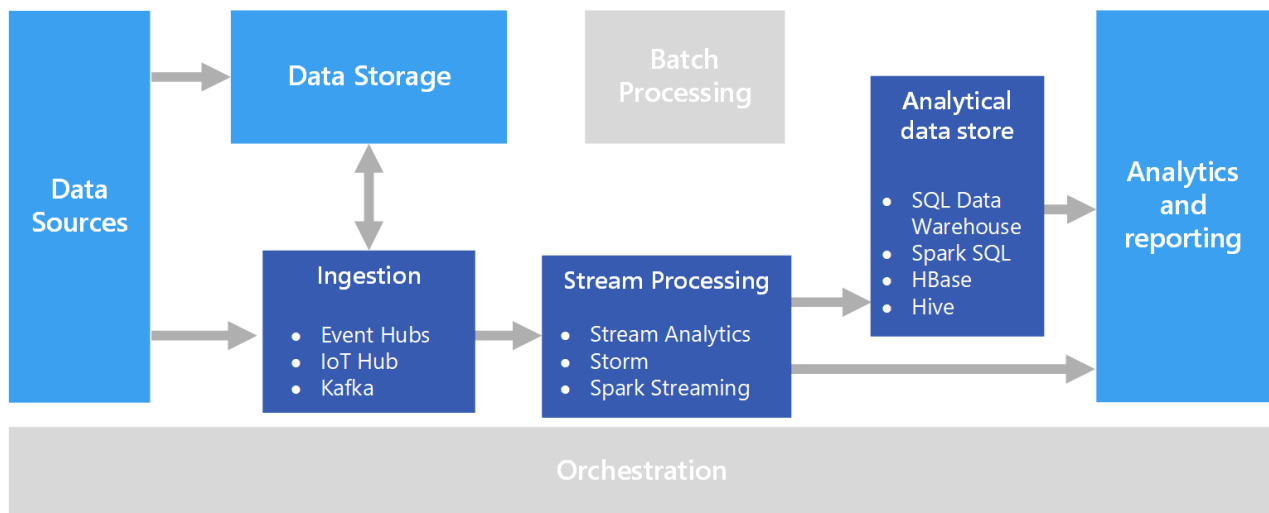
- **Azure Data Factory.** Azure Data Factory pipelines can be used to define a sequence of activities, scheduled for recurring temporal windows. These activities can initiate data copy operations as well as Hive, Pig, MapReduce, or Spark jobs in on-demand HDInsight clusters; U-SQL jobs in Azure Data Lake Analytics; and stored procedures in Azure SQL Data Warehouse or Azure SQL Database.
- **Oozie and Sqoop.** Oozie is a job automation engine for the Apache Hadoop ecosystem and can be used to initiate data copy operations as well as Hive, Pig, and MapReduce jobs to process data and Sqoop jobs to copy data between HDFS and SQL databases.

For more information, see [Pipeline orchestration](#)

# Real time processing

2/13/2018 • 4 min to read • [Edit Online](#)

Real time processing deals with streams of data that are captured in real-time and processed with minimal latency to generate real-time (or near-real-time) reports or automated responses. For example, a real-time traffic monitoring solution might use sensor data to detect high traffic volumes. This data could be used to dynamically update a map to show congestion, or automatically initiate high-occupancy lanes or other traffic management systems.



Real-time processing is defined as the processing of unbounded stream of input data, with very short latency requirements for processing — measured in milliseconds or seconds. This incoming data typically arrives in an unstructured or semi-structured format, such as JSON, and has the same processing requirements as [batch processing](#), but with shorter turnaround times to support real-time consumption.

Processed data is often written to an analytical data store, which is optimized for analytics and visualization. The processed data can also be ingested directly into the analytics and reporting layer for analysis, business intelligence, and real-time dashboard visualization.

## Challenges

One of the big challenges of real-time processing solutions is to ingest, process, and store messages in real time, especially at high volumes. Processing must be done in such a way that it does not block the ingestion pipeline. The data store must support high-volume writes. Another challenge is being able to act on the data quickly, such as generating alerts in real time or presenting the data in a real-time (or near-real-time) dashboard.

## Architecture

A real-time processing architecture has the following logical components.

- **Real-time message ingestion.** The architecture must include a way to capture and store real-time messages to be consumed by a stream processing consumer. In simple cases, this service could be implemented as a simple data store in which new messages are deposited in a folder. But often the solution requires a message broker, such as Azure Event Hubs, that acts as a buffer for the messages. The message broker should support scale-out processing and reliable delivery.
- **Stream processing.** After capturing real-time messages, the solution must process them by filtering, aggregating, and otherwise preparing the data for analysis.

- **Analytical data store.** Many big data solutions are designed to prepare data for analysis and then serve the processed data in a structured format that can be queried using analytical tools.
- **Analysis and reporting.** The goal of most big data solutions is to provide insights into the data through analysis and reporting.

## Technology choices

The following technologies are recommended choices for real-time processing solutions in Azure.

### Real-time message ingestion

- **Azure Event Hubs.** Azure Event Hubs is a message queuing solution for ingesting millions of event messages per second. The captured event data can be processed by multiple consumers in parallel.
- **Azure IoT Hub.** Azure IoT Hub provides bi-directional communication between Internet-connected devices, and a scalable message queue that can handle millions of simultaneously connected devices.
- **Apache Kafka.** Kafka is an open source message queuing and stream processing application that can scale to handle millions of messages per second from multiple message producers, and route them to multiple consumers. Kafka is available in Azure as an HDInsight cluster type.

For more information, see [Real-time message ingestion](#).

### Data storage

- **Azure Storage Blob Containers or Azure Data Lake Store.** Incoming real-time data is usually captured in a message broker (see above), but in some scenarios, it can make sense to monitor a folder for new files and process them as they are created or updated. Additionally, many real-time processing solutions combine streaming data with static reference data, which can be stored in a file store. Finally, file storage may be used as an output destination for captured real-time data for archiving, or for further batch processing in a [lambda architecture](#).

For more information, see [Data storage](#).

### Stream processing

- **Azure Stream Analytics.** Azure Stream Analytics can run perpetual queries against an unbounded stream of data. These queries consume streams of data from storage or message brokers, filter and aggregate the data based on temporal windows, and write the results to sinks such as storage, databases, or directly to reports in Power BI.
- **Storm.** Apache Storm is an open source framework for stream processing that uses a topology of spouts and bolts to consume, process, and output the results from real-time streaming data sources. You can provision Storm in an Azure HDInsight cluster, and implement a topology in Java or C#.
- **Spark Streaming.** Apache Spark is an open source distributed platform for general data processing. Spark provides the Spark Streaming API, in which you can write code in any supported Spark language, including Java, Scala, and Python. Spark 2.0 introduced the Spark Structured Streaming API, which provides a simpler and more consistent programming model. Spark 2.0 is available in an Azure HDInsight cluster.

For more information, see [Stream processing](#).

### Analytical data store

- **SQL Data Warehouse, HBase, Spark, or Hive.** Processed real-time data can be stored in a relational database such as Azure SQL Data Warehouse, a NoSQL store such as HBase, or as files in distributed storage over which Spark or Hive tables can be defined and queried.

For more information, see [Analytical data stores](#).

### Analytics and reporting

- **Azure Analysis Services, Power BI, and Microsoft Excel.** Processed real-time data that is stored in an



analytical data store can be used for historical reporting and analysis in the same way as batch processed data. Additionally, Power BI can be used to publish real-time (or near-real-time) reports and visualizations from analytical data sources where latency is sufficiently low, or in some cases directly from the stream processing output.

For more information, see [Analytics and reporting](#).

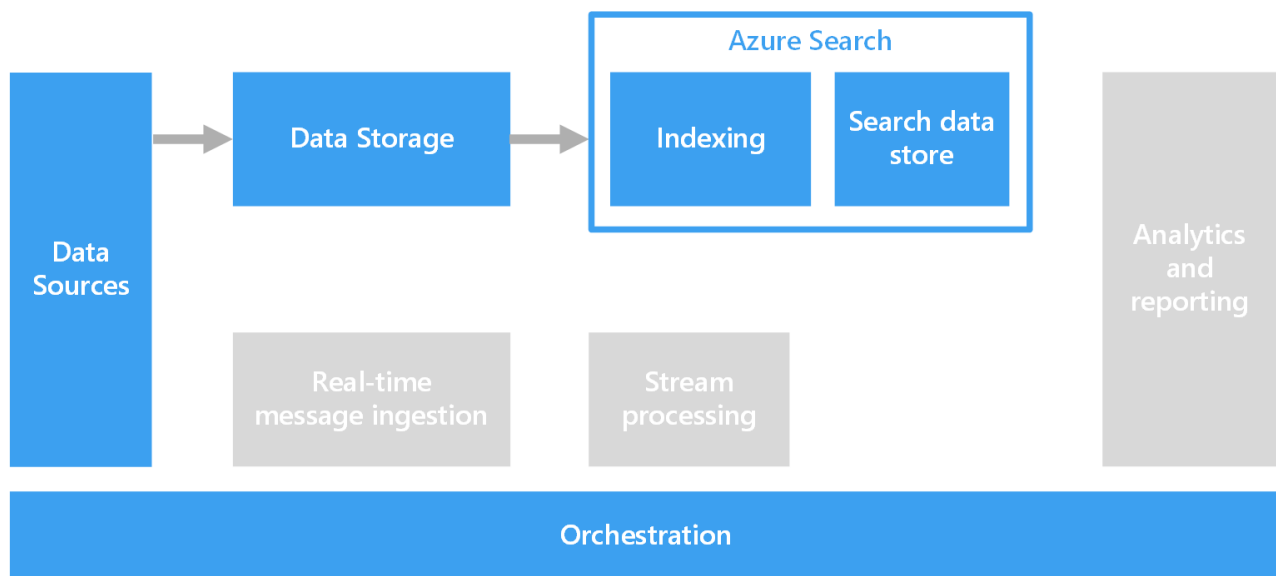
In a purely real-time solution, most of the processing orchestration is managed by the message ingestion and stream processing components. However, in a lambda architecture that combines batch processing and real-time processing, you may need to use an orchestration framework such as Azure Data Factory or Apache Oozie and Sqoop to manage batch workflows for captured real-time data.

# Processing free-form text for search

2/13/2018 • 1 min to read • [Edit Online](#)

To support search, free-form text processing can be performed against documents containing paragraphs of text.

Text search works by constructing a specialized index that is precomputed against a collection of documents. A client application submits a query that contains the search terms. The query returns a result set, consisting of a list of documents sorted by how well each document matches the search criteria. The result set may also include the context in which the document matches the criteria, which enables the application to highlight the matching phrase in the document.



Free-form text processing can produce useful, actionable data from large amounts of noisy text data. The results can give unstructured documents a well-defined and queryable structure.

## Challenges

- Processing a collection of free-form text documents is typically computationally intensive, as well as time intensive.
- In order to search free-form text effectively, the search index should support fuzzy search based on terms that have a similar construction. For example, search indexes are built with lemmatization and linguistic stemming, so that queries for "run" will match documents that contain "ran" and "running."

## Architecture

In most scenarios, the source text documents are loaded into object storage such as Azure Storage or Azure Data Lake Store. An exception is using full text search within SQL Server or Azure SQL Database. In this case, the document data is loaded into tables managed by the database. Once stored, the documents are processed in a batch to create the index.

## Technology choices

Options for creating a search index include Azure Search, Elasticsearch, and HDInsight with Solr. Each of these technologies can populate a search index from a collection of documents. Azure Search provides indexers that can automatically populate the index for documents ranging from plain text to Excel and PDF formats. On HDInsight, Apache Solr can index binary files of many types, including plain text, Word, and PDF. Once the index is

constructed, clients can access the search interface by means of a REST API.

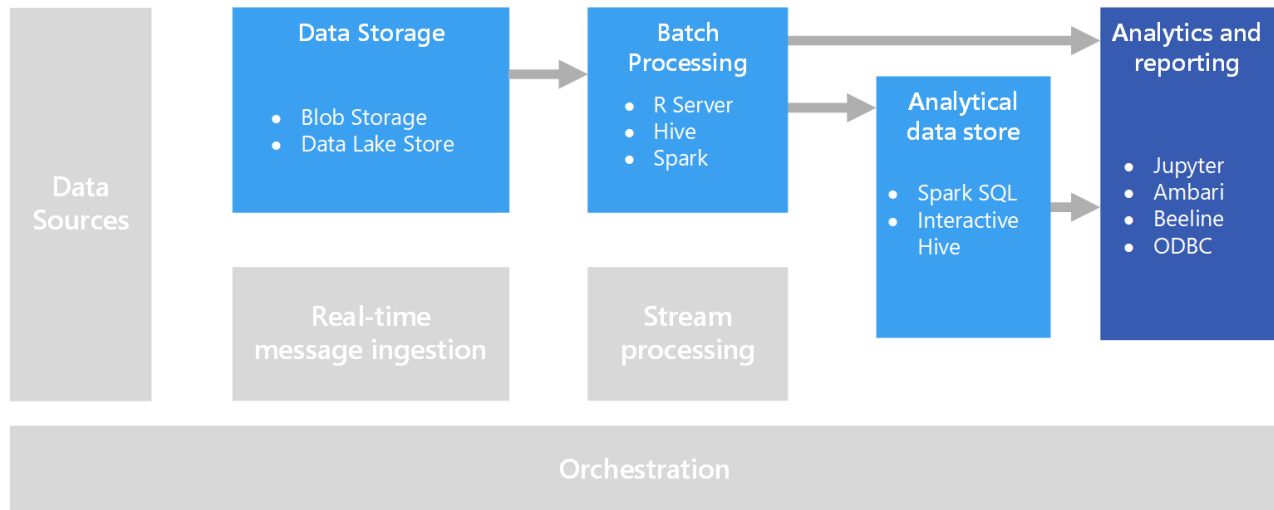
If your text data is stored in SQL Server or Azure SQL Database, you can use the full-text search that is built into the database. The database populates the index from text, binary, or XML data stored within the same database. Clients search by using T-SQL queries.

For more information, see [Search data stores](#).

# Interactive data exploration

2/13/2018 • 4 min to read • [Edit Online](#)

In many corporate business intelligence (BI) solutions, reports and semantic models are created by BI specialists and managed centrally. Increasingly, however, organizations want to enable users to make data-driven decisions. Additionally, a growing number of organizations are hiring *data scientists* or *data analysts*, whose job is to explore data interactively and apply statistical models and analytical techniques to find trends and patterns in the data. Interactive data exploration requires tools and platforms that provide low-latency processing for ad-hoc queries and data visualizations.



## Self-service BI

Self-service BI is a name given to a modern approach to business decision making in which users are empowered to find, explore, and share insights from data across the enterprise. To accomplish this, the data solution must support several requirements:

- Discovery of business data sources through a data catalog.
- Master data management to ensure consistency of data entity definitions and values.
- Interactive data modeling and visualization tools for business users.

In a self-service BI solution, business users typically find and consume data sources that are relevant to their particular area of the business, and use intuitive tools and productivity applications to define personal data models and reports that they can share with their colleagues.

Relevant Azure services:

- [Azure Data Catalog](#)
- [Microsoft Power BI](#)

## Data science experimentation

When an organization requires advanced analytics and predictive modeling, the initial preparation work is usually undertaken by specialist data scientists. A data scientist explores the data and applies statistical analytical techniques to find relationships between data *features* and the desired predicted *labels*. Data exploration is typically done using programming languages such as Python or R that natively support statistical modeling and visualization. The scripts used to explore the data are typically hosted in specialized environments such as Jupyter Notebooks. These tools enable data scientists to explore the data programmatically while documenting and sharing

the insights they find.

Relevant Azure services:

- [Azure Notebooks](#)
- [Azure Machine Learning Studio](#)
- [Azure Machine Learning Experimentation Services](#)
- [The Data Science Virtual Machine](#)

## Challenges

- **Data privacy compliance.** You need to be careful about making personal data available to users for self-service analysis and reporting. There are likely to be compliance considerations, due to organizational policies and also regulatory issues.
- **Data volume.** While it may be useful to give users access to the full data source, it can result in very long-running Excel or Power BI operations, or Spark SQL queries that use a lot of cluster resources.
- **User knowledge.** Users create their own queries and aggregations in order to inform business decisions. Are you confident that users have the necessary analytical and querying skills to get accurate results?
- **Sharing results.** There may be security considerations if users can create and share reports or data visualizations.

## Architecture

Although the goal of this scenario is to support interactive data analysis, the data cleansing, sampling, and structuring tasks involved in data science often include long-running processes. That makes a [batch processing](#) architecture appropriate.

## Technology choices

The following technologies are recommended choices for interactive data exploration in Azure.

### Data storage

- **Azure Storage Blob Containers** or **Azure Data Lake Store.** Data scientists generally work with raw source data, to ensure they have access to all possible features, outliers, and errors in the data. In a big data scenario, this data usually takes the form of files in a data store.

For more information, see [Data storage](#).

### Batch processing

- **R Server** or **Spark.** Most data scientists use programming languages with strong support for mathematical and statistical packages, such as R or Python. When working with large volumes of data, you can reduce latency by using platforms that enable these languages to use distributed processing. R Server can be used on its own or in conjunction with Spark to scale out R processing functions, and Spark natively supports Python for similar scale-out capabilities in that language.
- **Hive.** Hive is a good choice for transforming data using SQL-like semantics. Users can create and load tables using HiveQL statements, which are semantically similar to SQL.

For more information, see [Batch processing](#).

### Analytical Data Store

- **Spark SQL.** Spark SQL is an API built on Spark that supports the creation of dataframes and tables that can be queried using SQL syntax. Regardless of whether the data files to be analyzed are raw source files or new files that have been cleaned and prepared by a batch process, users can define Spark SQL tables on them for further

querying an analysis.

- **Hive.** In addition to batch processing raw data by using Hive, you can create a Hive database that contains Hive tables and views based on the folders where the data is stored, enabling interactive queries for analysis and reporting. HDInsight includes an Interactive Hive cluster type that uses in-memory caching to reduce Hive query response times. Users who are comfortable with SQL-like syntax can use Interactive Hive to explore data.

For more information, see [Analytical data stores](#).

### **Analytics and reporting**

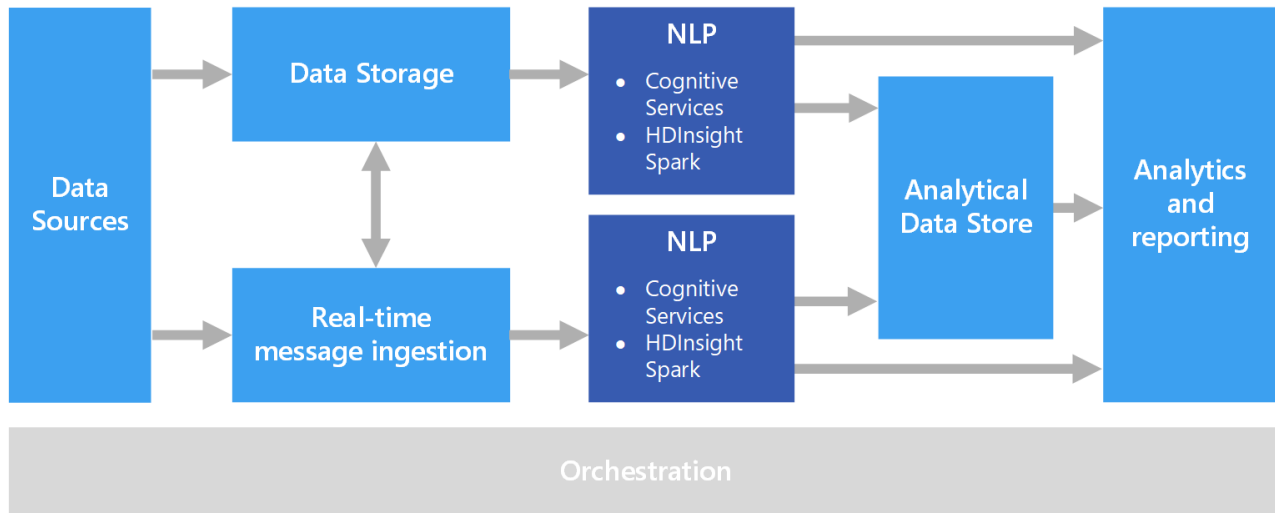
- **Jupyter.** Jupyter Notebooks provides a browser-based interface for running code in languages such as R, Python, or Scala. When using R Server or Spark to batch process data, or when using Spark SQL to define a schema of tables for querying, Jupyter can be a good choice for querying the data. When using Spark, you can use the standard Spark dataframe API or the Spark SQL API as well as embedded SQL statements to query the data and produce visualizations.
- **Interactive Hive Clients.** If you use an Interactive Hive cluster to query the data, you can use the Hive view in the Ambari cluster dashboard, the Beeline command line tool, or any ODBC-based tool (using the Hive ODBC driver), such as Microsoft Excel or Power BI.

For more information, see [Data analytics and reporting technology](#).

# Natural language processing

2/13/2018 • 2 min to read • [Edit Online](#)

Natural language processing (NLP) is used for tasks such as sentiment analysis, topic detection, language detection, key phrase extraction, and document categorization.



## When to use this solution

NLP can be used to classify documents, such as labeling documents as sensitive or spam. The output of NLP can be used for subsequent processing or search. Another use for NLP is to summarize text by identifying the entities present in the document. These entities can also be used to tag documents with keywords, which enables search and retrieval based on content. Entities might be combined into topics, with summaries that describe the important topics present in each document. The detected topics may be used to categorize the documents for navigation, or to enumerate related documents given a selected topic. Another use for NLP is to score text for sentiment, to assess the positive or negative tone of a document. These approaches use many techniques from natural language processing, such as:

- **Tokenizer.** Splitting the text into words or phrases.
- **Stemming and lemmatization.** Normalizing words so that different forms map to the canonical word with the same meaning. For example, "running" and "ran" map to "run."
- **Entity extraction.** Identifying subjects in the text.
- **Part of speech detection.** Identifying text as a verb, noun, participle, verb phrase, and so on.
- **Sentence boundary detection.** Detecting complete sentences within paragraphs of text.

When using NLP to extract information and insight from free-form text, the starting point is typically the raw documents stored in object storage such as Azure Storage or Azure Data Lake Store.

## Challenges

- Processing a collection of free-form text documents is typically computationally resource intensive, as well as being time intensive.
- Without a standardized document format, it can be very difficult to achieve consistently accurate results using free-form text processing to extract specific facts from a document. For example, think of a text representation of an invoice—it can be difficult to build a process that correctly extracts the invoice number and invoice date for invoices across any number of vendors.

# Architecture

In an NLP solution, free-form text processing is performed against documents containing paragraphs of text. The overall architecture can be a [batch processing](#) or [real-time stream processing](#) architecture.

The actual processing varies based on the desired outcome, but in terms of the pipeline, NLP may be applied in a batch or real-time fashion. For example, sentiment analysis can be used against blocks of text to produce a sentiment score. This can be done by running a batch process against data in storage, or in real time using smaller chunks of data flowing through a messaging service.

## Technology choices

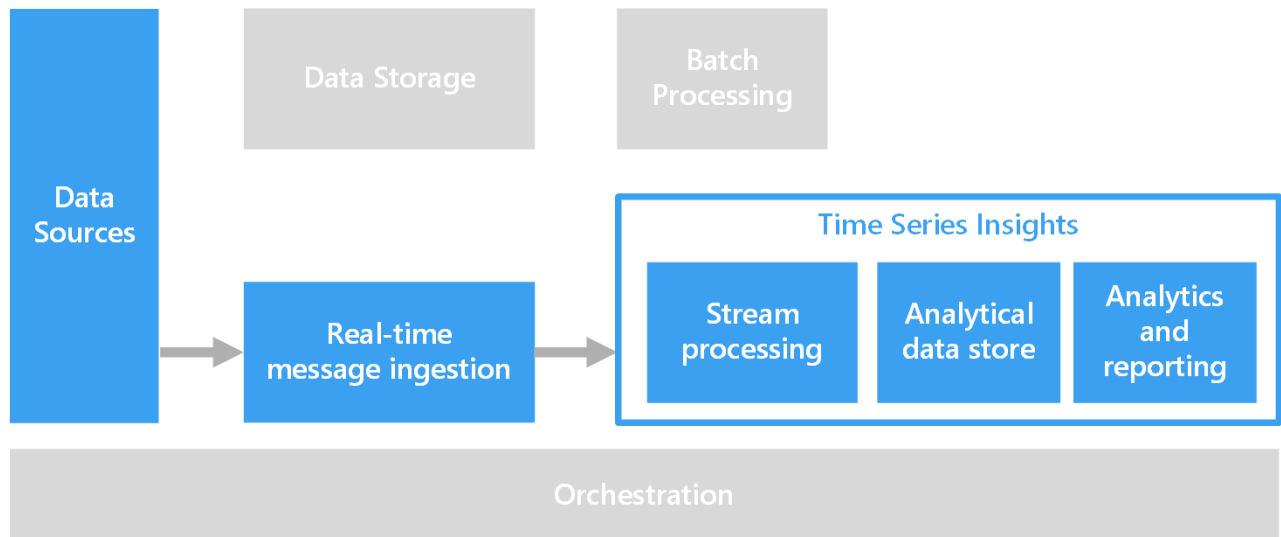
- [Natural language processing](#)



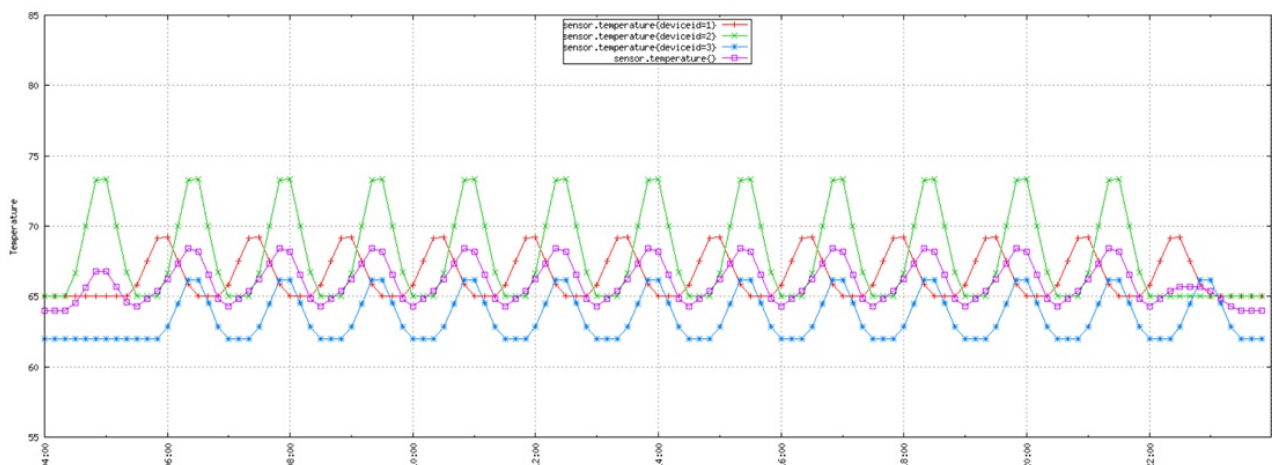
# Time series solutions

2/13/2018 • 4 min to read • [Edit Online](#)

Time series data is a set of values organized by time. Examples of time series data include sensor data, stock prices, click stream data, and application telemetry. Time series data can be analyzed for historical trends, real-time alerts, or predictive modeling.



Time series data represents how an asset or process changes over time. The data has a timestamp, but more importantly, time is the most meaningful axis for viewing or analyzing the data. Time series data typically arrives in order of time and is usually treated as an insert rather than an update to your database. Because of this, change is measured over time, enabling you to look backward and to predict future change. As such, time series data is best visualized with scatter or line charts.



Some examples of time series data are:

- Stock prices captured over time to detect trends.
- Server performance, such as CPU usage, I/O load, memory usage, and network bandwidth consumption.
- Telemetry from sensors on industrial equipment, which can be used to detect pending equipment failure and trigger alert notifications.
- Real-time car telemetry data including speed, braking, and acceleration over a time window to produce an aggregate risk score for the driver.

In each of these cases, you can see how time is most meaningful as an axis. Displaying the events in the order in

which they arrived is a key characteristic of time series data, as there is a natural temporal ordering. This differs from data captured for standard OLTP data pipelines where data can be entered in any order, and updated at any time.

## When to use this solution

Choose a time series solution when you need to ingest data whose strategic value is centered around changes over a period of time, and you are primarily inserting new data and rarely updating, if at all. You can use this information to detect anomalies, visualize trends, and compare current data to historical data, among other things. This type of architecture is also best suited for predictive modeling and forecasting results, because you have the historical record of changes over time, which can be applied to any number of forecasting models.

Using time series offers the following benefits:

- Clearly represents how an asset or process changes over time.
- Helps you quickly detect changes to a number of related sources, making anomalies and emerging trends clearly stand out.
- Best suited for predictive modeling and forecasting.

### Internet of Things (IoT)

Data collected by IoT devices is a natural fit for time series storage and analysis. The incoming data is inserted and rarely, if ever, updated. The data is time stamped and inserted in the order it was received, and this data is typically displayed in chronological order, enabling users to discover trends, spot anomalies, and use the information for predictive analysis.

For more information, see [Internet of Things](#).

### Real-time analytics

Time series data is often time sensitive — that is, it must be acted on quickly, to spot trends in real time or generate alerts. In these scenarios, any delay in insights can cause downtime and business impact. In addition, there is often a need to correlate data from a variety of different sources, such as sensors.

Ideally, you would have a stream processing layer that can handle the incoming data in real time and process all of it with high precision and high granularity. This isn't always possible, depending on your streaming architecture and the components of your stream buffering and stream processing layers. You may need to sacrifice some precision of the time series data by reducing it. This is done by processing sliding time windows (several seconds, for example), allowing the processing layer to perform calculations in a timely manner. You may also need to downsample and aggregate your data when displaying longer periods of time, such as zooming to display data captured over several months.

## Challenges

- Time series data is often very high volume, especially in IoT scenarios. Storing, indexing, querying, analyzing, and visualizing time series data can be challenging.
- It can be challenging to find the right combination of high-speed storage and powerful compute operations for handling real-time analytics, while minimizing time to market and overall cost investment.

## Architecture

In many scenarios that involve time series data, such as IoT, the data is captured in real time. As such, a [real-time processing](#) architecture is appropriate.

Data from one or more data sources is ingested into the stream buffering layer by [IoT Hub](#), [Event Hubs](#), or [Kafka on HDInsight](#). Next, the data is processed in the stream processing layer that can optionally hand off the processed data to a machine learning service for predictive analytics. The processed data is stored in an analytical data store,

such as [HBase](#), [Azure Cosmos DB](#), Azure Data Lake, or Blob Storage. An analytics and reporting application or service, like Power BI or OpenTSDB (if stored in HBase) can be used to display the time series data for analysis.

Another option is to use [Azure Time Series Insights](#). Time Series Insights is a fully managed service for time series data. In this architecture, Time Series Insights performs the roles of stream processing, data store, and analytics and reporting. It accepts streaming data from either IoT Hub or Event Hubs and stores, processes, analyzes, and displays the data in near real time. It does not pre-aggregate the data, but stores the raw events.

Time Series Insights is schema adaptive, which means that you do not have to do any data preparation to start deriving insights. This enables you to explore, compare, and correlate a variety of data sources seamlessly. It also provides SQL-like filters and aggregates, ability to construct, visualize, compare, and overlay various time series patterns, heat maps, and the ability to save and share queries.

## Technology choices

- [Data Storage](#)
- [Analysis, visualizations, and reporting](#)
- [Analytical Data Stores](#)
- [Stream processing](#)

# Choosing an analytical data store in Azure

2/13/2018 • 4 min to read • [Edit Online](#)

In a [big data](#) architecture, there is often a need for an analytical data store that serves processed data in a structured format that can be queried using analytical tools. Analytical data stores that support querying of both hot-path and cold-path data are collectively referred to as the serving layer, or data serving storage.

The serving layer deals with processed data from both the hot path and cold path. In the [lambda architecture](#), the serving layer is subdivided into a *speed serving* layer, which stores data that has been processed incrementally, and a *batch serving* layer, which contains the batch-processed output. The serving layer requires strong support for random reads with low latency. Data storage for the speed layer should also support random writes, because batch loading data into this store would introduce undesired delays. On the other hand, data storage for the batch layer does not need to support random writes, but batch writes instead.

There is no single best data management choice for all data storage tasks. Different data management solutions are optimized for different tasks. Most real-world cloud apps and big data processes have a variety of data storage requirements and often use a combination of data storage solutions.

## What are your options when choosing an analytical data store?

There are several options for data serving storage in Azure, depending on your needs:

- [SQL Data Warehouse](#)
- [Azure SQL Database](#)
- [SQL Server in Azure VM](#)
- [HBase/Phoenix on HDInsight](#)
- [Hive LLAP on HDInsight](#)
- [Azure Analysis Services](#)
- [Azure Cosmos DB](#)

These options provide various database models that are optimized for different types of tasks:

- [Key/value](#) databases hold a single serialized object for each key value. They're good for storing large volumes of data where you want to get one item for a given key value and you don't have to query based on other properties of the item.
- [Document](#) databases are key/value databases in which the values are *documents*. A "document" in this context is a collection of named fields and values. The database typically stores the data in a format such as XML, YAML, JSON, or BSON, but may use plain text. Document databases can query on non-key fields and define secondary indexes to make querying more efficient. This makes a document database more suitable for applications that need to retrieve data based on criteria more complex than the value of the document key. For example, you could query on fields such as product ID, customer ID, or customer name.
- [Column-family](#) databases are key/value data stores that structure data storage into collections of related columns called column families. For example, a census database might have one group of columns for a person's name (first, middle, last), one group for the person's address, and one group for the person's profile information (data of birth, gender). The database can store each column family in a separate partition, while keeping all of the data for one person related to the same key. An application can read a single column family without reading through all of the data for an entity.
- [Graph](#) databases store information as a collection of objects and relationships. A graph database can efficiently perform queries that traverse the network of objects and the relationships between them. For example, the objects might be employees in a human resources database, and you might want to facilitate queries such as

"find all employees who directly or indirectly work for Scott."

## Key selection criteria

To narrow the choices, start by answering these questions:

- Do you need serving storage that can serve as a hot path for your data? If yes, narrow your options to those that are optimized for a speed serving layer.
- Do you need massively parallel processing (MPP) support, where queries are automatically distributed across several processes or nodes? If yes, select an option that supports query scale out.
- Do you prefer to use a relational data store? If so, narrow your options to those with a relational database model. However, note that some non-relational stores support SQL syntax for querying, and tools such as PolyBase can be used to query non-relational data stores.

## Capability matrix

The following tables summarize the key differences in capabilities.

### General capabilities

	SQL DATABASE	SQL DATA WAREHOUSE	HBASE/PHOENIX ON HDINSIGHT	HIVE LLAP ON HDINSIGHT	AZURE ANALYSIS SERVICES	COSMOS DB
Is managed service	Yes	Yes	Yes <sup>1</sup>	Yes <sup>1</sup>	Yes	Yes
Primary database model	Relational (columnar format when using columnstore indexes)	Relational tables with columnar storage	Wide column store	Hive/In-Memory	Tabular/MOLAP semantic models	Document store, graph, key-value store, wide column store
SQL language support	Yes	Yes	Yes (using <a href="#">Phoenix</a> JDBC driver)	Yes	No	Yes
Optimized for speed serving layer	Yes <sup>2</sup>	No	Yes	Yes	No	Yes

[1] With manual configuration and scaling.

[2] Using memory-optimized tables and hash or nonclustered indexes.

### Scalability capabilities

	SQL DATABASE	SQL DATA WAREHOUSE	HBASE/PHOENIX ON HDINSIGHT	HIVE LLAP ON HDINSIGHT	AZURE ANALYSIS SERVICES	COSMOS DB
Redundant regional servers for high availability	Yes	Yes	Yes	No	No	Yes

	SQL DATABASE	SQL DATA WAREHOUSE	HBASE/PHOENIX ON HDINSIGHT	HIVE LLAP ON HDINSIGHT	AZURE ANALYSIS SERVICES	COSMOS DB
Supports query scale out	No	Yes	Yes	Yes	Yes	Yes
Dynamic scalability (scale up)	Yes	Yes	No	No	Yes	Yes
Supports in-memory caching of data	Yes	Yes	No	Yes	Yes	No

### Security capabilities

	SQL DATABASE	SQL DATA WAREHOUSE	HBASE/PHOENIX ON HDINSIGHT	HIVE LLAP ON HDINSIGHT	AZURE ANALYSIS SERVICES	COSMOS DB
Authentication	SQL / Azure Active Directory (Azure AD)	SQL / Azure AD	local / Azure AD <sup>1</sup>	local / Azure AD <sup>1</sup>	Azure AD	database users / Azure AD via access control (IAM)
Data encryption at rest	Yes <sup>2</sup>	Yes <sup>2</sup>	Yes <sup>1</sup>	Yes <sup>1</sup>	Yes	Yes
Row-level security	Yes	No	Yes <sup>1</sup>	Yes <sup>1</sup>	Yes (through object-level security in model)	No
Supports firewalls	Yes	Yes	Yes <sup>3</sup>	Yes <sup>3</sup>	Yes	Yes
Dynamic data masking	Yes	No	Yes <sup>1</sup>	Yes *	No	No

[1] Requires using a [domain-joined HDInsight cluster](#).

[2] Requires using transparent data encryption (TDE) to encrypt and decrypt your data at rest.

[3] When used within an Azure Virtual Network. See [Extend Azure HDInsight using an Azure Virtual Network](#).

# Choosing a data analytics technology in Azure

2/13/2018 • 4 min to read • [Edit Online](#)

The goal of most big data solutions is to provide insights into the data through analysis and reporting. This can include preconfigured reports and visualizations, or interactive data exploration.

## What are your options when choosing a data analytics technology?

There are several options for analysis, visualizations, and reporting in Azure, depending on your needs:

- [Power BI](#)
- [Jupyter Notebooks](#)
- [Zeppelin Notebooks](#)
- [Microsoft Azure Notebooks](#)

### Power BI

[Power BI](#) is a suite of business analytics tools. It can connect to hundreds of data sources, and can be used for ad hoc analysis. See [this list](#) of the currently available data sources. Use [Power BI Embedded](#) to integrate Power BI within your own applications without requiring any additional licensing.

Organizations can use Power BI to produce reports and publish them to the organization. Everyone can create personalized dashboards, with governance and [security built in](#). Power BI uses [Azure Active Directory](#) (Azure AD) to authenticate users who log in to the Power BI service, and uses the Power BI login credentials whenever a user attempts to access resources that require authentication.

### Jupyter Notebooks

[Jupyter Notebooks](#) provide a browser-based shell that lets data scientists create *notebook* files that contain Python, Scala, or R code and markdown text, making it an effective way to collaborate by sharing and documenting code and results in a single document.

Most varieties of HDInsight clusters, such as Spark or Hadoop, come [preconfigured with Jupyter notebooks](#) for interacting with data and submitting jobs for processing. Depending on the type of HDInsight cluster you are using, one or more kernels will be provided for interpreting and running your code. For example, Spark clusters on HDInsight provide Spark-related kernels that you can select from to execute Python or Scala code using the Spark engine.

Jupyter notebooks provide a great environment for analyzing, visualizing, and processing your data prior to building more advanced visualizations with a BI/reporting tool like Power BI.

### Zeppelin Notebooks

[Zeppelin Notebooks](#) are another option for a browser-based shell, similar to Jupyter in functionality. Some HDInsight clusters come [preconfigured with Zeppelin notebooks](#). However, if you are using an [HDInsight Interactive Query](#) (Hive LLAP) cluster, [Zeppelin](#) is currently your only choice of notebook that you can use to run interactive Hive queries. Also, if you are using a [domain-joined HDInsight cluster](#), Zeppelin notebooks are the only type that enables you to assign different user logins to control access to notebooks and the underlying Hive tables.

### Microsoft Azure Notebooks

[Azure Notebooks](#) is an online Jupyter Notebooks-based service that enables data scientists to create, run, and share Jupyter Notebooks in cloud-based libraries. Azure Notebooks provides execution environments for Python 2, Python 3, F#, and R, and provides several charting libraries for visualizing your data, such as ggplot, matplotlib, bokeh, and seaborn.

Unlike Jupyter notebooks running on an HDInsight cluster, which are connected to the cluster's default storage account, Azure Notebooks does not provide any data. You must [load data](#) in a variety of ways, such as downloading data from an online source, interacting with Azure Blobs or Table Storage, connecting to a SQL database, or loading data with the Copy Wizard for Azure Data Factory.

Key benefits:

- Free service—no Azure subscription required.
- No need to install Jupyter and the supporting R or Python distributions locally—just use a browser.
- Manage your own online libraries and access them from any device.
- Share your notebooks with collaborators.

Considerations:

- You will be unable to access your notebooks when offline.
- Limited processing capabilities of the free notebook service may not be enough to train large or complex models.

## Key selection criteria

To narrow the choices, start by answering these questions:

- Do you need to connect to numerous data sources, providing a centralized place to create reports for data spread throughout your domain? If so, choose an option that allows you to connect to 100s of data sources.
- Do you want to embed dynamic visualizations in an external website or application? If so, choose an option that provides embedding capabilities.
- Do you want to design your visualizations and reports while offline? If yes, choose an option with offline capabilities.
- Do you need heavy processing power to train large or complex AI models or work with very large data sets? If yes, choose an option that can connect to a big data cluster.

## Capability matrix

The following tables summarize the key differences in capabilities.

### General capabilities

	POWER BI	JUPYTER NOTEBOOKS	ZEPELIN NOTEBOOKS	MICROSOFT AZURE NOTEBOOKS
Connect to big data cluster for advanced processing	Yes	Yes	Yes	No
Managed service	Yes	Yes <sup>1</sup>	Yes <sup>1</sup>	Yes
Connect to 100s of data sources	Yes	No	No	No
Offline capabilities	Yes <sup>2</sup>	No	No	No
Embedding capabilities	Yes	No	No	No



	POWER BI	JUPYTER NOTEBOOKS	ZEPELIN NOTEBOOKS	MICROSOFT AZURE NOTEBOOKS
Automatic data refresh	Yes	No	No	No
Access to numerous open source packages	No	Yes <sup>3</sup>	Yes <sup>3</sup>	Yes <sup>4</sup>
Data transformation/cleansing options	<a href="#">Power Query</a> , R	40 languages, including Python, R, Julia, and Scala	20+ interpreters, including Python, JDBC, and R	Python, F#, R
Pricing	Free for Power BI Desktop (authoring), see <a href="#">pricing</a> for hosting options	Free	Free	Free
Multiusers collaboration	<a href="#">Yes</a>	Yes (through sharing or with a multiusers server like <a href="#">JupyterHub</a> )	Yes	Yes (through sharing)

[1] When used as part of a managed HDInsight cluster.

[2] With the use of Power BI Desktop.

[2] You can search the [Maven repository](#) for community-contributed packages.

[3] Python packages can be installed using either pip or conda. R packages can be installed from CRAN or GitHub. Packages in F# can be installed via nuget.org using the [Paket dependency manager](#).

# Choosing a batch processing technology in Azure

2/13/2018 • 2 min to read • [Edit Online](#)

Big data solutions often use long-running batch jobs to filter, aggregate, and otherwise prepare the data for analysis. Usually these jobs involve reading source files from scalable storage (like HDFS, Azure Data Lake Store, and Azure Storage), processing them, and writing the output to new files in scalable storage.

The key requirement of such batch processing engines is the ability to scale out computations, in order to handle a large volume of data. Unlike real-time processing, however, batch processing is expected to have latencies (the time between data ingestion and computing a result) that measure in minutes to hours.

## What are your options when choosing a batch processing technology?

In Azure, all of the following data stores will meet the core requirements for batch processing:

- [Azure Data Lake Analytics](#)
- [Azure SQL Data Warehouse](#)
- [HDInsight with Spark](#)
- [HDInsight with Hive](#)
- [HDInsight with Hive LLAP](#)

## Key selection criteria

To narrow the choices, start by answering these questions:

- Do you want a managed service rather than managing your own servers?
- Do you want to author batch processing logic declaratively or imperatively?
- Will you perform batch processing in bursts? If yes, consider options that let you pause the cluster or whose pricing model is per batch job.
- Do you need to query relational data stores along with your batch processing, for example to look up reference data? If yes, consider the options that enable querying of external relational stores.

## Capability matrix

The following tables summarize the key differences in capabilities.

### General capabilities

	AZURE DATA LAKE ANALYTICS	AZURE SQL DATA WAREHOUSE	HDINSIGHT WITH SPARK	HDINSIGHT WITH HIVE	HDINSIGHT WITH HIVE LLAP
Is managed service	Yes	Yes	Yes <sup>1</sup>	Yes <sup>1</sup>	Yes <sup>1</sup>
Supports pausing compute	No	Yes	No	No	No
Relational data store	Yes	Yes	No	No	No

	AZURE DATA LAKE ANALYTICS	AZURE SQL DATA WAREHOUSE	HDINSIGHT WITH SPARK	HDINSIGHT WITH HIVE	HDINSIGHT WITH HIVE LLAP
Programmability	U-SQL	T-SQL	Python, Scala, Java, R	HiveQL	HiveQL
Programming paradigm	Mixture of declarative and imperative	Declarative	Mixture of declarative and imperative	Declarative	Declarative
Pricing model	Per batch job	By cluster hour	By cluster hour	By cluster hour	By cluster hour

[1] With manual configuration and scaling.

### Integration capabilities

	AZURE DATA LAKE ANALYTICS	SQL DATA WAREHOUSE	HDINSIGHT WITH SPARK	HDINSIGHT WITH HIVE	HDINSIGHT WITH HIVE LLAP
Access from Azure Data Lake Store	Yes	Yes	Yes	Yes	Yes
Query from Azure Storage	Yes	Yes	Yes	Yes	Yes
Query from external relational stores	Yes	No	Yes	No	No

### Scalability capabilities

	AZURE DATA LAKE ANALYTICS	SQL DATA WAREHOUSE	HDINSIGHT WITH SPARK	HDINSIGHT WITH HIVE	HDINSIGHT WITH HIVE LLAP
Scale-out granularity	Per job	Per cluster	Per cluster	Per cluster	Per cluster
Fast scale out (less than 1 minute)	Yes	Yes	No	No	No
In-memory caching of data	No	Yes	Yes	No	Yes

### Security capabilities

	AZURE DATA LAKE ANALYTICS	SQL DATA WAREHOUSE	HDINSIGHT WITH SPARK	APACHE HIVE ON HDINSIGHT	HIVE LLAP ON HDINSIGHT
Authentication	Azure Active Directory (Azure AD)	SQL / Azure AD	No	local / Azure AD <sup>1</sup>	local / Azure AD <sup>1</sup>
Authorization	Yes	Yes	No	Yes <sup>1</sup>	Yes <sup>1</sup>
Auditing	Yes	Yes	No	Yes <sup>1</sup>	Yes <sup>1</sup>

	<b>AZURE DATA LAKE ANALYTICS</b>	<b>SQL DATA WAREHOUSE</b>	<b>HDINSIGHT WITH SPARK</b>	<b>APACHE HIVE ON HDINSIGHT</b>	<b>HIVE LLAP ON HDINSIGHT</b>
Data encryption at rest	Yes	Yes <sup>2</sup>	Yes	Yes	Yes
Row-level security	No	Yes	No	Yes <sup>1</sup>	Yes <sup>1</sup>
Supports firewalls	Yes	Yes	Yes	Yes <sup>3</sup>	Yes <sup>3</sup>
Dynamic data masking	No	No	No	Yes <sup>1</sup>	Yes <sup>1</sup>

[1] Requires using a [domain-joined HDInsight cluster](#).

[2] Requires using Transparent Data Encryption (TDE) to encrypt and decrypt your data at rest.

[3] Supported when [used within an Azure Virtual Network](#).

# Choosing a Microsoft cognitive services technology

2/13/2018 • 3 min to read • [Edit Online](#)

Microsoft cognitive services are cloud-based APIs that you can use in artificial intelligence (AI) applications and data flows. They provide you with pretrained models that are ready to use in your application, requiring no data and no model training on your part. The cognitive services are developed by Microsoft's AI and Research team and leverage the latest deep learning algorithms. They are consumed over HTTP REST interfaces. In addition, SDKs are available for many common application development frameworks.

The cognitive services include:

- Text analysis
- Computer vision
- Video analytics
- Speech recognition and generation
- Natural language understanding
- Intelligent search

Key benefits:

- Minimal development effort for state-of-the-art AI services.
- Easy integration into apps via HTTP REST interfaces.
- Built-in support for consuming cognitive services in Azure Data Lake Analytics.

Considerations:

- Only available over the web. Internet connectivity is generally required. An exception is the Custom Vision Service, whose trained model you can export for prediction on devices and at the IoT edge.
- Although considerable customization is supported, the available services may not suit all predictive analytics requirements.

## What are your options when choosing amongst the cognitive services?

In Azure, there are dozens of Cognitive Services available. The current listing of these is available in a directory categorized by the functional area they support:

- [Vision](#)
- [Speech](#)
- [Knowledge](#)
- [Search](#)
- [Language](#)

## Key Selection Criteria

To narrow the choices, start by answering these questions:

- What type of data are you dealing with? Narrow your options based on the type of input data you are working with. For example, if your input is text, select from the services that have an input type of text.
- Do you have the data to train a model? If yes, consider the custom services that enable you to train their underlying models with data that you provide, for improved accuracy and performance.

# Capability matrix

The following tables summarize the key differences in capabilities.

## Uses prebuilt models

	INPUT TYPE	KEY BENEFIT
Text Analytics API	Text	Evaluate sentiment and topics to understand what users want.
Entity Linking API	Text	Power your app's data links with named entity recognition and disambiguation.
Language Understanding Intelligent Service (LUIS)	Text	Teach your apps to understand commands from your users.
QnA Maker Service	Text	Distill FAQ formatted information into conversational, easy-to-navigate answers.
Linguistic Analysis API	Text	Simplify complex language concepts and parse text.
Knowledge Exploration Service	Text	Enable interactive search experiences over structured data via natural language inputs.
Web Language Model API	Text	Use predictive language models trained on web-scale data.
Academic Knowledge API	Text	Tap into the wealth of academic content in the Microsoft Academic Graph populated by Bing.
Bing Autosuggest API	Text	Give your app intelligent autosuggest options for searches.
Bing Spell Check API	Text	Detect and correct spelling mistakes in your app.
Translator Text API	Text	Machine translation.
Recommendations API	Text	Predict and recommend items your customers want.
Bing Entity Search API	Text (web search query)	Identify and augment entity information from the web.
Bing Image Search API	Text (web search query)	Search for images.
Bing News Search API	Text (web search query)	Search for news.
Bing Video Search API	Text (web search query)	Search for videos.

	INPUT TYPE	KEY BENEFIT
Bing Web Search API	Text (web search query)	Get enhanced search details from billions of web documents.
Bing Speech API	Text or Speech	Convert speech to text and back again.
Speaker Recognition API	Speech	Use speech to identify and authenticate individual speakers.
Translator Speech API	Speech	Perform real-time speech translation.
Computer Vision API	Images (or frames from video)	Distill actionable information from images, automatically create description of photos, derive tags, recognize celebrities, extract text, and create accurate thumbnails.
Content Moderator	Text, Images or Video	Automated image, text, and video moderation.
Emotion API	Images (photos with human subjects)	Identify the range emotions of human subjects.
Face API	Images (photos with human subjects)	Detect, identify, analyze, organize, and tag faces in photos.
Video Indexer	Video	Video insights such as sentiment, transcript speech, translate speech, recognize faces and emotions, and extract keywords.

#### Trained with custom data you provide

	INPUT TYPE	KEY BENEFIT
Custom Vision Service	Images (or frames from video)	Customize your own computer vision models.
Custom Speech Service	Speech	Overcome speech recognition barriers like speaking style, background noise, and vocabulary.
Custom Decision Service	Web content (for example, RSS feed)	Use machine learning to automatically select the appropriate content for your home page
Bing Custom Search API	Text (web search query)	Commercial-grade search tool.

# Choosing a big data storage technology in Azure

2/15/2018 • 7 min to read • [Edit Online](#)

This topic compares options for data storage for big data solutions — specifically, data storage for bulk data ingestion and batch processing, as opposed to [analytical data stores](#) or [real-time streaming ingestion](#).

## What are your options when choosing data storage in Azure?

There are several options for ingesting data into Azure, depending on your needs:

### File storage

- [Azure Storage blobs](#)
- [Azure Data Lake Store](#)

### NoSQL databases

- [Azure Cosmos DB](#)
- [HBase on HDInsight](#)

## Azure Storage blobs

Azure Storage is a managed storage service that is highly available, secure, durable, scalable, and redundant. Microsoft takes care of maintenance and handles critical problems for you. Azure Storage is the most ubiquitous storage solution Azure provides, due to the number of services and tools that can be used with it.

There are various Azure Storage services you can use to store data. The most flexible option for storing blobs from a number of data sources is [Blob storage](#). Blobs are basically files. They store pictures, documents, HTML files, virtual hard disks (VHDs), big data such as logs, database backups — pretty much anything. Blobs are stored in containers, which are similar to folders. A container provides a grouping of a set of blobs. A storage account can contain an unlimited number of containers, and a container can store an unlimited number of blobs.

Azure Storage is a good choice for big data and analytics solutions, because of its flexibility, high availability, and low cost. It provides hot, cool, and archive storage tiers for different use cases. For more information, see [Azure Blob Storage: Hot, cool, and archive storage tiers](#).

Azure Blob storage can be accessed from Hadoop (available through HDInsight). HDInsight can use a blob container in Azure Storage as the default file system for the cluster. Through a Hadoop distributed file system (HDFS) interface provided by a WASB driver, the full set of components in HDInsight can operate directly on structured or unstructured data stored as blobs. Azure Blob storage can also be accessed via Azure SQL Data Warehouse using its PolyBase feature.

Other features that make Azure Storage a good choice are:

- [Multiple concurrency strategies](#).
- [Disaster recovery and high availability options](#).
- [Encryption at rest](#).
- [Role-Based Access Control \(RBAC\)](#) to control access using Azure Active Directory users and groups.

## Azure Data Lake Store

[Azure Data Lake Store](#) is an enterprise-wide hyper-scale repository for big data analytic workloads. Data Lake



enables you to capture data of any size, type, and ingestion speed in one single [secure](#) location for operational and exploratory analytics.

Data Lake Store does not impose any limits on account sizes, file sizes, or the amount of data that can be stored in a data lake. Data is stored durably by making multiple copies and there is no limit on the duration of time that the data can be stored in the Data Lake. In addition to making multiple copies of files to guard against any unexpected failures, Data lake spreads parts of a file over a number of individual storage servers. This improves the read throughput when reading the file in parallel for performing data analytics.

Data Lake Store can be accessed from Hadoop (available through HDInsight) using the WebHDFS-compatible REST APIs. You may consider using this as an alternative to Azure Storage when your individual or combined file sizes exceed that which is supported by Azure Storage. However, there are [performance tuning guidelines](#) you should follow when using Data Lake Store as your primary storage for an HDInsight cluster, with specific guidelines for [Spark](#), [Hive](#), [MapReduce](#), and [Storm](#). Also, be sure to check Data Lake Store's [regional availability](#), because it is not available in as many regions as Azure Storage, and it needs to be located in the same region as your HDInsight cluster.

Coupled with Azure Data Lake Analytics, Data Lake Store is specifically designed to enable analytics on the stored data and is tuned for performance for data analytics scenarios. Data Lake Store can also be accessed via Azure SQL Data Warehouse using its PolyBase feature.

## Azure Cosmos DB

[Azure Cosmos DB](#) is Microsoft's globally distributed multi-model database. Cosmos DB guarantees single-digit-millisecond latencies at the 99th percentile anywhere in the world, offers multiple well-defined consistency models to fine-tune performance, and guarantees high availability with multi-homing capabilities.

Azure Cosmos DB is schema-agnostic. It automatically indexes all the data without requiring you to deal with schema and index management. It's also multi-model, natively supporting document, key-value, graph, and column-family data models.

Azure Cosmos DB features:

- [Geo-replication](#)
- [Elastic scaling of throughput and storage](#) worldwide
- [Five well-defined consistency levels](#)

## HBase on HDInsight

[Apache HBase](#) is an open-source, NoSQL database that is built on Hadoop and modeled after Google BigTable. HBase provides random access and strong consistency for large amounts of unstructured and semi-structured data in a schemaless database organized by column families.

Data is stored in the rows of a table, and data within a row is grouped by column family. HBase is schemaless in the sense that neither the columns nor the type of data stored in them need to be defined before using them. The open-source code scales linearly to handle petabytes of data on thousands of nodes. It can rely on data redundancy, batch processing, and other features that are provided by distributed applications in the Hadoop ecosystem.

The [HDInsight implementation](#) leverages the scale-out architecture of HBase to provide automatic sharding of tables, strong consistency for reads and writes, and automatic failover. Performance is enhanced by in-memory caching for reads and high-throughput streaming for writes. In most cases, you'll want to [create the HBase cluster inside a virtual network](#) so other HDInsight clusters and applications can directly access the tables.

## Key selection criteria

To narrow the choices, start by answering these questions:

- Do you need managed, high speed, cloud-based storage for any type of text or binary data? If yes, then select one of the file storage options.
- Do you need file storage that is optimized for parallel analytics workloads and high throughput/IOPS? If yes, then choose an option that is tuned to analytics workload performance.
- Do you need to store unstructured or semi-structured data in a schemaless database? If so, select one of the non-relational options. Compare options for indexing and database models. Depending on the type of data you need to store, the primary database models may be the largest factor.
- Can you use the service in your region? Check the regional availability for each Azure service. See [Products available by region](#).

## Capability matrix

The following tables summarize the key differences in capabilities.

### File storage capabilities

	AZURE DATA LAKE STORE	AZURE BLOB STORAGE CONTAINERS
Purpose	Optimized storage for big data analytics workloads	General purpose object store for a wide variety of storage scenarios
Use cases	Batch, streaming analytics, and machine learning data such as log files, IoT data, click streams, large datasets	Any type of text or binary data, such as application back end, backup data, media storage for streaming, and general purpose data
Structure	Hierarchical file system	Object store with flat namespace
Authentication	Based on <a href="#">Azure Active Directory Identities</a>	Based on shared secrets <a href="#">Account Access Keys</a> and <a href="#">Shared Access Signature Keys</a> , and <a href="#">Role-Based Access Control (RBAC)</a>
Authentication protocol	OAuth 2.0. Calls must contain a valid JWT (JSON web token) issued by Azure Active Directory	Hash-based message authentication code (HMAC). Calls must contain a Base64-encoded SHA-256 hash over a part of the HTTP request.
Authorization	POSIX access control lists (ACLs). ACLs based on Azure Active Directory identities can be set file and folder level.	For account-level authorization use <a href="#">Account Access Keys</a> . For account, container, or blob authorization use <a href="#">Shared Access Signature Keys</a> .
Auditing	Available.	Available
Encryption at rest	Transparent, server side	Transparent, server side; Client-side encryption
Developer SDKs	.NET, Java, Python, Node.js	.Net, Java, Python, Node.js, C++, Ruby
Analytics workload performance	Optimized performance for parallel analytics workloads, High Throughput and IOPS	Not optimized for analytics workloads

	AZURE DATA LAKE STORE	AZURE BLOB STORAGE CONTAINERS
Size limits	No limits on account sizes, file sizes or number of files	Specific limits documented <a href="#">here</a>
Geo-redundancy	Locally-redundant (multiple copies of data in one Azure region)	Locally redundant (LRS), globally redundant (GRS), read-access globally redundant (RA-GRS). See <a href="#">here</a> for more information

## NoSQL database capabilities

	AZURE COSMOS DB	HBASE ON HDINSIGHT
Primary database model	Document store, graph, key-value store, wide column store	Wide column store
Secondary indexes	Yes	No
SQL language support	Yes	Yes (using the <a href="#">Phoenix</a> JDBC driver)
Consistency	Strong, bounded-staleness, session, consistent prefix, eventual	Strong
Native Azure Functions integration	<a href="#">Yes</a>	No
Automatic global distribution	<a href="#">Yes</a>	No <a href="#">HBase cluster replication can be configured</a> across regions with eventual consistency
Pricing model	Elastically scalable request units (RUs) charged per-second as needed, elastically scalable storage	Per-minute pricing for HDInsight cluster (horizontal scaling of nodes), storage

# Choosing a machine learning technology in Azure

2/13/2018 • 7 min to read • [Edit Online](#)

Data science and machine learning is a workload that is usually undertaken by data scientists. It requires specialist tools, many of which are designed specifically for the type of interactive data exploration and modeling tasks that a data scientist must perform.

Machine learning solutions are built iteratively, and have two distinct phases:

- Data preparation and modeling.
- Deployment and consumption of predictive services.

## Tools and services for data preparation and modeling

Data scientists typically prefer to work with data using custom code written in Python or R. This code is generally run interactively, with the data scientists using it to query and explore the data, generating visualizations and statistics to help determine the relationships with it. There are many interactive environments for R and Python that data scientists can use. A particular favorite is **Jupyter Notebooks** that provides a browser-based shell that enables data scientists to create *notebook* files that contain R or Python code and markdown text. This is an effective way to collaborate by sharing and documenting code and results in a single document.

Other commonly used tools include:

- **Spyder**: The interactive development environment (IDE) for Python provided with the Anaconda Python distribution.
- **R Studio**: An IDE for the R programming language.
- **Visual Studio Code**: A lightweight, cross-platform coding environment that supports Python as well as commonly used frameworks for machine learning and AI development.

In addition to these tools, data scientists can leverage Azure services to simplify code and model management.

### Azure Notebooks

Azure Notebooks is an online Jupyter Notebooks service that enables data scientists to create, run, and share Jupyter Notebooks in cloud-based libraries.

Key benefits:

- Free service—no Azure subscription required.
- No need to install Jupyter and the supporting R or Python distributions locally—just use a browser.
- Manage your own online libraries and access them from any device.
- Share your notebooks with collaborators.

Considerations:

- You will be unable to access your notebooks when offline.
- Limited processing capabilities of the free notebook service may not be enough to train large or complex models.

### Data science virtual machine

The data science virtual machine is an Azure virtual machine image that includes the tools and frameworks commonly used by data scientists, including R, Python, Jupyter Notebooks, Visual Studio Code, and libraries for machine learning modeling such as the Microsoft Cognitive Toolkit. These tools can be complex and time

consuming to install, and contain many interdependencies that often lead to version management issues. Having a preinstalled image can reduce the time data scientists spend troubleshooting environment issues, allowing them to focus on the data exploration and modeling tasks they need to perform.

Key benefits:

- Reduced time to install, manage, and troubleshoot data science tools and frameworks.
- The latest versions of all commonly used tools and frameworks are included.
- Virtual machine options include highly scalable images with GPU capabilities for intensive data modeling.

Considerations:

- The virtual machine cannot be accessed when offline.
- Running a virtual machine incurs Azure charges, so you must be careful to have it running only when required.

### **Azure Machine Learning**

Azure Machine Learning is a cloud-based service for managing machine learning experiments and models. It includes an experimentation service that tracks data preparation and modeling training scripts, maintaining a history of all executions so you can compare model performance across iterations. A cross-platform client tool named Azure Machine Learning Workbench provides a central interface for script management and history, while still enabling data scientists to create scripts in their tool of choice, such as Jupyter Notebooks or Visual Studio Code.

From Azure Machine Learning Workbench, you can use the interactive data preparation tools to simplify common data transformation tasks, and you can configure the script execution environment to run model training scripts locally, in a scalable Docker container, or in Spark.

When you are ready to deploy your model, use the Workbench environment to package the model and deploy it as a web service to a Docker container, Spark on Azure HDInsight, Microsoft Machine Learning Server, or SQL Server. The Azure Machine Learning Model Management service then enables you to track and manage model deployments in the cloud, on edge devices, or across the enterprise.

Key benefits:

- Central management of scripts and run history, making it easy to compare model versions.
- Interactive data transformation through a visual editor.
- Easy deployment and management of models to the cloud or edge devices.

Considerations:

- Requires some familiarity with the model management model and Workbench tool environment.

### **Azure Batch AI**

Azure Batch AI enables you to run your machine learning experiments in parallel, and perform model training at scale across a cluster of virtual machines with GPUs. Batch AI training enables you to scale out deep learning jobs across clustered GPUs, using frameworks such as Cognitive Toolkit, Caffe, Chainer, and TensorFlow.

Azure Machine Learning Model Management can be used to take models from Batch AI training to deploy, manage, and monitor them.

### **Azure Machine Learning Studio**

Azure Machine Learning Studio is a cloud-based, visual development environment for creating data experiments, training machine learning models, and publishing them as web services in Azure. Its visual drag-and-drop interface lets data scientists and power users create machine learning solutions quickly, while supporting custom R and Python logic, a wide range of established statistical algorithms and techniques for machine learning modeling tasks, and built-in support for Jupyter Notebooks.

Key benefits:

- Interactive visual interface enables machine learning modeling with minimal code.
- Built-in Jupyter Notebooks for data exploration.
- Direct deployment of trained models as Azure web services.

Considerations:

- Limited scalability. The maximum size of a training dataset is 10 GB.
- Online only. No offline development environment.

## Tools and services for deploying machine learning models

After a data scientist has created a machine learning model, you will typically need to deploy it and consume it from applications or in other data flows. There are a number of potential deployment targets for machine learning models.

### **Spark on Azure HDInsight**

Apache Spark includes Spark MLlib, a framework and library for machine learning models. The Microsoft Machine Learning library for Spark (MMLSpark) also provides deep learning algorithm support for predictive models in Spark.

Key benefits:

- Spark is a distributed platform that offers high scalability for high-volume machine learning processes.
- You can deploy models directly to Spark in HDInsight from Azure Machine Learning Workbench, and manage them using the Azure Machine Learning Model Management service.

Considerations:

- Spark runs in an HDInsight cluster that incurs charges the whole time it is running. If the machine learning service will only be used occasionally, this may result in unnecessary costs.

### **Web service in a container**

You can deploy a machine learning model as a Python web service in a Docker container. You can deploy the model to Azure or to an edge device, where it can be used locally with the data on which it operates.

Key Benefits:

- Containers are a lightweight and generally cost effective way to package and deploy services.
- The ability to deploy to an edge device enables you to move your predictive logic closer to the data.
- You can deploy to a container directly from Azure Machine Learning Workbench.

Considerations:

- This deployment model is based on Docker containers, so you should be familiar with this technology before deploying a web service this way.

### **Microsoft Machine Learning Server**

Machine Learning Server (formerly Microsoft R Server) is a scalable platform for R and Python code, specifically designed for machine learning scenarios.

Key benefits:

- High scalability.
- Direct deployment from Azure Machine Learning Workbench.

Considerations:

- You need to deploy and manage Machine Learning Server in your enterprise.

### **Microsoft SQL Server**

Microsoft SQL Server supports R and Python natively, enabling you to encapsulate machine learning models built in these languages as Transact-SQL functions in a database.

Key benefits:

- Encapsulate predictive logic in a database function, making it easy to include in data-tier logic.

Considerations:

- Assumes a SQL Server database as the data tier for your application.

### **Azure Machine Learning web service**

When you create a machine learning model using Azure Machine Learning Studio, you can deploy it as a web service. This can then be consumed through a REST interface from any client applications capable of communicating by HTTP.

Key benefits:

- Ease of development and deployment.
- Web service management portal with basic monitoring metrics.
- Built-in support for calling Azure Machine Learning web services from Azure Data Lake Analytics, Azure Data Factory, and Azure Stream Analytics.

Considerations:

- Only available for models built using Azure Machine Learning Studio.
- Web-based access only, trained models cannot run on-premises or offline.

# Choosing a natural language processing technology in Azure

2/13/2018 • 1 min to read • [Edit Online](#)

Free-form text processing is performed against documents containing paragraphs of text, typically for the purpose of supporting search, but is also used to perform other natural language processing (NLP) tasks such as sentiment analysis, topic detection, language detection, key phrase extraction, and document categorization. This article focuses on the technology choices that act in support of the NLP tasks.

## What are your options when choosing an NLP service?

In Azure, the following services provide natural language processing (NLP) capabilities:

- [Azure HDInsight with Spark and Spark MLlib](#)
- [Microsoft Cognitive Services](#)

## Key selection criteria

To narrow the choices, start by answering these questions:

- Do you want to use prebuilt models? If yes, consider using the APIs offered by Microsoft Cognitive Services.
- Do you need to train custom models against a large corpus of text data? If yes, consider using Azure HDInsight with Spark MLlib and Spark NLP.
- Do you need low-level NLP capabilities like tokenization, stemming, lemmatization, and term frequency/inverse document frequency (TF/IDF)? If yes, consider using Azure HDInsight with Spark MLlib and Spark NLP.
- Do you need simple, high-level NLP capabilities like entity and intent identification, topic detection, spell check, or sentiment analysis? If yes, consider using the APIs offered by Microsoft Cognitive Services.

## Capability matrix

The following tables summarize the key differences in capabilities.

### General capabilities

	AZURE HDINSIGHT	MICROSOFT COGNITIVE SERVICES
Provides pretrained models as a service	No	Yes
REST API	Yes	Yes
Programmability	Python, Scala, Java	C#, Java, Node.js, Python, PHP, Ruby
Support processing of big data sets and large documents	Yes	No

### Low-level natural language processing capabilities



	<b>AZURE HDINSIGHT</b>	<b>MICROSOFT COGNITIVE SERVICES</b>
Tokenizer	Yes (Spark NLP)	Yes (Linguistic Analysis API)
Stemmer	Yes (Spark NLP)	No
Lemmatizer	Yes (Spark NLP)	No
Part of speech tagging	Yes (Spark NLP)	Yes (Linguistic Analysis API)
Term frequency/inverse-document frequency (TF/IDF)	Yes (Spark MLlib)	No
String similarity—edit distance calculation	Yes (Spark MLlib)	No
N-gram calculation	Yes (Spark MLlib)	No
Stop word removal	Yes (Spark MLlib)	No

### High-level natural language processing capabilities

	<b>AZURE HDINSIGHT</b>	<b>MICROSOFT COGNITIVE SERVICES</b>
Entity/intent identification & extraction	No	Yes (Language Understanding Intelligent Service (LUIS) API)
Topic detection	Yes (Spark NLP)	Yes (Text Analytics API)
Spell checking	Yes (Spark NLP)	Yes (Bing Spell Check API)
Sentiment analysis	Yes (Spark NLP)	Yes (Text Analytics API)
Language detection	No	Yes (Text Analytics API)
Supports multiple languages besides English	No	Yes (varies by API)

## See also

[Natural language processing](#)

# Choosing a data pipeline orchestration technology in Azure

2/13/2018 • 2 min to read • [Edit Online](#)

Most big data solutions consist of repeated data processing operations, encapsulated in workflows. A pipeline orchestrator is a tool that helps to automate these workflows. An orchestrator can schedule jobs, execute workflows, and coordinate dependencies among tasks.

## What are your options for data pipeline orchestration?

In Azure, the following services and tools will meet the core requirements for pipeline orchestration, control flow, and data movement:

- [Azure Data Factory](#)
- [Oozie on HDInsight](#)
- [SQL Server Integration Services \(SSIS\)](#)

These services and tools can be used independently from one another, or used together to create a hybrid solution. For example, the Integration Runtime (IR) in Azure Data Factory V2 can natively execute SSIS packages in a managed Azure compute environment. While there is some overlap in functionality between these services, there are a few key differences.

## Key Selection Criteria

To narrow the choices, start by answering these questions:

- Do you need big data capabilities for moving and transforming your data? Usually this means multi-gigabytes to terabytes of data. If yes, then narrow your options to those that best suited for big data.
- Do you require a managed service that can operate at scale? If yes, select one of the cloud-based services that aren't limited by your local processing power.
- Are some of your data sources located on-premises? If yes, look for options that can work with both cloud and on-premises data sources or destinations.
- Is your source data stored in Blob storage on an HDFS filesystem? If so, choose an option that supports Hive queries.

## Capability matrix

The following tables summarize the key differences in capabilities.

### General capabilities

	AZURE DATA FACTORY	SQL SERVER INTEGRATION SERVICES (SSIS)	OOZIE ON HDINSIGHT
Managed	Yes	No	Yes
Cloud-based	Yes	No (local)	Yes

	<b>AZURE DATA FACTORY</b>	<b>SQL SERVER INTEGRATION SERVICES (SSIS)</b>	<b>OOZIE ON HDINSIGHT</b>
Prerequisite	Azure Subscription	SQL Server	Azure Subscription, HDInsight cluster
Management tools	Azure Portal, PowerShell, CLI, .NET SDK	SSMS, PowerShell	Bash shell, Oozie REST API, Oozie web UI
Pricing	Pay per usage	Licensing / pay for features	No additional charge on top of running the HDInsight cluster

### Pipeline capabilities

	<b>AZURE DATA FACTORY</b>	<b>SQL SERVER INTEGRATION SERVICES (SSIS)</b>	<b>OOZIE ON HDINSIGHT</b>
Copy data	Yes	Yes	Yes
Custom transformations	Yes	Yes	Yes (MapReduce, Pig, and Hive jobs)
Azure Machine Learning scoring	Yes	Yes (with scripting)	No
HDInsight On-Demand	Yes	No	No
Azure Batch	Yes	No	No
Pig, Hive, MapReduce	Yes	No	Yes
Spark	Yes	No	No
Execute SSIS Package	Yes	Yes	No
Control flow	Yes	Yes	Yes
Access on-premises data	Yes	Yes	No

### Scalability capabilities

	<b>AZURE DATA FACTORY</b>	<b>SQL SERVER INTEGRATION SERVICES (SSIS)</b>	<b>OOZIE ON HDINSIGHT</b>
Scale up	Yes	No	No
Scale out	Yes	No	Yes (by adding worker nodes to cluster)
Optimized for big data	Yes	No	Yes

# Choosing a real-time message ingestion technology in Azure

2/22/2018 • 2 min to read • [Edit Online](#)

Real time processing deals with streams of data that are captured in real-time and processed with minimal latency. Many real-time processing solutions need a message ingestion store to act as a buffer for messages, and to support scale-out processing, reliable delivery, and other message queuing semantics.

## What are your options for real-time message ingestion?

- [Azure Event Hubs](#)
- [Azure IoT Hub](#)
- [Kafka on HDInsight](#)

## Azure Event Hubs

[Azure Event Hubs](#) is a highly scalable data streaming platform and event ingestion service, capable of receiving and processing millions of events per second. Event Hubs can process and store events, data, or telemetry produced by distributed software and devices. Data sent to an event hub can be transformed and stored using any real-time analytics provider or batching/storage adapters. Event Hubs provides publish-subscribe capabilities with low latency at massive scale, which makes it appropriate for big data scenarios.

## Azure IoT Hub

[Azure IoT Hub](#) is a managed service that enables reliable and secure bidirectional communications between millions of IoT devices and a cloud-based back end.

Feature of IoT Hub include:

- Multiple options for device-to-cloud and cloud-to-device communication. These options include one-way messaging, file transfer, and request-reply methods.
- Message routing to other Azure services.
- Queryable store for device metadata and synchronized state information.
- Secure communications and access control using per-device security keys or X.509 certificates.
- Monitoring of device connectivity and device identity management events.

In terms of message ingestion, IoT Hub is similar to Event Hubs. However, it was specifically designed for managing IoT device connectivity, not just message ingestion. For more information, see [Comparison of Azure IoT Hub and Azure Event Hubs](#).

## Kafka on HDInsight

[Apache Kafka](#) is an open-source distributed streaming platform that can be used to build real-time data pipelines and streaming applications. Kafka also provides message broker functionality similar to a message queue, where you can publish and subscribe to named data streams. It is horizontally scalable, fault-tolerant, and extremely fast. [Kafka on HDInsight](#) provides a Kafka as a managed, highly scalable, and highly available service in Azure.

Some common use cases for Kafka are:

- **Messaging.** Because it supports the publish-subscribe message pattern, Kafka is often used as a message

broker.

- **Activity tracking.** Because Kafka provides in-order logging of records, it can be used to track and re-create activities, such as user actions on a web site.
- **Aggregation.** Using stream processing, you can aggregate information from different streams to combine and centralize the information into operational data.
- **Transformation.** Using stream processing, you can combine and enrich data from multiple input topics into one or more output topics.

## Key selection criteria

To narrow the choices, start by answering these questions:

- Do you need two-way communication between your IoT devices and Azure? If so, choose IoT Hub.
- Do you need to manage access for individual devices and be able to revoke access to a specific device? If yes, choose IoT Hub.

## Capability matrix

The following tables summarize the key differences in capabilities.

	IOT HUB	EVENT HUBS	KAFKA ON HDINSIGHT
Cloud-to-device communications	Yes	No	No
Device-initiated file upload	Yes	No	No
Device state information	<a href="#">Device twins</a>	No	No
Protocol support	MQTT, AMQP, HTTPS <sup>1</sup>	AMQP, HTTPS	<a href="#">Kafka Protocol</a>
Security	Per-device identity; revocable access control.	Shared access policies; limited revocation through publisher policies.	Authentication using SASL; pluggable authorization; integration with external authentication services supported.

[1] You can also use [Azure IoT protocol gateway](#) as a custom gateway to enable protocol adaptation for IoT Hub.

For more information, see [Comparison of Azure IoT Hub and Azure Event Hubs](#).

# Choosing a search data store in Azure

2/13/2018 • 2 min to read • [Edit Online](#)

This article compares technology choices for search data stores in Azure. A search data store is used to create and store specialized indexes for performing searches on free-form text. The text that is indexed may reside in a separate data store, such as blob storage. An application submits a query to the search data store, and the result is a list of matching documents. For more information about this scenario, see [Processing free-form text for search](#).

## What are your options when choosing a search data store?

In Azure, all of the following data stores will meet the core requirements for search against free-form text data by providing a search index:

- [Azure Search](#)
- [Elasticsearch](#)
- [HDInsight with Solr](#)
- [Azure SQL Database with full text search](#)

## Key selection criteria

For search scenarios, begin choosing the appropriate search data store for your needs by answering these questions:

- Do you want a managed service rather than managing your own servers?
- Can you specify your index schema at design time? If not, choose an option that supports updateable schemas.
- Do you need an index only for full-text search, or do you also need rapid aggregation of numeric data and other analytics? If you need functionality beyond full-text search, consider options that support additional analytics.
- Do you need a search index for log analytics, with support for log collection, aggregation, and visualizations on indexed data? If so, consider Elasticsearch, which is part of a log analytics stack.
- Do you need to index data in common document formats such as PDF, Word, PowerPoint, and Excel? If yes, choose an option that provides document indexers.
- Does your database have specific security needs? If yes, consider the security features listed below.

## Capability matrix

The following tables summarize the key differences in capabilities.

### General capabilities

	AZURE SEARCH	ELASTICSEARCH	HDINSIGHT WITH SOLR	SQL DATABASE
Is managed service	Yes	No	Yes	Yes
REST API	Yes	Yes	Yes	No

	AZURE SEARCH	ELASTICSEARCH	HDINSIGHT WITH SOLR	SQL DATABASE
Programmability	.NET	Java	Java	T-SQL
Document indexers for common file types (PDF, DOCX, TXT, and so on)	Yes	No	Yes	No

### Manageability capabilities

	AZURE SEARCH	ELASTICSEARCH	HDINSIGHT WITH SOLR	SQL DATABASE
Updateable schema	No	Yes	Yes	Yes
Supports scale out	Yes	Yes	Yes	No

### Analytic workload capabilities

	AZURE SEARCH	ELASTICSEARCH	HDINSIGHT WITH SOLR	SQL DATABASH
Supports analytics beyond full text search	No	Yes	Yes	Yes
Part of a log analytics stack	No	Yes (ELK)	No	No
Supports semantic search	Yes (find similar documents only)	Yes	Yes	Yes

### Security capabilities

	AZURE SEARCH	ELASTICSEARCH	HDINSIGHT WITH SOLR	SQL DATABASH
Row-level security	Partial (requires application query to filter by group id)	Partial (requires application query to filter by group id)	Yes	Yes
Transparent data encryption	No	No	No	Yes
Restrict access to specific IP addresses	No	Yes	Yes	Yes
Restrict access to allow virtual network access only	No	Yes	Yes	Yes
Active Directory authentication (integrated authentication)	No	No	No	Yes

See also





# Choosing a stream processing technology in Azure

3/4/2018 • 2 min to read • [Edit Online](#)

This article compares technology choices for real-time stream processing in Azure.

Real-time stream processing consumes messages from either queue or file-based storage, process the messages, and forward the result to another message queue, file store, or database. Processing may include querying, filtering, and aggregating messages. Stream processing engines must be able to consume an endless streams of data and produce results with minimal latency. For more information, see [Real time processing](#).

## What are your options when choosing a technology for real-time processing?

In Azure, all of the following data stores will meet the core requirements supporting real-time processing:

- [Azure Stream Analytics](#)
- [HDInsight with Spark Streaming](#)
- [Apache Spark in Azure Databricks](#)
- [HDInsight with Storm](#)
- [Azure Functions](#)
- [Azure App Service WebJobs](#)

## Key Selection Criteria

For real-time processing scenarios, begin choosing the appropriate service for your needs by answering these questions:

- Do you prefer a declarative or imperative approach to authoring stream processing logic?
- Do you need built-in support for temporal processing or windowing?
- Does your data arrive in formats besides Avro, JSON, or CSV? If yes, consider options support any format using custom code.
- Do you need to scale your processing beyond 1 GB/s? If yes, consider the options that scale with the cluster size.

## Capability matrix

The following tables summarize the key differences in capabilities.

### General capabilities

	AZURE STREAM ANALYTICS	HDINSIGHT WITH SPARK STREAMING	APACHE SPARK IN AZURE DATABRICKS	HDINSIGHT WITH STORM	AZURE FUNCTIONS	AZURE APP SERVICE WEBJOBS
Programmability	Stream analytics query language, JavaScript	Scala, Python, Java	Scala, Python, Java, R	Java, C#	C#, F#, Node.js	C#, Node.js, PHP, Java, Python

	AZURE STREAM ANALYTICS	HDINSIGHT WITH SPARK STREAMING	APACHE SPARK IN AZURE DATABRICKS	HDINSIGHT WITH STORM	AZURE FUNCTIONS	AZURE APP SERVICE WEBJOBS
Programming paradigm	Declarative	Mixture of declarative and imperative	Mixture of declarative and imperative	Imperative	Imperative	Imperative
Pricing model	<a href="#">Streaming units</a>	Per cluster hour	<a href="#">Databricks units</a>	Per cluster hour	Per function execution and resource consumption	Per app service plan hour

### Integration capabilities

	AZURE STREAM ANALYTICS	HDINSIGHT WITH SPARK STREAMING	APACHE SPARK IN AZURE DATABRICKS	HDINSIGHT WITH STORM	AZURE FUNCTIONS	AZURE APP SERVICE WEBJOBS
Inputs	<a href="#">Stream Analytics inputs</a>	Event Hubs, IoT Hub, Kafka, HDFS, Storage Blobs, Azure Data Lake Store	Event Hubs, IoT Hub, Kafka, HDFS, Storage Blobs, Azure Data Lake Store	Event Hubs, IoT Hub, Storage Blobs, Azure Data Lake Store	<a href="#">Supported bindings</a>	Service Bus, Storage Queues, Storage Blobs, Event Hubs, WebHooks, Cosmos DB, Files
Sinks	<a href="#">Stream Analytics outputs</a>	HDFS, Kafka, Storage Blobs, Azure Data Lake Store, Cosmos DB	HDFS, Kafka, Storage Blobs, Azure Data Lake Store, Cosmos DB	Event Hubs, Service Bus, Kafka	<a href="#">Supported bindings</a>	Service Bus, Storage Queues, Storage Blobs, Event Hubs, WebHooks, Cosmos DB, Files

### Processing capabilities

	AZURE STREAM ANALYTICS	HDINSIGHT WITH SPARK STREAMING	APACHE SPARK IN AZURE DATABRICKS	HDINSIGHT WITH STORM	AZURE FUNCTIONS	AZURE APP SERVICE WEBJOBS
Built-in temporal/windowing support	Yes	Yes	Yes	Yes	No	No
Input data formats	Avro, JSON or CSV, UTF-8 encoded	Any format using custom code	Any format using custom code	Any format using custom code	Any format using custom code	Any format using custom code
Scalability	<a href="#">Query partitions</a>	Bounded by cluster size	Bounded by Databricks cluster scale configuration	Bounded by cluster size	Up to 200 function app instances processing in parallel	Bounded by app service plan capacity

	<b>AZURE STREAM ANALYTICS</b>	<b>HDINSIGHT WITH SPARK STREAMING</b>	<b>APACHE SPARK IN AZURE DATABRICKS</b>	<b>HDINSIGHT WITH STORM</b>	<b>AZURE FUNCTIONS</b>	<b>AZURE APP SERVICE WEBJOBS</b>
Late arrival and out of order event handling support	Yes	Yes	Yes	Yes	No	No

See also:

- [Choosing a real-time message ingestion technology](#)
- [Comparing Apache Storm and Azure Stream Analytics](#)
- [Real time processing](#)

# Transferring data to and from Azure

2/13/2018 • 7 min to read • [Edit Online](#)

There are several options for transferring data to and from Azure, depending on your needs.

## Physical transfer

Using physical hardware to transfer data to Azure is a good option when:

- Your network is slow or unreliable.
- Getting additional network bandwidth is cost-prohibitive.
- Security or organizational policies do not allow outbound connections when dealing with sensitive data.

If your primary concern is how long it will take to transfer your data, you may want to run a test to verify whether network transfer is actually slower than physical transport.

There are two main options for physically transporting data to Azure:

- **Azure Import/Export.** The [Azure Import/Export service](#) lets you securely transfer large amounts of data to Azure Blob Storage or Azure Files by shipping internal SATA HDDs or SDDs to an Azure datacenter. You can also use this service to transfer data from Azure Storage to hard disk drives and have these shipped to you for loading on-premises.
- **Azure Data Box.** [Azure Data Box](#) is a Microsoft-provided appliance that works much like the Azure Import/Export service. Microsoft ships you a proprietary, secure, and tamper-resistant transfer appliance and handles the end-to-end logistics, which you can track through the portal. One benefit of the Azure Data Box service is ease of use. You don't need to purchase several hard drives, prepare them, and transfer files to each one. Azure Data Box is supported by a number of industry-leading Azure partners to make it easier to seamlessly leverage offline transport to the cloud from their products.

## Command line tools and APIs

Consider these options when you want scripted and programmatic data transfer.

- **Azure CLI.** The [Azure CLI](#) is a cross-platform tool that allows you to manage Azure services and upload data to Azure Storage.
- **AzCopy.** Use AzCopy from a [Windows](#) or [Linux](#) command-line to easily copy data to and from Azure Blob, File, and Table storage with optimal performance. AzCopy supports concurrency and parallelism, and the ability to resume copy operations when interrupted. It is also faster than most other options. For programmatic access, the [Microsoft Azure Storage Data Movement Library](#) is the core framework that powers AzCopy. It is provided as a .NET Core library.
- **PowerShell.** The `Start-AzureStorageBlobCopy` [PowerShell cmdlet](#) is an option for Windows administrators who are used to PowerShell.
- **AdlCopy.** [AdlCopy](#) enables you to copy data from Azure Storage Blobs into Data Lake Store. It can also be used to copy data between two Azure Data Lake Store accounts. However, it cannot be used to copy data from Data Lake Store to Storage Blobs.
- **Distcp.** If you have an HDInsight cluster with access to Data Lake Store, you can use Hadoop ecosystem tools like [Distcp](#) to copy data to and from an HDInsight cluster storage (WASB) into a Data Lake Store account.

- **Sqoop.** [Sqoop](#) is an Apache project and part of the Hadoop ecosystem. It comes preinstalled on all HDInsight clusters. It allows data transfer between an HDInsight cluster and relational databases such as SQL, Oracle, MySQL, and so on. Sqoop is a collection of related tools, including import and export. Sqoop works with HDInsight clusters using either Azure Storage blobs or Data Lake Store attached storage.
- **PolyBase.** [PolyBase](#) is a technology that accesses data outside of the database through the T-SQL language. In SQL Server 2016, it allows you to run queries on external data in Hadoop or to import/export data from Azure Blob Storage. In Azure SQL Data Warehouse, you can import/export data from Azure Blob Storage and Azure Data Lake Store. Currently, PolyBase is the fastest method of importing data into SQL Data Warehouse.
- **Hadoop command line.** When you have data that resides on an HDInsight cluster head node, you can use the `hadoop-copyFromLocal` command to copy that data to your cluster's attached storage, such as Azure Storage blob or Azure Data Lake Store. In order to use the Hadoop command, you must first connect to the head node. Once connected, you can upload a file to storage.

## Graphical interface

Consider the following options if you are only transferring a few files or data objects and don't need to automate the process.

- **Azure Storage Explorer.** [Azure Storage Explorer](#) is a cross-platform tool that lets you manage the contents of your Azure storage accounts. It allows you to upload, download, and manage blobs, files, queues, tables, and Azure Cosmos DB entities. Use it with Blob storage to manage blobs and folders, as well as upload and download blobs between your local file system and Blob storage, or between storage accounts.
- **Azure portal.** Both Blob storage and Data Lake Store provide a web-based interface for exploring files and uploading new files one at a time. This is a good option if you do not want to install any tools or issue commands to quickly explore your files, or to simply upload a handful of new ones.

## Data pipeline

**Azure Data Factory.** [Azure Data Factory](#) is a managed service best suited for regularly transferring files between a number of Azure services, on-premises, or a combination of the two. Using Azure Data Factory, you can create and schedule data-driven workflows (called pipelines) that ingest data from disparate data stores. It can process and transform the data by using compute services such as Azure HDInsight Hadoop, Spark, Azure Data Lake Analytics, and Azure Machine Learning. Create data-driven workflows for [orchestrating](#) and automating data movement and data transformation.

## Key Selection Criteria

For data transfer scenarios, choose the appropriate system for your needs by answering these questions:

- Do you need to transfer very large amounts of data, where doing so over an Internet connection would take too long, be unreliable, or too expensive? If yes, consider physical transfer.
- Do you prefer to script your data transfer tasks, so they are reusable? If so, select one of the command line options or Azure Data Factory.
- Do you need to transfer a very large amount of data over a network connection? If so, selection an option that is optimized for big data.
- Do you need to transfer data to or from a relational database? If yes, choose an option that supports one or more relational databases. Note that some of these options also require a Hadoop cluster.
- Do you need an automated data pipeline or workflow orchestration? If yes, consider Azure Data Factory.

# Capability matrix

The following tables summarize the key differences in capabilities.

## Physical transfer

	AZURE IMPORT/EXPORT SERVICE	AZURE DATA BOX
Form factor	Internal SATA HDDs or SSDs	Secure, tamper-proof, single hardware appliance
Microsoft manages shipping logistics	No	Yes
Integrates with partner products	No	Yes
Custom appliance	No	Yes

## Command line tools

### Hadoop/HDInsight

	DISTCP	SQOOP	HADOOP CLI
Optimized for big data	Yes	Yes	Yes
Copy to relational database	No	Yes	No
Copy from relational database	No	Yes	No
Copy to Blob storage	Yes	Yes	Yes
Copy from Blob storage	Yes	Yes	No
Copy to Data Lake Store	Yes	Yes	Yes
Copy from Data Lake Store	Yes	Yes	No

## Other

	AZURE CLI	AZCOPY	POWERSHELL	ADLCOPY	POLYBASE
Compatible platforms	Linux, OS X, Windows	Linux, Windows	Windows	Linux, OS X, Windows	SQL Server, Azure SQL Data Warehouse
Optimized for big data	No	No	No	Yes <sup>1</sup>	Yes <sup>2</sup>
Copy to relational database	No	No	No	No	Yes
Copy from relational database	No	No	No	No	Yes

	AZURE CLI	AZCOPY	POWERSHELL	ADLCOPY	POLYBASE
Copy to Blob storage	Yes	Yes	Yes	No	Yes
Copy from Blob storage	Yes	Yes	Yes	Yes	Yes
Copy to Data Lake Store	No	No	Yes	Yes	Yes
Copy from Data Lake Store	No	No	Yes	Yes	Yes

[1] AdlCopy is optimized for transferring big data when used with a Data Lake Analytics account.

[2] PolyBase [performance can be increased](#) by pushing computation to Hadoop and using [PolyBase scale-out groups](#) to enable parallel data transfer between SQL Server instances and Hadoop nodes.

### Graphical interface and Azure Data Factory

	AZURE STORAGE EXPLORER	AZURE PORTAL *	AZURE DATA FACTORY
Optimized for big data	No	No	Yes
Copy to relational database	No	No	Yes
Copy to relational database	No	No	Yes
Copy to Blob storage	Yes	No	Yes
Copy from Blob storage	Yes	No	Yes
Copy to Data Lake Store	No	No	Yes
Copy from Data Lake Store	No	No	Yes
Upload to Blob storage	Yes	Yes	Yes
Upload to Data Lake Store	Yes	Yes	Yes
Orchestrate data transfers	No	No	Yes
Custom data transformations	No	No	Yes
Pricing model	Free	Free	Pay per usage

\* Azure portal in this case means using the web-based exploration tools for Blob storage and Data Lake Store.

# Extending on-premises data solutions to the cloud

2/13/2018 • 7 min to read • [Edit Online](#)

When organizations move workloads and data to the cloud, their on-premises datacenters often continue to play an important role. The term *hybrid cloud* refers to a combination of public cloud and on-premises data centers, to create an integrated IT environment that spans both. Some organizations use hybrid cloud as a path to migrate their entire datacenter to the cloud over time. Other organizations use cloud services to extend their existing on-premises infrastructure.

This article describes some considerations and best practices for managing data in a hybrid cloud solution,

## When to use a hybrid solution

Consider using a hybrid solution in the following scenarios:

- As a transition strategy during a longer-term migration to a fully cloud native solution.
- When regulations or policies do not permit moving specific data or workloads to the cloud.
- For disaster recovery and fault tolerance, by replicating data and services between on-premises and cloud environments.
- To reduce latency between your on-premises data center and remote locations, by hosting part of your architecture in Azure.

## Challenges

- Creating a consistent environment in terms of security, management, and development, and avoiding duplication of work.
- Creating a reliable, low latency and secure data connection between your on-premises and cloud environments.
- Replicating your data and modifying applications and tools to use the correct data stores within each environment.
- Securing and encrypting data that is hosted in the cloud but accessed from on-premises, or vice versa.

## On-premises data stores

On-premises data stores include databases and files. There may be several reasons to keep these local. There may be regulations or policies that do not permit moving specific data or workloads to the cloud. Data sovereignty, privacy, or security concerns may favor on-premises placement. During a migration, you may want to keep some data local to an application that hasn't been migrated yet.

Considerations in placing application data in a public cloud include:

- **Cost.** The cost of storage in Azure can be significantly lower than the cost of maintaining storage with similar characteristics in an on-premises datacenter. Of course, many companies have existing investments in high-end SANs, so these cost advantages may not reach full fruition until existing hardware ages out.
- **Elastic scale.** Planning and managing data capacity growth in an on-premises environment can be challenging, particularly when data growth is difficult to predict. These applications can take advantage of the capacity-on-demand and virtually unlimited storage available in the cloud. This consideration is less relevant for applications that consist of relatively static sized datasets.



- **Disaster recovery.** Data stored in Azure can be automatically replicated within an Azure region and across geographic regions. In hybrid environments, these same technologies can be used to replicate between on-premises and cloud-based data stores.

## Extending data stores to the cloud

There are several options for extending on-premises data stores to the cloud. One option is to have on-premises and cloud replicas. This can help achieve a high level of fault tolerance, but may require making changes to applications to connect to the appropriate data store in the event of a failover.

Another option is to move a portion of the data to cloud storage, while keeping the more current or more highly accessed data on-premises. This method can provide a more cost-effective option for long-term storage, as well as improve data access response times by reducing your operational data set.

A third option is to keep all data on-premises, but use cloud computing to host applications. To do this, you would host your application in the cloud and connect it to your on-premises data store over a secure connection.

## Azure Stack

For a complete hybrid cloud solution, consider using [Microsoft Azure Stack](#). Azure Stack is a hybrid cloud platform that lets you provide Azure services from your datacenter. This helps maintain consistency between on-premises and Azure, by using identical tools and requiring no code changes.

The following are some use cases for Azure and Azure Stack:

- **Edge and disconnected solutions.** Address latency and connectivity requirements by processing data locally in Azure Stack and then aggregating in Azure for further analytics, with common application logic across both.
- **Cloud applications that meet varied regulations.** Develop and deploy applications in Azure, with the flexibility to deploy the same applications on-premises on Azure Stack to meet regulatory or policy requirements.
- **Cloud application model on-premises.** Use Azure to update and extend existing applications or build new ones. Use a consistent DevOps processes across Azure in the cloud and Azure Stack on-premises.

## SQL Server data stores

If you are running SQL Server on-premises, you can use Microsoft Azure Blob Storage service for backup and restore. For more information, see [SQL Server Backup and Restore with Microsoft Azure Blob Storage Service](#). This capability gives you limitless off-site storage, and the ability to share the same backups between SQL Server running on-premises and SQL Server running in a virtual machine in Azure.

[Azure SQL Database](#) is a managed relational database-as-a service. Because Azure SQL Database uses the Microsoft SQL Server Engine, applications can access data in the same way with both technologies. Azure SQL Database can also be combined with SQL Server in useful ways. For example, the [SQL Server Stretch Database](#) feature lets an application access what looks like a single table in a SQL Server database while some or all rows of that table might be stored in Azure SQL Database. This technology automatically moves data that's not accessed for a defined period of time to the cloud. Applications reading this data are unaware that any data has been moved to the cloud.

Maintaining data stores on-premises and in the cloud can be challenging when you desire to keep the data synchronized. You can address this with [SQL Data Sync](#), a service built on Azure SQL Database that lets you synchronize the data you select, bi-directionally across multiple Azure SQL databases and SQL Server instances. While Data Sync makes it easy to keep your data up-to-date across these various data stores, it should not be used for disaster recovery or for migrating from on-premises SQL Server to Azure SQL Database.

For disaster recovery and business continuity, you can use [AlwaysOn Availability Groups](#) to replicate data across

two or more instances of SQL Server, some of which can be running on Azure virtual machines in another geographic region.

## Network shares and file-based data stores

In a hybrid cloud architecture, it is common for an organization to keep newer files on-premises while archiving older files to the cloud. This is sometimes called file tiering, where there is seamless access to both sets of files, on-premises and cloud-hosted. This approach helps to minimize network bandwidth usage and access times for newer files, which are likely to be accessed the most often. At the same time, you get the benefits of cloud-based storage for archived data.

Organizations may also wish to move their network shares entirely to the cloud. This would be desirable, for example, if the applications that access them are also located in the cloud. This procedure can be done using [data orchestration](#) tools.

[Azure StorSimple](#) offers the most complete integrated storage solution for managing storage tasks between your on-premises devices and Azure cloud storage. StorSimple is an efficient, cost-effective, and easily manageable storage area network (SAN) solution that eliminates many of the issues and expenses associated with enterprise storage and data protection. It uses the proprietary StorSimple 8000 series device, integrates with cloud services, and provides a set of integrated management tools.

Another way to use on-premises network shares alongside cloud-based file storage is with [Azure Files](#). Azure Files offers fully managed file shares that you can access with the standard [Server Message Block](#) (SMB) protocol (sometimes referred to as CIFS). You can mount Azure Files as a file share on your local computer, or use them with existing applications that access local or network share files.

To synchronize file shares in Azure Files with your on-premises Windows Servers, use [Azure File Sync](#). One major benefit of Azure File Sync is the ability to tier files between your on-premises file server and Azure Files. This lets you keep only the newest and most recently accessed files locally.

For more information, see [Deciding when to use Azure Blob storage, Azure Files, or Azure Disks](#).

## Hybrid networking

This article focused on hybrid data solutions, but another consideration is how to extend your on-premises network to Azure. For more information about this aspect of hybrid solutions, see the following topics:

- [Choose a solution for connecting an on-premises network to Azure](#)
- [Hybrid network reference architectures](#)

# Securing data solutions

2/13/2018 • 5 min to read • [Edit Online](#)

For many, making data accessible in the cloud, particularly when transitioning from working exclusively in on-premises data stores, can cause some concern around increased accessibility to that data and new ways in which to secure it.

## Challenges

- Centralizing the monitoring and analysis of security events stored in numerous logs.
- Implementing encryption and authorization management across your applications and services.
- Ensuring that centralized identity management works across all of your solution components, whether on-premises or in the cloud.

## Data Protection

The first step to protecting information is identifying what to protect. Develop clear, simple, and well-communicated guidelines to identify, protect, and monitor the most important data assets anywhere they reside. Establish the strongest protection for assets that have a disproportionate impact on the organization's mission or profitability. These are known as high value assets, or HVAs. Perform stringent analysis of HVA lifecycle and security dependencies, and establish appropriate security controls and conditions. Similarly, identify and classify sensitive assets, and define the technologies and processes to automatically apply security controls.

Once the data you need to protect has been identified, consider how you will protect the data *at rest* and data *in transit*.

- **Data at rest:** Data that exists statically on physical media, whether magnetic or optical disk, on premises or in the cloud.
- **Data in transit:** Data while it is being transferred between components, locations or programs, such as over the network, across a service bus (from on-premises to cloud and vice-versa), or during an input/output process.

To learn more about protecting your data at rest or in transit, see [Azure Data Security and Encryption Best Practices](#).

## Access Control

Central to protecting your data in the cloud is a combination of identity management and access control. Given the variety and type of cloud services, as well as the rising popularity of [hybrid cloud](#), there are several key practices you should follow when it comes to identity and access control:

- Centralize your identity management.
- Enable Single Sign-On (SSO).
- Deploy password management.
- Enforce multi-factor authentication (MFA) for users.
- Use role based access control (RBAC).
- Conditional Access Policies should be configured, which enhances the classic concept of user identity with additional properties related to user location, device type, patch level, and so on.
- Control locations where resources are created using resource manager.
- Actively monitor for suspicious activities

For more information, see [Azure Identity Management and access control security best practices](#).

## Auditing

Beyond the identity and access monitoring previously mentioned, the services and applications that you use in the cloud should be generating security-related events that you can monitor. The primary challenge to monitoring these events is handling the quantities of logs, in order to avoid potential problems or troubleshoot past ones. Cloud-based applications tend to contain many moving parts, most of which generate some level of logging and telemetry. Use centralized monitoring and analysis to help you manage and make sense of the large amount of information.

For more information, see [Azure Logging and Auditing](#).

## Securing data solutions in Azure

### Encryption

**Virtual machines.** Use [Azure Disk Encryption](#) to encrypt the attached disks on Windows or Linux VMs. This solution integrates with [Azure Key Vault](#) to control and manage the disk-encryption keys and secrets.

**Azure Storage.** Use [Azure Storage Service Encryption](#) to automatically encrypt data at rest in Azure Storage. Encryption, decryption, and key management are totally transparent to users. Data can also be secured in transit by using client-side encryption with Azure Key Vault. For more information, see [Client-Side Encryption and Azure Key Vault for Microsoft Azure Storage](#).

**SQL Database and Azure SQL Data Warehouse.** Use [Transparent Data Encryption](#) (TDE) to perform real-time encryption and decryption of your databases, associated backups, and transaction log files without requiring any changes to your applications. SQL Database can also use [Always Encrypted](#) to help protect sensitive data at rest on the server, during movement between client and server, and while the data is in use. You can use Azure Key Vault to store your Always Encrypted encryption keys.

### Rights management

[Azure Rights Management](#) is a cloud-based service that uses encryption, identity, and authorization policies to secure files and email. It works across multiple devices—phones, tablets, and PCs. Information can be protected both within your organization and outside your organization because that protection remains with the data, even when it leaves your organization's boundaries.

### Access control

Use [Role-Based Access Control](#) (RBAC) to restrict access to Azure resources based on user roles. If you are using Active Directory on-premises, you can [synchronize with Azure AD](#) to provide users with a cloud identity based on their on-premises identity.

Use [Conditional access in Azure Active Directory](#) to enforce controls on the access to applications in your environment based on specific conditions. For example, your policy statement could take the form of: *When contractors are trying to access our cloud apps from networks that are not trusted, then block access.*

[Azure AD Privileged Identity Management](#) can help you manage, control, and monitor your users and what sorts of tasks they are performing with their admin privileges. This is an important step to limiting who in your organization can carry out privileged operations in Azure AD, Azure, Office 365, or SaaS apps, as well as monitor their activities.

### Network

To protect data in transit, always use SSL/TLS when exchanging data across different locations. Sometimes you need to isolate your entire communication channel between your on-premises and cloud infrastructure by using either a virtual private network (VPN) or [ExpressRoute](#). For more information, see [Extending on-premises data solutions to the cloud](#).

Use [network security groups](#) (NSGs) to reduce the number of potential attack vectors. A network security group contains a list of security rules that allow or deny inbound or outbound network traffic based on source or destination IP address, port, and protocol.

Use [Virtual Network service endpoints](#) to secure Azure SQL or Azure Storage resources, so that only traffic from your virtual network can access these resources.

VMs within an Azure Virtual Network (VNet) can securely communicate with other VNets using [virtual network peering](#). Network traffic between peered virtual networks is private. Traffic between the virtual networks is kept on the Microsoft backbone network.

For more information, see [Azure network security](#)

## **Monitoring**

[Azure Security Center](#) automatically collects, analyzes, and integrates log data from your Azure resources, the network, and connected partner solutions, such as firewall solutions, to detect real threats and reduce false positives.

[Log Analytics](#) provides centralized access to your logs and helps you analyze that data and create custom alerts.

[Azure SQL Database Threat Detection](#) detects anomalous activities indicating unusual and potentially harmful attempts to access or exploit databases. Security officers or other designated administrators can receive an immediate notification about suspicious database activities as they occur. Each notification provides details of the suspicious activity and recommends how to further investigate and mitigate the threat.