**AZURE Data Engineer DP-203 Notes**

Over the last decade, the amount of data that systems and devices generate has increased significantly. Because of this increase, new technologies, roles, and approaches to working with data are affecting data professionals. In many industries, data professionals want to understand better how these changes affect both their careers and their daily working lives.

To generate value, anyone working with data needs to understand how the data landscape has changed and how roles and technologies are evolving. You should be able to explain this shift to any stakeholder:

* [Data abundance](https://docs.microsoft.com/en-us/learn/modules/evolving-world-of-data/2-data-abundance),
* [Understand the difference between on-premises and cloud-based servers](https://docs.microsoft.com/en-us/learn/modules/evolving-world-of-data/3-systems-on-premise-vs-cloud),
* [Understand job responsibilities](https://docs.microsoft.com/en-us/learn/modules/evolving-world-of-data/4-understand-job-responsibility),
* [Identify the tasks of a data engineer in a cloud-hosted architecture](https://docs.microsoft.com/en-us/learn/modules/data-engineering-processes/),
* [Decide how many storage accounts you need](https://docs.microsoft.com/en-us/learn/modules/create-azure-storage-account/2-decide-how-many-storage-accounts-you-need)
* [Use the REST API for Storage accounts](https://docs.microsoft.com/en-us/learn/modules/connect-an-app-to-azure-storage/5-interact-with-azure-storage-apis)
* [Azure Storage Services REST API Reference](https://docs.microsoft.com/en-us/rest/api/storageservices/)
* [Use a client library for storage accounts](https://docs.microsoft.com/en-us/learn/modules/connect-an-app-to-azure-storage/5-interact-with-azure-storage-apis)
* [Azure Storage Blobs client library for .NET GitHub page](https://github.com/Azure/azure-sdk-for-net/tree/Azure.Storage.Blobs_12.7.0/sdk/storage/Azure.Storage.Blobs)
* [Azure Storage Client Library for JavaScript](https://github.com/Azure/azure-storage-node#azure-storage-javascript-client-library-for-browsers)
* [SDKs and tools](https://azure.microsoft.com/en-gb/downloads/)
* [All About SSIS](https://docs.microsoft.com/en-us/learn/modules/execute-existing-ssis-packages-azure-data-factory/1-introduction)
* [Operationalize your Azure Data Factory or Azure Synapse Pipeline](https://docs.microsoft.com/en-us/learn/modules/operationalize-azure-data-factory-pipelines/?source=learn)
* [Azure DevOps CICD](https://docs.microsoft.com/en-us/learn/modules/operationalize-azure-data-factory-pipelines/4-continuous-integration-deployment)
* [Hotfix production environment](https://docs.microsoft.com/en-us/learn/modules/operationalize-azure-data-factory-pipelines/4-continuous-integration-deployment)
* [Data factory Monitoring and Alerts using Azure Monitor](https://docs.microsoft.com/en-us/learn/modules/operationalize-azure-data-factory-pipelines/5-monitor)
* [Power BI in Synapse](https://docs.microsoft.com/en-gb/learn/modules/survey-components-of-azure-synapse-analytics/9-visualize-your-analytics-with-power-bi)
* [Azure Synapse Link](https://docs.microsoft.com/en-gb/learn/modules/survey-components-of-azure-synapse-analytics/10-understand-hybrid-transactional-analytical-processing)
* [Design a data warehouse schema or Model](https://docs.microsoft.com/en-gb/learn/modules/design-multidimensional-schema-to-optimize-analytical-workloads/2-design-star-schema)
* [Indexes on dedicated SQL pool tables](https://docs.microsoft.com/en-us/azure/synapse-analytics/sql-data-warehouse/sql-data-warehouse-tables-index)
* [Guidance for designing distributed tables using dedicated SQL pool in Azure Synapse Analytics](https://docs.microsoft.com/en-us/azure/synapse-analytics/sql-data-warehouse/sql-data-warehouse-tables-distribute)
* [Design guidance for using replicated tables in Synapse SQL pool](https://docs.microsoft.com/en-us/azure/synapse-analytics/sql-data-warehouse/design-guidance-for-replicated-tables)
* [Columnstore indexes: Overview](https://docs.microsoft.com/en-us/sql/relational-databases/indexes/columnstore-indexes-overview?toc=%2Fazure%2Fsynapse-analytics%2Fsql-data-warehouse%2Ftoc.json&bc=%2Fazure%2Fsynapse-analytics%2Fsql-data-warehouse%2Fbreadcrumb%2Ftoc.json&view=azure-sqldw-latest&preserve-view=true)

Executives have asked the network infrastructure team to explain the benefits and drawbacks of running IT operations in Azure. The network team approaches you for information about Azure data services. Could you answer their high-level questions?[Survey the services on the Azure Data platform](https://docs.microsoft.com/en-us/learn/modules/survey-the-azure-data-platform/1-introduction), [Choose a data storage approach in Azure,deciding what storage solution to use](https://docs.microsoft.com/en-us/learn/modules/choose-storage-approach-in-azure/3-operations-and-latency)

1. **Cloud Vs On-Premise:**
   1. **Multilingual support in On-premise Support**:One issue with multiple languages is the sorting order of text data. Different languages can sort text data differently. To address this issue, the SQL Server database administrator must install and configure the data's collation settings. But these settings can work only if the SQL database developers considered multilingual functionality when they were designing the system. Systems like this are complex to manage and maintain
   2. **The term total cost of ownership (TCO)** describes the final cost of owning a given technology. In on-premises systems, TCO includes the following costs:
   3. Hardware
   4. Software licensing
   5. Labor (installation, upgrades, maintenance)
   6. Datacenter overhead (power, telecommunications, building, heating and cooling)
   7. Because on-premises server systems are very expensive, costs are often capitalized. This means that on financial statements, costs are spread out across the expected lifetime of the server equipment. Capitalization restricts an IT manager's ability to buy upgraded server equipment during the expected lifetime of a server. This restriction limits the server system's ability to accommodate increased demand.
   8. **In the cloud**, Microsoft manages many operations to create a stable computing environment. This service is part of the Azure product benefit. Microsoft manages key infrastructure services such as physical hardware, computer networking, firewalls and network security, datacenter fault tolerance, compliance, and physical security of the buildings. Microsoft also invests heavily to battle cybersecurity threats, and it updates operating systems and firmware for the customer. These services allow data engineers to focus more on data engineering and eliminating system complexity.
   9. **Multilingual support in On-Cloud**: Cloud systems often store data as a JSON file that includes the language code identifier (LCID). The LCID identifies the language that the data uses. Apps that process the data can use translation services such as the Bing Translator API to convert the data into an expected language when the data is consumed or as part of a process to prepare the data.
   10. **Reduce Underutilization**:Organizations can reduce the costs of underutilization by adopting a best practice to provision production instances only after their developers are ready to deploy an application to production. Developers can use tools like the Azure Cosmos DB emulator or the Azure Storage emulator to develop and test cloud applications without incurring production costs.
   11. When moving to the cloud, many customers migrate from physical or virtualized on-premises servers to Azure Virtual Machines. This strategy is known as **lift and shift**.Server administrators lift and shift an application from a physical environment to Azure Virtual Machines without rearchitecting the application.The disadvantage is that the application can't take advantage of the many features available within Azure.Your rearchitected application can take advantage of Azure offerings such as Cognitive Services, Bot Service, and machine learning capabilities.
2. **Understand job responsibilities:**
   1. If you're a SQL Server professional, over time you'll focus less on SQL Server and more on data in general. You'll be a **data engineer**.To master data engineering, you'll need to learn a new set of tools, architectures, and platforms. As a SQL Server professional, your primary data manipulation tool might be T-SQL. As a data engineer, you might use other technologies, like Azure HDInsight and Azure Cosmos DB. To manipulate the data in big-data systems, you might use languages such as HiveQL or Python.
   2. **Transformation:** transform the data from the source schema to the destination schema.
   3. **ETL**: A disadvantage of the ETL approach is that the transformation stage can take a long time. This stage can potentially tie up source system resources. This process is typically used for ingesting data from an on-premises database to an on-premises data warehouse.
   4. **ELT**:An alternative approach is extract, load, and transform (ELT). In ELT, the data is immediately extracted and loaded into a large data repository, such as Azure Cosmos DB or Azure Data Lake Storage.
      * This change in process reduces the resource contention on source systems. Data engineers can begin transforming the data as soon as the load is complete.
      * ELT also has more architectural flexibility to support multiple transformations. For example, how the marketing department needs to transform the data can be different than how the operations department needs to transform that same data. This process is typically used for ingesting data from an on-premises database into the cloud.The benefit of ELT is that you can store data in its original format, be it JSON, XML, PDF, or images. In ELT, you define the data's structure during the transformation phase, so you can use the source data in multiple downstream systems.
      * Another process like ELT is called extract, load, transform, and load (ELTL). The difference with ELTL is that it has a final load into a destination system.
      * In an ELT process, data is extracted and loaded in its native format. This change reduces the time required to load the data into a destination system. The change also limits resource contention on the data sources.
      * Having a staging area enables you to deal with the ingestion of source systems on different schedules.
      * A staging environment provides the opportunity to bring together a single view of data from different source systems.
      * To rerun failed data warehouse loads from a staging area. By holding onto the staging data, you are able to rerun the ETL process from the staging area, rather than the source system.
   5. Data engineers must also get, ingest, transform, validate, and clean up data to meet business requirements. This process is called **data wrangling**
   6. Both **database administrators** and **business intelligence professionals** can easily transition to a data engineer role. They just need to learn the tools and technology that are used to process large amounts of data
   7. When a Cognitive Services application reaches its capacity, AI engineers call on data scientists. Data scientists develop machine learning models and customize components for an AI engineer's application.
3. **Use cases for the cloud:**

* **Web**: use the Azure Cosmos DB multimaster replication model. It can achieve a response time of less than 10 ms anywhere in the world.
* **Healthcare**: use Azure Databricks to accelerate big-data analytics and AI solutions. Apply these technologies to genome studies or pharmacy sales forecasting at a petabyte scale. Using Databricks features, you can set up your Spark environment in minutes and autoscale quickly and easily.
* **IoT solutions**: Use technologies like Azure IoT Hub. Over the last couple of years, hundreds of thousands of devices have been produced to generate sensor data. These are known as IoT devices.

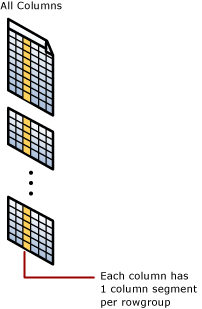
1. **Structured and Nonstructured data:**In relational database systems like Microsoft SQL Server, Azure SQL Database, and Azure SQL Data Warehouse, data structure is defined at design time. Relational systems react slowly to changes in data requirements. Nonstructured data is stored in nonrelational systems, commonly called unstructured or NoSQL systems. In nonrelational systems, the data structure isn't defined at design time, and data is typically loaded in its raw format. The data structure is defined only when the data is read. The difference in the definition point gives you flexibility to use the same source data for different outputs. Nonrelational systems can also support semistructured data such as JSON file formats. The open-source world offers four types of NoSQL databases:
   1. Key-value store: Stores key-value pairs of data in a table structure.
   2. Document database: Stores documents that are tagged with metadata to aid document searches.
   3. Graph database: Finds relationships between data points by using a structure that's composed of vertices and edges.
   4. Column database: Stores data based on columns rather than rows. Columns can be defined at the query's runtime, allowing flexibility in the data that's returned performantly.
2. **Index**: An index is an on-disk structure associated with a table or view that speeds retrieval of rows from the table or view. An index contains keys built from one or more columns in the table or view. These keys are stored in a structure (B-tree) that enables SQL Server to find the row or rows associated with the key values quickly and efficiently.Indexes can be unique. This means no two rows can have the same value for the index key. Otherwise, the index is not unique and multiple rows can share the same key value.A table or view can contain the following types of indexes:
   1. **Clustered**: A clustered index is **an index which defines the physical order in which table records are stored in a database**. Since there can be only one way in which records are physically stored in a database table, there can be only one clustered index per table. By default a clustered index is created on a primary key column. When a table has a clustered index, the table is called a clustered table. If a table has no clustered index, its data rows are stored in an unordered structure called a **heap**
   2. **Nonclustered**:A nonclustered index is an index structure separate from the data stored in a table that **reorders one or more selected columns**.A nonclustered index contains the nonclustered index key values and each key value entry has a pointer to the data row that contains the key value.The pointer from an index row in a nonclustered index to a data row is called a row locator.The structure of the row locator depends on whether the data pages are stored in a heap or a clustered table. For a heap, a row locator is a pointer to the row. For a clustered table, the row locator is the clustered index key.
   3. **Heap**:A heap is a **table without a clustered index**.One or more nonclustered indexes can be created on tables stored as a heap. Data is stored in the heap without specifying an order. Usually data is initially stored in the order in which is the rows are inserted into the table, but the Database Engine can move data around in the heap to store the rows efficiently; so the data order cannot be predicted. To guarantee the order of rows returned from a heap, you must use the **ORDER BY** clause. To specify a permanent logical order for storing the rows, **create a clustered index on the table, so that the table is not a heap.**

**When to Use a Heap**: Most tables should have a carefully chosen clustered index unless a good reason exists for leaving the table as a heap. When a table is stored as a heap, individual rows are identified by reference to an 8-byte row identifier (RID) consisting of the file number, data page number, and slot on the page (FileID:PageID:SlotID). The row ID is a small and efficient structure.Heaps can be used as staging tables for large, unordered insert operations. Because data is inserted without enforcing a strict order, the insert operation is usually faster than the equivalent insert into a clustered index. If the heap's data will be read and processed into a final destination, it may be useful to create a narrow nonclustered index that covers the search predicate used by the read query.

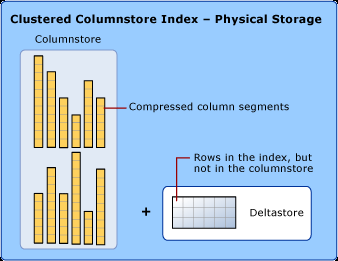
**Data is retrieved from a heap in order of data pages, but not necessarily the order in which data was inserted.**

Indexes are automatically created when PRIMARY KEY and UNIQUE constraints are defined on table columns. For example, when you create a table with a UNIQUE constraint, Database Engine automatically creates a nonclustered index. If you configure a PRIMARY KEY, Database Engine automatically creates a clustered index, unless a clustered index already exists. When you try to enforce a PRIMARY KEY constraint on an existing table and a clustered index already exists on that table, SQL Server enforces the primary key using a nonclustered index.

**Columnstore indexes**:A **columnstore index** is a technology for storing, retrieving, and managing data by using a columnar data format, called a columnstore.Columnstore indexes are the standard for storing and querying large data warehousing fact tables. This index uses column-based data storage and query processing to achieve gains up to 10 times the query performance in your data warehouse over traditional row-oriented storage.Beginning with SQL Server 2016 (13.x) SP1, columnstore indexes enable operational analytics: the ability to run performant real-time analytics on a transactional workload.A **columnstore** is data that's logically organized as a table with rows and columns, and physically stored in a column-wise data format.A **rowstore** is data that's logically organized as a table with rows and columns, and physically stored in a row-wise data format. This format is the traditional way to store relational table data. In SQL Server, rowstore refers to a table where the underlying data storage format is a heap, a clustered index, or a memory-optimized table.A **rowgroup** is a group of rows that are compressed into columnstore format at the same time. A rowgroup usually contains the maximum number of rows per rowgroup, which is 1,048,576 rows.For high performance and high compression rates, the columnstore index slices the table into rowgroups, and then compresses each rowgroup in a column-wise manner. The number of rows in the rowgroup must be large enough to improve compression rates, and small enough to benefit from in-memory operations.A **column segment** is a column of data from within the rowgroup. (1. Each rowgroup contains one column segment for every column in the table. 2. Each column segment is compressed together and stored on physical media.)



**Clustered columnstore index**: A clustered columnstore index is the physical storage for the entire table.



To reduce fragmentation of the column segments and improve performance, the columnstore index might store some data temporarily into a clustered index called a deltastore and a B-tree list of IDs for deleted rows. The deltastore operations are handled behind the scenes. To return the correct query results, the clustered columnstore index combines query results from both the columnstore and the deltastore.[More on Columnstore indexes: Overview](https://docs.microsoft.com/en-us/sql/relational-databases/indexes/columnstore-indexes-overview?toc=%2Fazure%2Fsynapse-analytics%2Fsql-data-warehouse%2Ftoc.json&bc=%2Fazure%2Fsynapse-analytics%2Fsql-data-warehouse%2Fbreadcrumb%2Ftoc.json&view=azure-sqldw-latest&preserve-view=true)

1. **Data Services on Azure:**
   1. **Azure Storage**:
   * Azure Data Lake Storage is a Hadoop-compatible data repository that can store any size or type of data.
   * If you need to provision a data store that will store but not query data, your cheapest option is to set up a storage account as a Blob store.
   * Supported languages, runtime environments, and developer platforms include, .NET, Node.js, Java, Python, PHP, Ruby, and Go. Azure Storage also supports scripting in Azure PowerShell and the Azure CLI.
   * If you use the File Upload feature to import file sizes above 2 GB, use PowerShell or Visual Studio. AzCopy supports a maximum file size of 1 TB and automatically splits data files that exceed 200 GB.
   * In Gen2, developers can access data through either the Blob API or the Data Lake file API. Data Lake Storage (Gen2) is designed to store massive amounts of data for big-data analytics.
   * An optimized Azure Blob File System (ABFS) driver that's designed for big-data analytics.
   * In Data Lake Storage Gen1, data engineers **query data by using the U-SQL language**. In Gen 2, use the Azure Blob Storage API or the Azure Data Lake System (ADLS) API.
   1. **Azure Cosmos DB**:
   * Azure Cosmos DB is a globally distributed, multimodel database. You can deploy it by using several API models: SQL API, MongoDB API, Cassandra API, Gremlin API, Table API. When you move your data from SQL, MongoDB, or Cassandra to Azure Cosmos DB, applications that are built using the SQL, MongoDB, or Cassandra APIs will continue to operate.
   * Deploy Azure Cosmos DB when you need a NoSQL database of the supported API model, at planet scale, and with low latency performance. Currently, Azure Cosmos DB supports five-nines uptime (99.999 percent). It can support response times below 10 ms when it's provisioned correctly.
   * Consistency levels in Azure Cosmos DB include: Strong, Bounded staleness, Session, Consistent prefix, Eventual.
   * To ingest data into Azure Cosmos DB, use Azure Data Factory, create an application that writes data into Azure Cosmos DB through its API, upload JSON documents, or directly edit the document.
   * As a data engineer, you can create stored procedures, triggers, and user-defined functions (UDFs). Or use the JavaScript query API. You'll also find other methods to query the other APIs within Azure Cosmos DB.
   * User authentication is based on tokens, and Azure Active Directory provides role-based security.
   * Azure Cosmos DB meets many security compliance certifications, including HIPAA, FedRAMP, SOX, and HITRUST.
   1. **Azure SQL Database**:
   * Use SQL Database when you need to scale up and scale down OLTP systems on demand.
   * SQL Database can ingest data through application integration from a wide range of developer SDKs. Supported programming languages, runtime environments, and developer platforms include, .NET, Node.js, Python, and Java. Beyond applications, you can also ingest data through Transact-SQL (T-SQL) techniques and from the movement of data using Azure Data Factory.
   * Use T-SQL to query the contents of a SQL Database.
   1. **Azure Synapse Analytics**:
   * Azure Synapse Analytics is a cloud-based data platform that brings together enterprise data warehousing and Big Data analytics. It can process massive amounts of data and answer complex business questions with limitless scale
   * Used to reduce processing time and release business intelligence reports faster with solution on a petabyte scale.
   * The volume and variety of data that is being generated are providing opportunities to perform different types of analysis on the data. This can include techniques such as exploratory data analysis to identify initial patterns or meaning in the data. It can also include conducting predictive analytics for forecasting, or segmenting data. The Big Data Analytics capability of Azure Synapse Analytics will accommodate this.
   * SQL Pools uses massively parallel processing (MPP) to quickly run queries across petabytes of data
   * Because the storage is separated from the compute nodes, you can scale the compute nodes independently to meet any demand at any time.
   * In Azure Synapse Analytics, the Data Movement Service (DMS) coordinates and transports data between compute nodes as necessary. But you can use a replicated table to reduce data movement and improve performance. Azure Synapse Analytics supports three types of distributed tables: hash, round-robin and replicated. Use these tables to tune performance.
   * SQL professionals are already familiar with bulk-copy tools such as bcp and the SQLBulkCopy API. Data engineers who work with Azure Synapse Analytics will soon learn how quickly PolyBase can load data.
   * Developers use PolyBase to apply stored procedures, labels, views, and SQL to their applications. You can also use Azure Data Factory to ingest and process data using PolyBase too.
   * As a data engineer, you can use the familiar Transact-SQL to query the contents of Azure Synapse Analytics
   1. **Azure Stream Analytics**: Applications, sensors, monitoring devices, and gateways broadcast continuous event data known as data streams. Streaming data is high volume and has a lighter payload than nonstreaming systems.
   * Data engineers use Azure Stream Analytics to process streaming data and respond to data anomalies in real time.
   * You can use Stream Analytics for Internet of Things (IoT) monitoring, web logs, remote patient monitoring, and point of sale (POS) systems.
   * If your organization must respond to data events in real time or analyze large batches of data in a continuous time-bound stream, Stream Analytics is a good solution. Your organization must decide whether to work with streaming data or batch data.
   * Don't use batch systems for business intelligence systems that can't tolerate the predefined interval. For example, an autonomous vehicle can't wait for a batch system to adjust its driving. Similarly, a fraud-detection system must decline a questionable financial transaction in real time.
   * As a data engineer, set up data ingestion in Stream Analytics by configuring data inputs from first-class integration sources. These sources include Azure Event Hubs, Azure IoT Hub, and Azure Blob storage.
   * Bidirectional communication capabilities mean that while you receive data from devices, you can also send commands and policies back to devices.
   * Azure IoT Hub can also authenticate access between the IoT device and the IoT hub.
   * The Stream Analytics query language is consistent with the SQL language. If you're familiar with the SQL language, you can start creating jobs.
   1. **Azure HDInsight**: Azure HDInsight provides technologies to help you ingest, process, and analyze big data
   * It supports batch processing, data warehousing, IoT, and data science.
   * **HDInsight is a low-cost cloud solution**. It includes Apache Hadoop, Spark, Kafka, HBase, Storm, and Interactive Query.
   * Hadoop includes Apache Hive, HBase, Spark, and Kafka. Hadoop stores data in a file system (HDFS). Spark stores data in memory. This difference in storage makes Spark about 100 times faster.
   * HBase is a NoSQL database built on Hadoop. It's commonly used for search engines. HBase offers automatic failover.
   * Storm is a distributed real-time streaming analytics solution.
   * Kafka is an open-source platform that's used to compose data pipelines. It offers message queue functionality, which allows users to publish or subscribe to real-time data streams.
   * As a data engineer, use Hive to run ETL operations on the data you're ingesting. Or orchestrate Hive queries in Azure Data Factory.
   * In Hadoop, use Java and Python to process big data. Mapper consumes and analyzes input data. It then emits tuples that Reducer can analyze. Reducer runs summary operations to create a smaller combined result set.
   * Spark processes streams by using Spark Streaming. For machine learning, use the 200 preloaded Anaconda libraries with Python. Use GraphX for graph computations.
   * Developers can remotely submit and monitor jobs from Spark. Storm supports common programming languages like Java, C#, and Python.
   * Hadoop supports Pig and HiveQL languages. In Spark, data engineers use Spark SQL.
   * Hadoop supports encryption, Secure Shell (SSH), shared access signatures, and Azure Active Directory security
   1. **Azure Databricks:**
   * Databricks is a data analytics platform that's optimized for Azure. It provides one-click setup, streamlined workflows, and an interactive workspace for Spark-based applications.
   * You can use REST APIs to program clusters.
   1. **Data Factory**:
   * Data Factory is a cloud-integration service. It orchestrates the movement of data between various data stores.
   * Data Factory processes and transforms data by using compute services such as Azure HDInsight, Hadoop, Spark, and Azure Machine Learning
   * ADF provides robust resources and nearly 100 enterprise connectors.
   1. **Azure Purview**:
   * Azure Purview is a unified data governance service that helps you manage and govern your on-premises, multicloud, and software-as-a-service (SaaS) data.
   * Easily create a holistic, up-to-date map of your data landscape with automated data discovery, sensitive data classification, and end-to-end data lineage.
   * Empower data consumers to find valuable, trustworthy data.
2. **Data transformation** operations can include splitting, combining, deriving, adding, removing, or pivoting columns. You might also need to aggregate or merge data.
3. Although Extensible Markup Language (XML) was common in the past, most systems have migrated to JSON because of its flexibility as a semistructured data type
4. Data comes in different shapes and sizes, and no single storage solution fits all data. For example, an online retail website has a number of distinct data sets that are all used to run the business: product catalog data, media files like photos and videos, and financial business data. Each data set has different requirements, and it's your job to figure out which storage solution is best. The key factors to consider in deciding on the optimal storage solution are: how to classify your data, how your data will be used, and how you can get the best performance for your application.
5. **Classification Data:**
   1. **Structured Data:**
   * Structured data, sometimes referred to as relational data, is data that adheres to a strict schema, so all of the data has the same fields or properties. The shared schema allows this type of data to be easily searched with query languages such as SQL (Structured Query Language).
   * This capability makes this data style perfect for applications such as CRM systems, reservations, and inventory management.
   * it's easy to enter, query, and analyze.
   1. **Semi-structured data**:
   * Semi-structured data is less organized than structured data, and is not stored in a relational format, as the fields do not neatly fit into tables, rows, and columns.
   * Semi-structured data is also referred to as non-relational or NoSQL data.
   * The expression and structure of the data in Semi-structured data is defined by a **serialization language**. For software developers, data serialization languages are important because they can be used to write data stored in memory to a file, sent to another system, parsed and read.
   * Today, there are three common serialization languages you're likely to encounter:**XML, JSON and YAML.**XML tends to be more verbose making it larger to store, process, or pass over a network. The downside to JSON is that it tends to be more programmer-oriented making it harder for non-technical people to read and modify. In YAML, the data structure is defined by line separation and indentation, and reduces the dependency on structural characters like parentheses, commas and brackets. YAML format is more readable than JSON and is often used for configuration files that need to be written by people but parsed by programs. However, YAML is the newest of these data formats and doesn't have as much support in programming languages as JSON and XML.
   1. **Unstructured Data:** Examples of unstructured data include:
   * Media files, such as photos, videos, and audio files
   * Office files, such as Word documents
   * Text files
   * Log files
6. **Transactions:**

* They enable you to group series of data updates so that if one event in a series of updates fails, the entire series can be rolled back, or undone.
* A transaction is a logical group of database operations that execute together.
* Transactions are often defined by a set of four requirements, referred to as ACID guarantees. ACID stands for Atomicity, Consistency, Isolation, and Durability
  + Atomicity means a transaction must execute exactly once and must be atomic; either all of the work is done, or none of it is. Operations within a transaction usually share a common intent and are interdependent.
  + Consistency ensures that the data is consistent both before and after the transaction.
  + Isolation ensures that one transaction is not impacted by another transaction.
  + Durability means that the changes made due to the transaction are permanently saved in the system. Committed data is saved by the system so that even in the event of a failure and system restart, the data is available in its correct state.
* If your data benefits from ACID principles, then choose a storage solution that supports transactions.

1. **OLTP** & **OLAP:**

* Transactional databases are often called OLTP (Online Transaction Processing) systems. OLTP systems commonly support lots of users, have quick response times, and handle large volumes of data. They are also highly available (meaning they have very minimal downtime), and typically handle small or relatively simple transactions.
* On the contrary, OLAP (Online Analytical Processing) systems commonly support fewer users, have longer response times, can be less available, and typically handle large and complex transactions.

1. When **deciding what storage solution to use**, think about how your data will be used. How often will your data be accessed? Is your data read-only? Does query time matter? The answers to these questions will help you decide on the best storage solution for your data. E.g., Product Catalog in E-Commerce website.

* **Data classification**: Semi-structured because of the need to extend or modify the schema for new products
* **Operations**:
  + Customers require a high number of read operations, with the ability to query many fields within the database.
  + The business requires a high number of write operations to track its constantly changing inventory.
* **Latency & throughput**: High throughput and low latency.
* **Transactional support**: Because all of the data is both historical and yet changing, transactional support is required.

Recommended Azure Storage: **Azure Cosmos DB**

* Azure Cosmos DB supports **semi-structured data**, or NoSQL data, by design. So, supporting new fields, such as the "Bluetooth-enabled" field or any new fields you need in the future, is a given with Azure Cosmos DB
* Azure Cosmos DB **supports SQL for queries** and **every property is indexed by default**. You can create queries so that your customers can filter on any property in the catalog.
* Azure Cosmos DB is also **ACID-compliant**, so you can be assured that your transactions are completed according to those strict requirements.
* As an added plus, Azure Cosmos DB also enables you to replicate your data anywhere in the world with the click of a buttonto reduce latency.

1. Azure Blob storage supports storing files such as photos and videos. It also works with Azure Content Delivery Network (**CDN**) by caching the most frequently used content and storing it on edge servers.
2. **Storage accounts:**

* **Number of Storage accounts**:
  + A storage account represents a collection of settings like location, replication strategy, and subscription owner. You need one storage account for each group of settings that you want to apply to your data.In general, increased diversity means an increased number of storage accounts.
  + **Cost**: A storage account by itself has no financial cost; however, the settings you choose for the account do influence the cost of services in the account. Geo-redundant storage costs more than locally redundant storage. Premium performance and the Hot access tier increase the cost of blobs.
  + You can use multiple storage accounts to reduce costs. For example, you could partition your data into critical and non-critical categories. You could place your critical data into a storage account with geo-redundant storage and put your non-critical data in a different storage account with locally redundant storage.
  + A typical strategy is to start with an analysis of your data and create partitions that share characteristics like location, billing, and replication strategy, and then create one storage account for each partition.
* **Deployment Model**: A deployment model is the system Azure uses to organize your resources. The model defines the API that you use to create, configure, and manage those resources. Azure provides two deployment models.
  + **Resource Manager**: the current model that uses the Azure Resource Manager API.
  + **Classic**: a legacy offering that uses the Azure Service Management API.

Microsoft recommends that you use Resource Manager for all new resources.

* The core advice here is to choose the Resource Manager deployment model and the StorageV2 (general purpose v2) account kind for all your storage accounts. There are a few special cases that can be exceptions to this rule. For example, pricing for transactions is lower in general purpose v1, which would allow you to slightly reduce costs if that matches your typical workload.
* The Storage **REST APIs** are accessible from anywhere on the Internet by any app that can send an HTTP/HTTPS request and receive an HTTP/HTTPS response.
* However, API approach requires a lot of manual parsing of output XML and the creation of HTTP packets to work with each API. For this reason, Azure provides **pre-built client libraries** that make working with the service easier for common languages and frameworks.Client libraries can save a significant amount of work for app developers because the API has been tested and often provides nicer wrappers around the data models sent and received by the REST API.Microsoft has Azure client libraries that support a number of languages and frameworks, including:**NET, Java, Python, Node.js, Go**.
* The **client libraries** are just thin wrappers around the REST API. They are doing exactly what you would do if you used the web services directly. These libraries are also open source, making them very transparent.
* **Key Vaults** include support to synchronize directly to the Storage Account and automatically rotate the keys periodically. Using a Key Vault provides an additional layer of security, so your app never has to work directly with an access key.
* Storage accounts offer a separate authentication mechanism called **shared access signatures** that support expiration and limited permissions for scenarios where you need to grant limited access. You should use this approach when you are allowing other users to read and write data to your storage account.
* **Security:**
  + **Encryption at rest**: All data written to Azure Storage is automatically encrypted by Storage Service Encryption (**SSE**) with a 256-bit Advanced Encryption Standard (AES) cipher, and is FIPS 140-2 compliant. For virtual machines (VMs), Azure lets you encrypt virtual hard disks (VHDs) by using Azure Disk Encryption. This encryption uses **BitLocker** for Windows images, and it uses **dm-crypt** for Linux.
  + **Encryption in transit**: Keep your data secure by enabling transport-level security between Azure and the client.
  + **CORS support**:Azure Storage supports cross-domain access through cross-origin resource sharing (CORS). CORS uses HTTP headers so that a web application at one domain can access resources from a server at a different domain. CORS support is an optional flag you can enable on Storage accounts. The flag adds the appropriate headers when you use HTTP GET requests to retrieve resources from the Storage account.
  + **RBAC**:The storage service ensures that the client has the permissions required to access the data. You can choose from several access options. Arguably, the most flexible option is role-based access.Azure Storage supports Azure Active Directory and role-based access control (RBAC) for both resource management and data operations.  
    To a security principal or a managed identity for Azure resources, you can assign RBAC roles that are scoped to a subscription, a resource group, a storage account, or an **individual container or queue**.
  + **Auditing access**: Auditing is another part of controlling access. You can audit Azure Storage access by using the built-in Storage Analytics service.Storage Analytics logs every operation in real time, and you can search the Storage Analytics logs for specific requests. Filter based on the authentication mechanism, the success of the operation, or the resource that was accessed.
  + **Storage account keys**:In Azure Storage accounts, shared keys are called storage account keys. Azure creates two of these keys (primary and secondary) for each storage account you create. The keys give access to everything in the account.
  + **Shared access signatures**: As a best practice, you shouldn't share storage account keys with external third-party applications. If these apps need access to your data, you'll need to secure their connections without using storage account keys. For untrusted clients, use a shared access signature (SAS). A SAS is a string that contains a security token that can be attached to a URI. Use a SAS to delegate access to storage objects and specify constraints, such as the permissions and the time range of access
  + **Control network access**: By default, storage accounts accept connections from clients on any network. To limit access to selected networks, you must first change the default action. You can restrict access to specific IP addresses, ranges, or virtual networks.
  + **Advanced Threat Protection**: Microsoft Defender for Storage provides an extra layer of security intelligence that detects unusual and potentially harmful attempts to access or exploit storage accounts. This layer of protection allows you to address threats without being a security expert or managing security monitoring systems.Security alerts are triggered when anomalies in activity occur. These security alerts are integrated with Microsoft Defender for Cloud, and are also sent via email to subscription administrators, with details of suspicious activity and recommendations on how to investigate and remediate threats.
  + Along with role-based access control (RBAC), Azure Data Lake Storage Gen2 provides access control lists (ACLs) that are **POSIX-compliant**, and that restrict access to only authorized users, groups, or service principals. It applies restrictions in a way that's flexible, fine-grained, and manageable.Azure Data Lake Storage Gen2 authenticates through Azure Active Directory **OAuth 2.0 bearer tokens**.More significantly, these authentication schemes are integrated into the main analytics services that use the data. These services include Azure Databricks, HDInsight, and Azure Synapse Analytics. Management tools, such as Azure Storage Explorer, are also included.

1. **Blobs**:

* **Block blobs**: Block blobs are used to hold text or binary files up to ~5 TB (50,000 blocks of 100 MB) in size. The primary use case for block blobs is the storage of files that are read from beginning to end, such as media files or image files for websites. They are named block blobs because files larger than 100 MB must be uploaded as small blocks. These blocks are then consolidated (or committed) into the final blob.
* **Page blobs**: Page blobs are used to hold random-access files up to 8 TB in size. Page blobs are used primarily as the backing storage for the VHDs used to provide durable disks for Azure Virtual Machines (Azure VMs). They are named page blobs because they provide random read/write access to 512-byte pages.
* **Append blobs**: Append blobs are made up of blocks like block blobs, but they are optimized for append operations. These blobs are frequently used for logging information from one or more sources into the same blob. For example, you might write all of your trace logging to the same append blob for an application running on multiple VMs. A single append blob can be up to 195 GB.
* Blobs aren't efficient for structured data that needs to be queried frequently. They have higher latency than memory and local disks, and don't have the indexing features that make databases efficient at running queries. However, blobs are frequently used in combination with databases to store non-queryable data.For example, an app with a database of user profiles could store profile pictures in blobs. Each user record in the database would include the name or URL of the blob containing the user's picture.
* Blobs and containers support metadata in the form of name-value string pairs. Blob Storage does not provide any mechanism for searching or sorting blobs by metadata.
* The Blob Storage API is REST-based and supported by client libraries in many popular languages. It lets you write apps that create and delete blobs and containers, upload and download blob data, and list the blobs in a container.
* Apps using blobs as part of a storage scheme that includes a database often don't need to rely heavily on organization, naming, or metadata to indicate anything about their data. Such apps commonly use identifiers like GUIDs as blob names and reference these identifiers in database records.Other apps may use Azure Blob Storage more like a personal file system, where container and blob names are used to indicate meaning and structure.
* Public downloading supports many use cases, such as hosting static website assets and sharing files. This works because downloading blob contents works the same way as reading any other kind of data over the web: you just point a browser or anything that can make a GET request at the blob URL.Never put blob data in a public container that you don't intend to share publicly.
* containers are "flat" and don't support any kind of nesting or hierarchy. But if you give your blobs hierarchical names that look like file paths (such as finance/budgets/2017/q1.xls), the API's listing operation can filter results to specific prefixes. This enables you to navigate the list as if it was a hierarchical system of files and folders.This feature is often called virtual directories, because some tools and client libraries use it to visualize and navigate Blob Storage as if it was a file system
* Moving data to and from a blob is a network operation that takes time. The Azure Storage SDK for .NET provides asynchronous implementation of all methods that require network activity. We recommend using these async implementations whenever possible in your application.We also recommend using streams instead of in-memory structures like byte arrays or strings when you're working with large data objects. This method avoids buffering the full content in memory before sending it to the target. ASP.NET Core supports reading and writing streams from requests and responses.

1. **Files**:

* Azure File storage enables you to set up highly available network file shares that can be accessed using the standard Server Message Block (SMB) protocol. This means that multiple VMs can share the same files with both read and write access.
* You can also read the files using the REST interface or the storage client libraries
* File shares can be used for many common scenarios:
  + Storing shared configuration files for VMs, tools, or utilities so that everyone is using the same version.
  + Log files such as diagnostics, metrics, and crash dumps.
  + Shared data between on-premises applications and Azure VMs to allow migration of apps to the cloud over a period of time.

1. **Queues**:

* Azure Queue Storage is used to store and retrieve messages. Queue messages can be up to 64 KB in size, and a queue can contain millions of messages. Queues are used to store lists of messages to be processed asynchronously.
* You can use queues to loosely connect different parts of your application together. For example, we could perform image processing on the photos uploaded by our users. Perhaps we want to provide some sort of face detection or tagging capability, so people can search through all the images they have stored in our service. We could use queues to pass messages to our image-processing service to let it know that new images have been uploaded and are ready for processing. This sort of architecture would allow you to develop and update each part of the service independently.

1. [Table Storage](https://docs.microsoft.com/en-us/azure/storage/tables/table-storage-overview):
2. **Orchestration**: To use an analogy, think about a symphony orchestra. The central member of the orchestra is the conductor. The conductor does not play the instruments, they simply lead the symphony members through the entire piece of music that they perform. The musicians use their own skills to produce particular sounds at various stages of the symphony, so they may only learn certain parts of the music. The conductor orchestrates the entire piece of music, and therefore is aware of the entire score that is being performed. They will also use specific arm movements that provide instructions to the musicians how a piece of music should be played.

ADF can use a similar approach, whilst it has native functionality to ingest and transform data, sometimes it will instruct another service to perform the actual work required on its behalf, such as a Databricks to execute a transformation query. So, in this case, it would be Databricks that performs the work, not ADF. ADF merely orchestrates the execution of the query, and then provides the pipelines to move the data onto the next step or destination

1. **Azure Data Factory**: The need to trigger the batch movement of data, or to set up a regular schedule is a requirement for most analytics solutions. Azure Data Factory (ADF) is the service that can be used to fulfill such a requirement. ADF provides a cloud-based data integration service that orchestrates the movement and transformation of data between various data stores and compute resources.

* You can build complex ETL processes that transform data visually with data flows or by using compute services such as Azure HDInsight Hadoop, Azure Databricks, and Azure Synapse Analytics.
* It also provides rich visualizations to display the lineage and dependencies between your data pipelines, and monitor all your data pipelines from a single unified view to easily pinpoint issues and setup monitoring alerts.
* Azure Data Factory provides nearly 100 enterprise connectors and robust resources for both code-free and code-based users to accomplish their data movement and transformation needs.
* The stored data is used for historical and trend analysis reporting.
* The data warehouse acts as a central repository for many subject areas and contains the "single source of truth."
* You can perform advanced analytics in the form of predictive or preemptive analytics using a range of Azure data platform services. Azure Data Factory provides the integration from source systems into a Data Lake store, and can initiate compute resources such as Azure Databricks, or HDInsight to use the data to perform the advanced analytical work
* **Publish**: After the raw data has been refined into a business-ready consumable form from the transform and enrich phase, you can load the data into **Azure Data Warehouse**, **Azure SQL Database, Azure Cosmos DB**, or whichever analytics engine your business users can point to from their business intelligence tools
* **Monitor**: Azure Data Factory has built-in support for pipeline monitoring via **AzureMonitor**, **API, PowerShell, Azure Monitor logs, and health panels on the Azure portal**, to monitor the scheduled activities and pipelines for success and failure rates.
* **Linked Service**:Data Factory supports a wide variety of data sources that you can connect to through the creation of an object known as a Linked Service, which enables you to ingest the data from a data source in readiness to prepare the data for transformation and/or analysis. In addition, Linked Services can fire up compute services on demand. For example, you may have a requirement to start an on-demand HDInsight cluster for the purpose of just processing data through a Hive query. So Linked Services enables you to define data sources, or compute resource that is required to ingest and prepare data.There are over 100 connectors that can be used to define a linked service.The list of connectors is constantly evolving. You can keep up to date with the latest list, and the activity support by looking at the [connectors overview page](https://docs.microsoft.com/en-us/azure/data-factory/connector-overview)

It's recommended to use Azure Key Vault to store any connection strings or passwords or managed identity authentication for Data Factory Linked Services. For security reasons, data factory **doesn't store secrets in Git**. Any changes to Linked Services containing secrets such as passwords are **published immediately** to the Azure Data Factory service.Using Key Vault or MSI authentication also makes continuous integration and deployment easier as you won't have to provide these secrets during Resource Manager template deployment.

If you choose to author the linked service through the user interface, Data Factory can provide you with **built-in parameterization** for some of the connectors:

Amazon Redshift

Azure Cosmos DB (SQL API)

Azure Database for MySQL

Azure SQL Database

Azure Synapse Analytics (formerly SQL DW)

MySQL

Oracle

SQL Server

Generic HTTP

Generic REST

* **Dataset**is a named view of data that simply points or references the data you want to use in your activities as inputs and outputs. It represents data structures within the data store that is being referenced by the Linked Service object. Datasets can also be used by an ADF object known as an Activity
* **Pipeline:** A pipeline in Azure Data Factory represents a logical grouping of activities where the activities together perform a certain task.
* **Activities:** Activities in a pipeline are referred to as actions that you perform on your data. An activity can take zero or more input datasets and produce one or more output datasets.They typically contain the transformation logic or the analysis commands of the Azure Data Factory’s work.

There are three categories including:

* + **Data movement activities**: Eg. Copy Activity
  + **Data transformation activities**: Eg.Mapping Data Flow, Azure Databricks, Azure Batch, SQL Database and Azure Synapse Analytics, Machine Learning Services, Azure Virtual machines and HDInsight. Existing SQL Server Integration Services (SSIS) Packages. As this list is always evolving, you can get the [latest information here](https://docs.microsoft.com/en-us/azure/data-factory/concepts-pipelines-activities)
  + **Control activities**: You can use the control flow within the design to orchestrate pipeline activities. Eg. Execute Pipeline Activity, ForEachActivity,WebActivity, Lookup Activity, Get Metadata Activity, Until Activity, If Condition Activity, Wait Activity

Activities includes:

* + the **Copy Activity** that can be used to ingest data from a variety of data sources.
  + It can also include the **Mapping Data Flow** to perform **code-free data transformations**.
  + It can also include the execution of a stored procedure, Hive Query, or Pig script to transform the data
  + You can push data into a Machine Learning model to perform analysis.
  + It is common for multiple activities to take place that may include transforming data using a SQL stored procedure and then perform analytics with Databricks
  + In this case, multiple activities can be logically grouped together with an object referred to as a **Pipeline**, and these can be scheduled to execute.
* **Debug data factory pipelines**:By authoring a pipeline using the pipeline canvas, you can test your activities and pipelines by using the Debug capability.
* **Trigger** can be defined that determines when a pipeline execution needs to be kicked off. There are different types of triggers for different types of events.
* **Control flow** is an orchestration of pipeline activities that includes chaining activities in a sequence, branching, defining parameters at the pipeline level, and passing arguments while invoking the pipeline on-demand or from a trigger. It also includes custom-state passing and looping containers, and For-each iterator.
* **Delta flows**: Use-cases related to using delta flows are delta loads. Delta loads in ETL patterns will only load **data that has changed since a previous iteration of a pipeline**. Capabilities such as lookup activity,LookUp transformation,Exist transformation, and flexible scheduling helps handling delta load jobs. In the case of using a Lookup activity, it will read or look up a record or table name value from any external source. This output can further be referenced by succeeding activities.
* **Parameters** are key-value pairs of read-only configuration.  Parameters are defined in the pipeline. The arguments for the defined parameters are passed during execution from the run context that was created by a trigger or a pipeline that was executed manually. Activities within the pipeline consume the parameter values.
* **Global Parameters**: Setting global parameters in an Azure Data Factory pipeline allows you to use these constants for consumption in pipeline expressions. A use-case for setting global parameters is when you have multiple pipelines where the parameters names and values are identical. The command or reference of global parameters in Azure Data Factory flows as follows: pipeline().globalParameters..

If you use the continuous integration and continuous (CI/CD) deployment process with Azure Data Factory, the global parameters can be overridden, if you want, for each environment that you have created. You have two ways in order to do so:

* + Include global parameters in the Azure Resource Manager template
  + Deploy global parameters via a PowerShell script
* **Integration runtime:**It enables to bridge between the activity and linked Services objects. It is referenced by the linked service and provides the compute environment where the activity either runs on or gets dispatched from. This way, the activity can be performed in the region closest possible. In short, the Integration Runtime (IR) is the compute infrastructure used by Azure Data Factory. There are three types of Integration Runtime, including **Azure, Self-hosted, and Azure-SSIS**.You can use Self-hosted Integration Runtime (if your **data store is located in a private network**). If not specified, it uses the default Azure Integration Runtime.It provides the following data integration capabilities across different network environments, including:
  + **Data Flow**: Execute a Data Flow in managed Azure compute environment.
  + **Data movement**: Copy data across data stores in public network and data stores in private network (on-premises or virtual private network). It provides support for built-in connectors, format conversion, column mapping, and performant and scalable data transfer.
  + **Activity dispatch**: Dispatch and monitor transformation activities running on a variety of compute services such as Azure Databricks, Azure HDInsight, Azure Machine Learning, Azure SQL Database, SQL Server, and more.
  + **SSIS package execution**: Natively execute SQL Server Integration Services (SSIS) packages in a managed Azure compute environment.The location of your Azure-SSIS IR does not need to be the same as the location of your data factory, but it should be the same as the location of your own Azure SQL Database or Azure SQL Database managed instance server where SSISDB is to be hosted. This way, your Azure-SSIS Integration Runtime can easily access SSISDB without incurring excessive traffics between different locations. When you have on-premises data sources/destinations, you should create a new Azure SQL Database or Azure SQL Database managed instance server in the same location of a virtual network connected to your on-premises network. If the location of your existing Azure SQL Database or Azure SQL Database managed instance server where SSISDB is hosted is not the same as the location of a virtual network connected to your on-premises network, first create your Azure-SSIS IR using an existing Azure SQL Database or Azure SQL Database managed instance server and joining another virtual network in the same location, and then configure a virtual network to virtual network connection between different locations.

If you choose to use the auto-resolve Azure IR which is the default, ADF will make a best effort to automatically detect your sink and source data store to choose the best location either in the same region if available or the closest one in the same geography for the Copy Activity. For anything else, it will use the IR in the Data Factory region. Azure Integration Runtime also has support for virtual networks.

[Set-AzDataFactoryV2IntegrationRuntime -DataFactoryName "SampleV2DataFactory1" -Name "MySampleAzureIR" -ResourceGroupName "ADFV2SampleRG" -**Type Managed** -Location "West Europe"]

When Azure IR, the type must be set to **Managed**. You do not need to specify compute details because it is **fully managed elastically in cloud**

* Once all the work is complete, you can then use Data Factory to publish the final dataset to another linked service that can then be consumed by technologies such as Power BI or Machine Learning.
* **Security**: To create Data Factory instances, the user account that you use to sign in to Azure must be a member of the **contributor** or **owner** role, or an **administrator** of the Azure **subscription**
  + To create and manage child **resources in the Azure portal**, you must belong to the **Data Factory Contributor role** at the **resource group level** or above.
  + To create and manage **resources with PowerShell** or the **SDK**, the **contributor** role at the **resource level or above is sufficient**.

**Data Factory Contributor role**: When you are added as a member of this role, you have the following permissions:

* + Create, edit, and delete data factories and child resources including datasets, linked services, pipelines, triggers, and integration runtimes.
  + Deploy Resource Manager templates. Resource Manager deployment is the deployment method used by Data Factory in the Azure portal.
  + Manage App Insights alerts for a data factory.
  + At the resource group level or above, lets users deploy Resource Manager template.
  + Create support tickets.

If the Data Factory Contributor role does not meet your requirement, you can create your own [custom role](https://docs.microsoft.com/en-us/azure/role-based-access-control/custom-roles).

[More on Security and Logging](https://docs.microsoft.com/en-us/learn/modules/petabyte-scale-ingestion-azure-data-factory/7-understand-data-ingestion-security-considerations): Check your organizations standards for logging and monitoring and snap to the standard including:

* + **Audit logging**: You can use Azure Data Factory diagnostic settings to configure **diagnostic logs** to track pipeline-run data, which is retained for **45 days**. You can save the diagnostic logs to Azure Storage accounts for future analysis.
  + Security logs
  + Anti-malware logging
  + Log retention policies
* **Ingesting data using SSIS packages**: Many organizations have decades of development investment in SQL Server Integration Services (SSIS) packages that contain both ingestion and transformation logic from on-premises and cloud data stores. Azure Data Factory provides the ability to lift and shift existing SSIS workload, by creating an Azure-SSIS Integration Runtime to natively execute SSIS packages, and will enable you to deploy and manage your existing SSIS packages with little to no change using familiar tools such as **SQL Server Data Tools (SSDT)** and **SQL Server Management Studio (SSMS)**, just like using SSIS on-premises. You can use the **Data Migration Assistant (DMA)** to perform an assessment of the SSIS packages that exist and identify any compatibility issues with them

A SSIS solution usually consists of one or more SSIS projects, each containing one or more SSIS packages.

1. **SSIS projects:**

From SQL Server 2012, a project is the unit of deployment for SSIS solutions. You can define project-level parameters to enable users to specify run-time settings, and project-level connection managers that reference data sources and destinations used in package data flows. You can then deploy projects to an SSIS catalog in a SQL Server instance, and configure project-level parameter values and connections as appropriate for execution environments.

1. **SSIS packages:**

A project contains one or more packages, each defining a workflow of tasks to be executed. **The workflow of tasks in a package is referred to as its control flow**. A package control flow can include one or more Data Flow task, each of which encapsulates its own data flow pipeline. Packages can include package-level parameters so that dynamic values can be passed to the package at run time.

An Azure-SSIS IR supports:

* + Running packages that are or to be deployed into SSIS catalog (**SSISDB**) hosted by your Azure SQL Database server or Managed Instance in **ProjectDeployment Model**
  + Running packages that are or to be deployed into file system, **Azure Files**, or SQL Server database (**MSDB**) hosted by your Azure SQL Managed Instance in **PackageDeployment Model**

**Deployment settings page:**

* + Click the Create SSIS catalog (SSISDB) hosted by Azure SQL Database server/Managed Instance to store your projects/packages/environments/execution logs check box to choose the package deployment mode
  + The alternative approach is to Select the Create package stores to manage your packages that are deployed into file system/Azure Files/SQL Server database (MSDB) hosted by Azure SQL Managed Instance check box to choose whether you want to manage your packages that are deployed into MSDB, file system, or Azure Files (Package Deployment Model) with Azure-SSIS IR package stores

After the Azure-SSIS IR is provisioned, the same **familiar tools for deployment and running** the packages in Azure can be used. Most of the familiar tools such as SQL Server Data Tools (SSDT), SQL Server Management Studio (SSMS), Azure Data Studio, and command-line utilities are Azure-enabled, and therefore ready to be used.

Using **SSDT allows you to check and assess the Azure Cloud compatibility** with Azure-SSIS Integration Runtime in Azure Data Factory of the SSIS packages you might already be running locally. This feature comes in handy when you want to test existing packages before an actual lift and shift or migration can take place to Azure. If you want to develop **new** packages to run in Azure, it's also good to test them with this feature

* **Mapping Data Flows** provide an environment for building a wide range of data transformations visually without the need to use code.The resulting data flows that are created are subsequently executed on scaled-out Apache Spark clusters that are automatically provisioned when you execute the Mapping Data Flow. Mapping Data Flows also provides the capability to monitor the execution of the transformations so that you can view how the transformations are progressing, or to understand any errors that may occur.

Mapping Data Flow follows an extract, load, transform (ELT) approach and works with staging datasets that are all in Azure.Azure Data Factory has access to over 80 native connectors. To include data from those other sources in your data flow, use the Copy Activity to load that data into one of the supported staging areas.

Mapping Data Flows provides a number of different transformation types that enable you to modify data. They are broken down into the following categories:Schema modifier, Row modifier, Multiple inputs/outputs. Some important ones are explained below:

* + **Alter row**: Set insert, delete, update, and upsert policies on rows. You can add one-to-many conditions as expressions. These conditions should be specified in order of priority, as each row will be marked with the policy corresponding to the first-matching expression. Each of those conditions can result in a row (or rows) being inserted, updated, deleted, or upserted. Alter Row can produce both DDL & DML actions against your database.
  + **Flatten**: Take array values inside hierarchical structures such as JSON and unroll them into individual rows.
  + **Join**:Combine data from two sources or streams.
  + **Union**: Combine multiple data streams vertically.
  + **Select**: Rename column names and stream names,drop or reorder columns.
  + **Pivot**: An aggregation where one or more grouping columns has distinct row values transformed into individual columns.
  + **Unpivot**: Pivot columns into row values.
  + **Surrogate key**: Add an incrementing non-business arbitrary key value.
  + **Window**: Define window-based aggregations of columns in your data streams.It’s like sorting data between the groups formed by different filters and derive new column based on each group. Say you are interested in how a movie ranks within its year for its specific genre. So, we will window over PrimaryGenre and year with an unbounded range, sort by Rotten Tomato descending, and calculate a new column called RatingsRank that is equal to the rank [rank()] each movie has within its specific genre-year. Window is like Aggregate transformation without decreasing number of rows.
  + **Aggregate**: Define different types of aggregations such as SUM, MIN, MAX, and COUNT grouped by existing or computed columns. For converting row values in column values. Example: Average Rotten Tomatoes rating, the highest and lowest rated movie (utilizing the windowing function sorted by ratings) and the number of movies that are in each group. All 4 columns are derived from single column row values.

**Slowly changing dimensions (SCD)** are tables in a dimensional model that handle changes to dimension values over time. Learning the best practices to design and load slowly changing dimensions will help you successfully handle changes in your data.A slowly changing dimension (SCD) is one that appropriately manages the change of dimension members over time. It applies when the values of a business entity change over time, and not on a set schedule. A good example of an SCD is a customer dimensiontable (as part of a star schema model for tracking sales transactions), specifically its contact detail columns like email address and phone number.

Star schema design theory refers to common **SCD types**. The most common are **Type 1** and **Type 2**. In practice a dimension table may support a combination of history tracking methods, including **Type 3** and **Type 6**.

**Data Flow Expression Builder**: Some of the transformations that you can define have an Data Flow Expression Builder that will enable you to customize the functionality of a transformation using columns, fields, variables, parameters, functions from your data flow in these boxes. E.g.

* + The **iifNull** function replaces null values with empty strings.
  + The hash values [sha2(256,….)] tend to duplicate when null entries are present.
  + InsertedDate = iif(isNull(InsertedDate), currentTimestamp(), {InsertedDate})

A **Data Preview** tab is available in Debug mode that will allow you to view the data at each stage of the pipeline. You can view the data after each transformation. The data previewer also provides the ability to actions on the data such as looking at descriptive **statistics** of the data, or the ability to **modify** the data.Finally, you can use the debug settings to control the **number of rows** that are returned within the data previewer.

* **Transforming data using compute resources**: Azure Data Factory can also call on compute resources to transform data by a data platform service that may be better suited to the job. Example: Azure Databricks, HDInsight,SSIS, Azure Synapse Spark pools (python) or SQL Pools (stored procedure using Transact-SQL), Azure Function, etc.
* **SDK**:While most Azure Data Factory users develop using the user interface, Azure Data Factory is available in a variety of **software development kits (SDKs)** for anyone who wish to develop programmatically. When using an SDK, a user works directly against the Azure Data Factory service and all updates are immediately applied to the factory. You can programmatically interact with Azure Data Factory with following languages and SDKs:
  + Python
  + .NET
  + REST APIs
  + PowerShell
  + Azure Resource Manager Templates
  + Data flow scripts

**Data flow script (DFS)** is the underlying metadata, similar to a coding language, that is used to execute the transformations that are included in a mapping data flow. Every transformation is represented by a series of properties that provide the necessary information to run the job properly. The script is visible and editable from ADF by clicking on the "script" button on the top ribbon of the browser UI

* **Continuous integration and Delivery**:Continuous integration is the practice of testing each change made to your codebase automatically and as early as possible. Continuous delivery follows the testing that happens during continuous integration and pushes changes to a staging or production system.In Azure Data Factory, continuous integration and delivery (CI/CD) means moving Data Factory pipelines from one environment (development, test, production) to another.

Only the development factory is associated with a git repository. The test and production factories shouldn't have a git repository associated with them and should only be updated via an Azure DevOps pipeline or via a Resource Management template.

In CI/CD scenarios, the integration runtime (IR) type in different environments must be the same. For example, if you have a self-hosted IR in the development environment, the same IR must also be of type self-hosted in other environments, such as test and production. Similarly, if you're sharing integration runtimes across multiple stages, you have to configure the integration runtimes as linked self-hosted in all environments, such as development, test, and production.

Stopping active triggers before deployment and restarting them after deployment are already included in the pre and post deployment scripts provided by the Azure Data Factory team.

If you are unable to use Azure DevOps or a different release management tool, you can manually promote a data factory using an ARM Template.

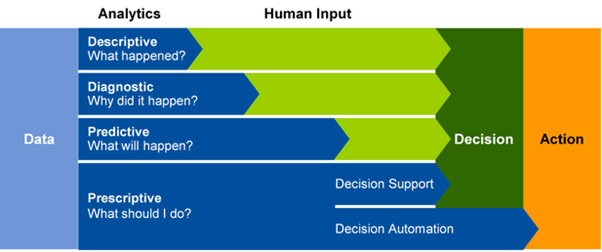
1. In the ARM Template list, select Export ARM Template to export the Resource Manager template for your data factory in the development environment.
2. In your test and production data factories, select Import ARM Template. This action takes you to the Azure portal, where you can import the exported template. Select Build your own template in the editor to open the Resource Manager template editor.

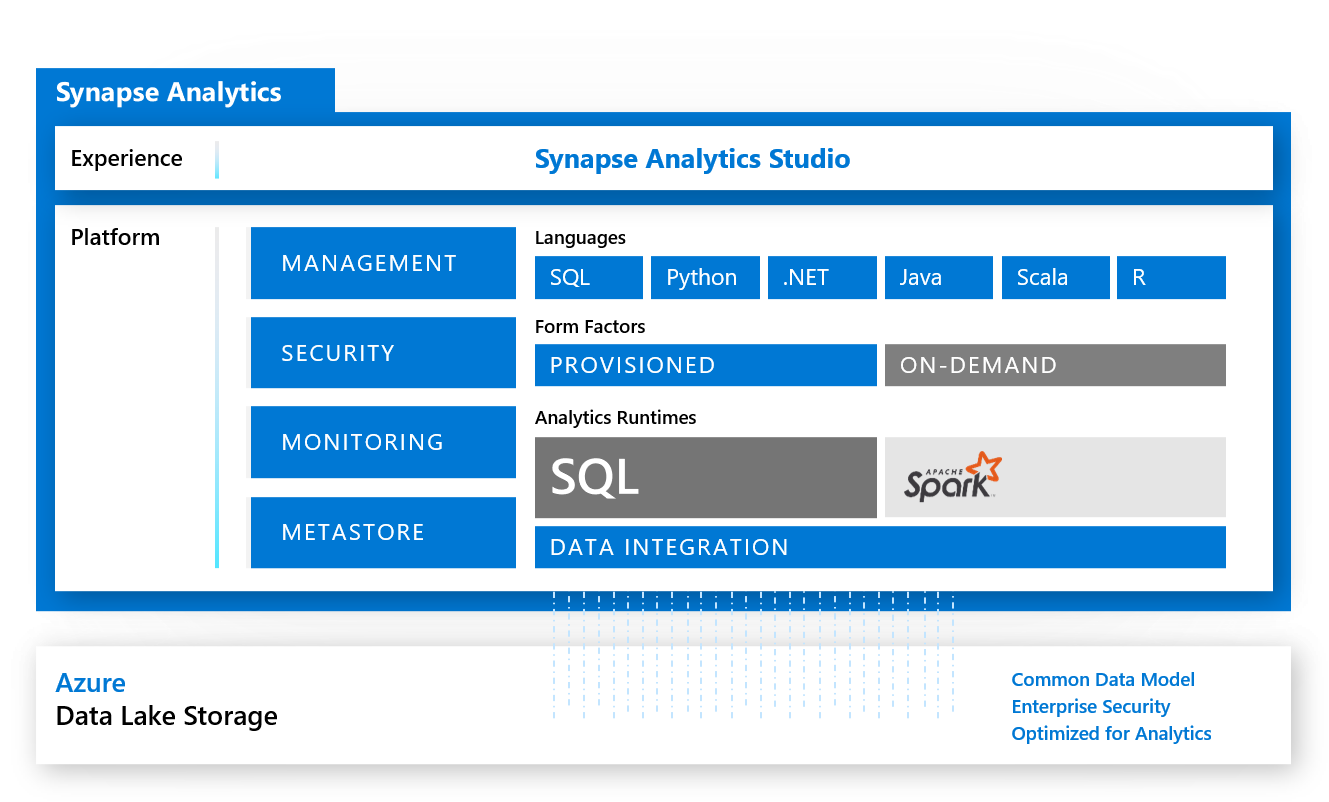
A **custom parameterization template** doesn't change the ARM template parameter limit of 256. It lets you choose and decrease the number of parameterized properties.

If you've set up CI/CD for your data factories, you might exceed the Azure Resource Manager template limits as your factory grows bigger. For example, one limit is the maximum number of resources in a Resource Manager template. To accommodate large factories while generating the full Resource Manager template for a factory, Data Factory now generates linked Resource Manager templates. With this feature, the entire factory payload is broken down into several files so that you aren't constrained by the limits. The linked templates are generated and saved alongside the full Resource Manager templates in the adf\_publish branch in a new folder called **linkedTemplates.** The parent template is called ArmTemplate\_master.json, and child templates are named with the pattern ArmTemplate\_0.json, ArmTemplate\_1.json, and so on.

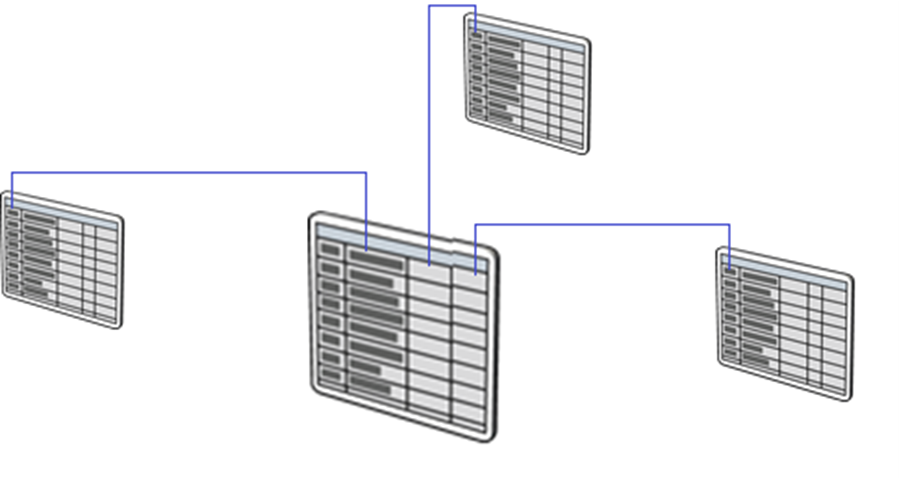
* [**Rerun Azure Data Factory pipelines**](https://docs.microsoft.com/en-us/learn/modules/operationalize-azure-data-factory-pipelines/7-rerun-azure-data-factory-pipelines)**:**

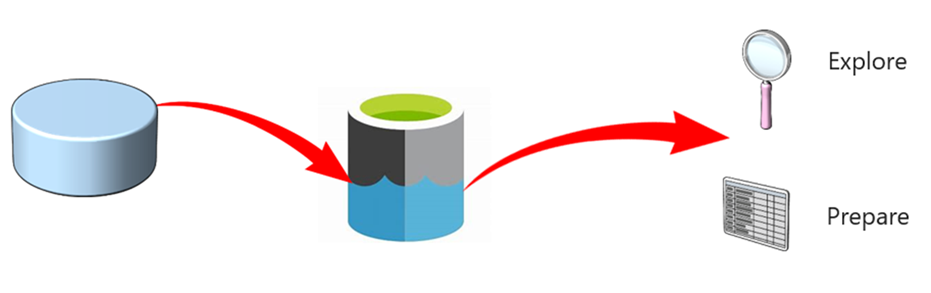
1. To rerun a pipeline that has previously ran from the start, hover over the specific pipeline run and select Rerun.
2. If you select multiple pipelines, you can use the Rerun button to **run them all**.
3. If you wish to rerun starting at a specific point, you can do so from the activity runs view. Select the activity you wish to start from and select **Rerun from activity**.
4. If an activity fails, times out, or is canceled, you can rerun the pipeline from that failed activity by selecting **Rerun from failed activity**.
5. You can view the rerun history for all the pipeline runs in the list view
6. You can also view **rerun history for a particular pipeline** run.
7. **Azure Synapse Analytics**: Azure Synapse Analytics is an integrated analytics platform, which combines data warehousing, big data analytics, data integration, and visualization into a single environment.Much of the functionality of Azure Data Factory appears in Azure Synapse Analytics as a feature referred to as Pipelines, which enables you to integrate data pipelines between SQL Pools, Spark Pools and SQL Serverless, providing a one stop shop for all your analytical needs.[Descriptive analytics (serverless SQL pool), Diagnostic analytics (serverless SQL pool),Predictive analytics (Spark pools) Prescriptive analytics (Apache Spark, Azure Synapse Link)]





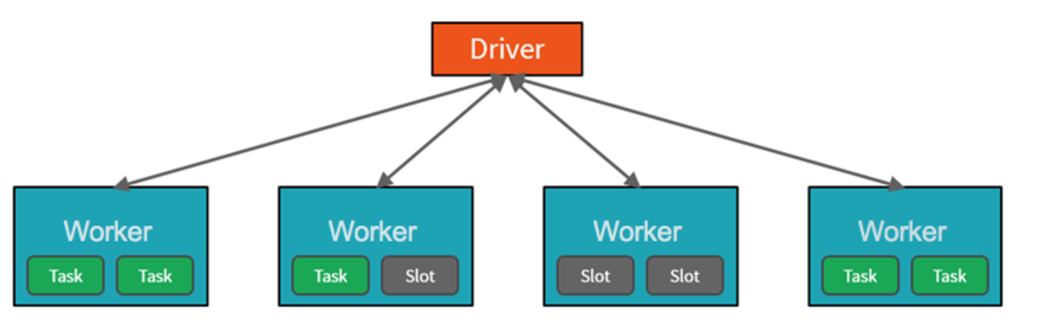
* **SQL Pool:** Azure Synapse SQL is a distributed query system that enables you to implement data warehousing and data virtualization scenarios using standard T-SQL experiences familiar to data engineers. Synapse SQL offers both serverless and dedicated resource models to work with both descriptive and diagnostic analytical scenarios. For predictable performance and cost, create dedicated SQL pools to reserve processing power for data stored in SQL tables. For unplanned or ad-hoc workloads, use the always-available, serverless SQL endpoint.
* **SQL Pool as**A **data warehouse** is a core component of Business Intelligence (BI) solutions that provides a central repository of data **stored in relational tables**.It facilitates solutions around descriptive analytics. The data is retrieved, cleansed, and transformed from a range of source data system, and is then served in a structured relational format commonly referred to as a **star schema**.Data in a data warehouse is stored in permanent tables that are populated using an extract, transform, and load (ETL) process by services such as Azure Synapse pipelines, or Azure Data Factory.To create a relational data warehouse in Azure Synapse Analytics, you must create a **dedicated SQL Pool**.



* **SQL Pool for Data virtualization** allows you to interact with data without the need to understand how the data is formatted, structured, or what is its data type. It enables you to explore the data without understanding the technical specifications of the source data, which can be very helpful when performing diagnostic analytics where the need to access data in a timely manner to answer a question is more important. Data virtualization also enables ad hoc data preparation scenarios, where organizations are wanting to unlock insights from their own data stores without going through the formal processes of setting up a data warehouse. You can extract data from a source system in a raw format and load it into a data lake. From here, transformations may be applied to present the data as required. As the most complex part of the extract, load, and transform (ELT) process is at the end, it means that the access to the data is much quicker.

To meet the delivery of these types of solutions, Azure Synapse SQL offers both a dedicated and serverless model of the service to meet the different demands of both solutions.

* **Apache Spark** is aopen-source distributed parallel processing framework that supports in-memory processing to boost the performance of big-data analytic applications. Apache Spark in Azure Synapse Analytics is one of Microsoft's implementations of Apache Spark in the cloud.Spark pool clusters are groups of computers that are treated as a single computer and handle the execution of commands issued from notebooks.It consists of a Spark Driver and Worker nodes. The Driver node sends work to the Worker nodes and instructs them to pull data from a specified data source. Moreover, you can configure the number of nodes that are required to perform the task.



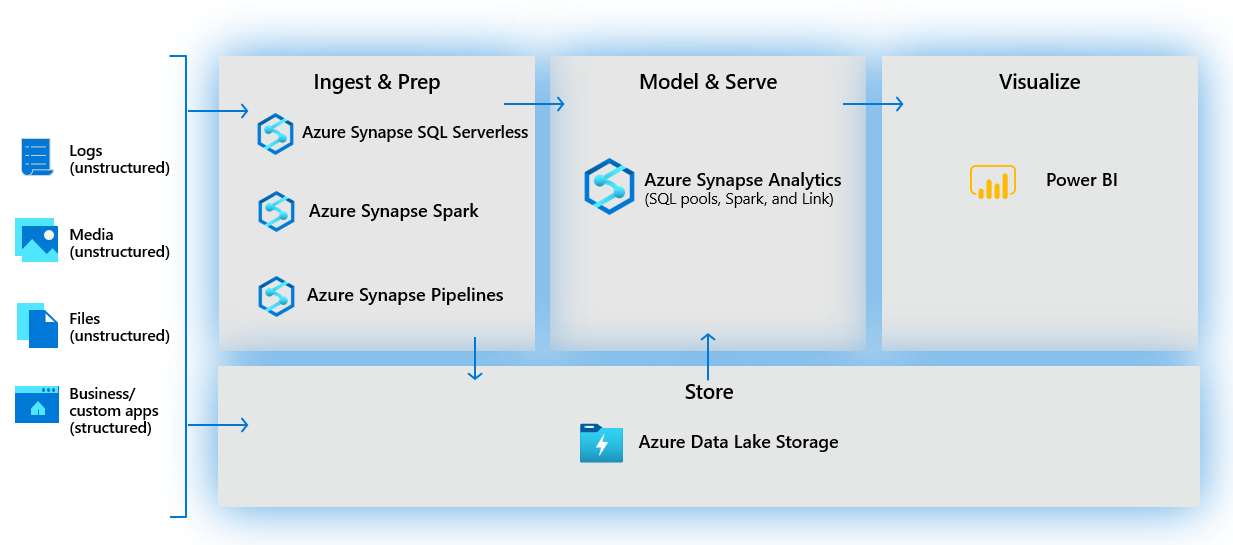
Spark pools in Azure Synapse Analytics offer a fully managed Spark service. The benefits of creating a Spark pool in Synapse Analytics include: Speed and efficiency, Ease of creation, Ease of use, Scalability, Support for Azure Data Lake as well as BLOB storage.

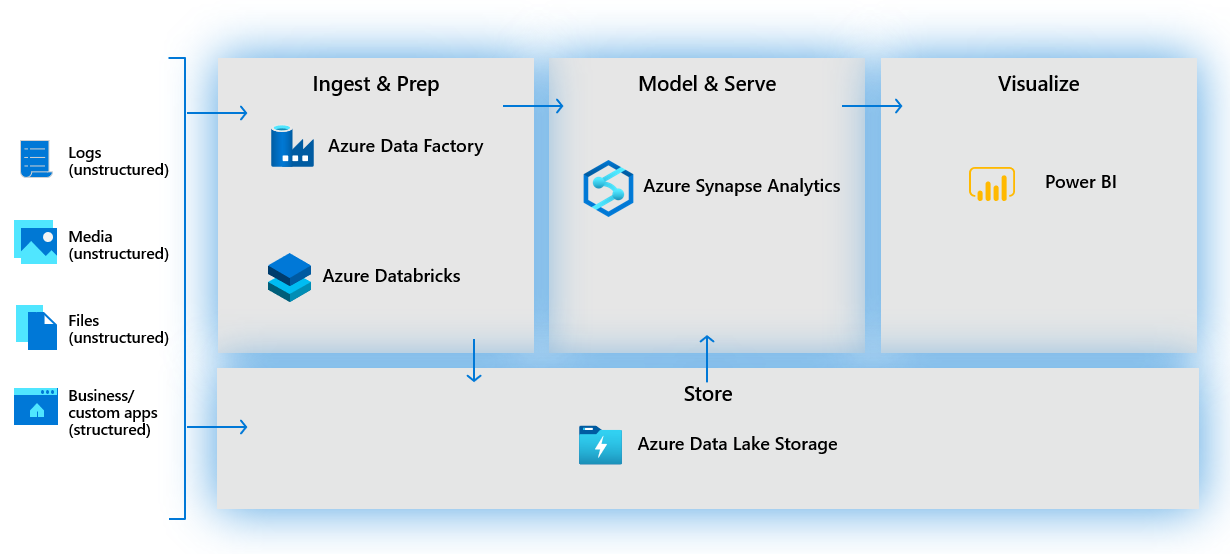
The primary use case for Apache Spark for Azure Synapse Analytics is to process big data workloads that cannot be handled by Azure Synapse SQL, and where you don’t have an existing Apache Spark implementation. Perhaps you must perform a complex calculation on large volumes of data. Handling this requirement in Spark pools will be far more efficient than in Synapse SQL. You can pass the data through to the Spark cluster to perform the calculation, and then pass the processed data back into the data warehouse, or back to the data lake.

If you already have a Spark implementation in place already, Azure Synapse Analytics can also integrate with other Spark implementations such as **Azure Databricks**, so you don’t have to use the feature in Azure Synapse Analytics if you already have a Spark setup already.

Finally, Spark pools in Azure Synapse Analytics come with Anaconda libraries pre-installed. Anaconda provides close to 200 libraries that enables you to use the spark pool to perform machine learning, data analysis, and data visualization. This can enable data scientists and data analysts to interact with the data using the Spark pool too.

* **Azure Synapse Studio** is a single web UI that allows you to:
  + Explore your data estate.
  + Develop TSQL scripts and notebooks to interact with the analytical engines.
  + Build data integration pipelines for managing data movement.
  + Monitor the workloads within the service.
  + Manage the components of the service.
* On the initial deployment of Azure Synapse Analytics, there are a few resources that deploy along with it, including the Azure Synapse Workspace and an Azure Data Lake Storage Gen2 (ADLS Gen2) account that acts as the primary storage for the workspace.
* **When to use Azure Synapse Analytics**: For,
  + Modern data warehousing
  + Advanced analytics
  + Data exploration and discovery
  + Real time analytics
  + Data integration





* + Integrated analytics: With the variety of analytics that can be performed on the data at your disposal, putting together the services in a cohesive solution can be a complex operation. Azure Synapse Analytics removes this complexity by integrating the analytics landscape into one service. That way you can spend more time working with the data to bring business benefit, than spending much of your time provisioning and maintaining multiple systems to achieve the same outcomes.
* Azure Synapse organizes itself into **hubs**:

1. **Home**: The home hub contains short cuts that enable you to ingest, explore, analyze, and visualize your data. These provide shortcuts to tools such as the Copy Data Tool for ingesting data, to connecting to a Power BI workspace for visualization. You will also find links to resources such as the documentation and pricing page. It will also list any resources you recently accessed, or pinned as favorite
2. **Data**: The data hub can be accessed by either clicking on the Explore link in the home hub, or by selecting data on the left of the application. In this hub, you can access your provisioned SQL pool databases and SQL serverless databases in your workspace, as well as external data sources, such as storage accounts and other linked services. You also can preview data tables and data files
3. **Develop**: The Develop hub is where you manage SQL scripts, Synapse notebooks, data flows, and Power BI reports. It can also be accessed by clicking on the Analyze icon in the home page.
4. **Integrate**: Manage data integration pipelines within the Integrate hub. If you are familiar with Azure Data Factory, then you will feel at home in this hub. It removes the need to use Azure Data Factory separately for data movement and transformation pipelines.
5. **Monitor**:Use the Monitor hub to view pipeline and trigger runs, view the status of the various integration runtimes that are running, view Apache Spark jobs, SQL requests, and data flow debug activities. If you want to see the status of a job or activity, this is where you want to go.
6. **Manage**:The Manage hub enables you to perform some of the same actions we saw in the Azure portal, such as managing SQL and Spark pools. However, there is a lot more you can do in this hub that you cannot do anywhere else, such as managing Linked Services and integration runtimes, and creating pipeline triggers.

* **Modern data warehouse**: The modern data warehouse serves to address following challenges:

1. **Increased volumes of data**: With Azure, services such as Azure Synapse Analytics can be scaled at the click of a button, and can even be auto-scaled.
2. **New varieties of data**: Staging data is also simplified using Azure Data Lake Store Gen2, which can store a wide variety of data in its raw format, making the process of ingesting data into a data warehouse much easier.For example, extrapolating data from sources such as PDF files through to sound files were either too complex or cost prohibitive. The improvements in AI technologies such as Form Recognizer and Speech to Text Cognitive Services means that these types of data sources can now be passed through a cognitive service and outputted in a text-based format that can be stored in the Azure Data Lake Store Gen2, along with the source files themselves.
3. **Data velocities**: Azure has made it easier and much more cost effective to provision streaming services that can interact with a wide variety of services so that a modern data warehouse can deliver solutions in a batch or a real-time manner without the obstruction of cost.

The process of building a modern data warehouse typically consists of:

* + Data Ingestion and Preparation.
  + Making the data ready for consumption by analytical tools.
  + Providing access to the data, in a shaped format so that it can easily be consumed by data visualization tools.

The modern data warehouse serves to address these challenges. A good data warehouse adds value, such as acting as a central location for all your data, scale with the data as it grows over time, and providing familiar tools and ecosystem for your data engineers, data analysts, data scientists, and developers.

1. **One place for all your data**: With a modern data warehouse, we have one hub for all data when using Synapse Analytics. Synapse Analytics enables you to ingest data from multiple data sources through its orchestration pipelines in Integrate hub.
2. **Unlimited data scale**: You can scale out or back compute by adjusting the number of Data Warehouse Units (DWUs) assigned to your SQL pool in Manage hub. This adjusts the loading and query performance linearly as you add more units.

To perform a scale operation, SQL pool first kills all incoming queries and then rolls back transactions to ensure a consistent state. Scaling only occurs once the transaction rollback is complete. You can scale SQL compute at any time by using this slider. You can also programmatically adjust the Data Warehouse Units, enabling scenarios where you automatically scale your pool based on a schedule or other factors.

For Spark pool, select the auto-scale settings button to change number of nodes. Autoscale load and scales the number of nodes up or down based on resource requirements. It does this by continuously monitoring pending CPU, pending memory, free CPU, free memory, and used memory per node metrics. It checks these metrics every 30 seconds and makes scaling decisions based on the values.It can take 1-5 minutes for a scaling operation to complete.

1. **Familiar tools and ecosystem**: Synapse provide tools such as SQL Script, Notebook, Data Flow and PowerBI in Develop hub. Synapse Studio provides several options in Data hub to work with files stored in attached storage accounts, such as creating a new SQL script, a notebook, data flow, or new dataset.Authentication and authorization with linked services, such as the primary data lake storage account, are fully integrated, allowing you to immediately start working with files without dealing with account credentials.

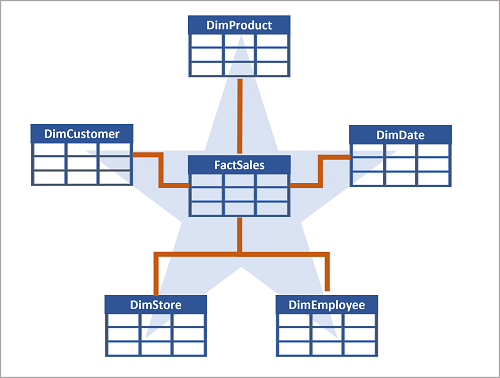
* **Latencies:** In data engineering, we tend to describe data loading velocity as one of three latencies:
  + Batch: Queries or programs that take tens of minutes, hours, or days to complete. Activities could include initial data wrangling, complete ETL pipeline, or preparation for downstream analytics.
  + Interactive query: Querying batch data at "human" interactive speeds, which with the current generation of technologies means results are ready in time frames measured in seconds to minutes.
  + Real-time / near real-time: Processing of a typically infinite stream of input data (stream), whose time until results ready is short—measured in milliseconds or seconds in the longest of cases.Streaming data sources can include IoT devices and sensors, financial transactions, web clickstream data, factories, and medical devices, to name a few.
* **File Formats for modern DWH**: When it comes to ingesting raw data in batch from new data sources, these data formats are natively supported by Synapse:
  + **Raw Data:**
    - **CSV:** Data from relational databases should typically be stored in CSV format.CSV files don't store file metadata, so readers need to either be supplied with the schema or the schema needs to be inferred. Supplying a schema is tedious and inferring a schema is error prone / expensive.
    - **JSON:** For data from web APIs and NoSQL databases, JSON is the recommended format.
  + **Refined versions for data:**
    - **Parquet:** There is industry alignment around the Parquet format for sharing data at the storage layer (for example, across Hadoop, Databricks, and SQL engine scenarios). Parquet is a high-performance, column-oriented format optimized for big data scenarios.The values are clustered by column so the compression is more efficient (to shrink the storage footprint), and a query engine can push down column projections (to reduce read I/O from network and disk by skipping unwanted columns), otherwise known as column pruning.Similar data types (for a column) are stored together in Parquet files, leading to efficient data compression and encoding schemes.Parquet stores the file schema in the file metadata.
    - **ORC:**
* **Organize file structure for analytical queries in modern DWH**: A common method for structuring folders within a data lake is to organize data in separate folders by the degree of refinement. For example, a bronze folder might contain raw data, silver contains the cleaned, prepared, and integrated data, and gold contains data ready to support analytics, which might include final refinements such as pre-computed aggregates. If more levels of refinement are required, this structure can be modified, as needed, to include more folders.

When working with Data Lake Storage Gen2, the following should be considered:

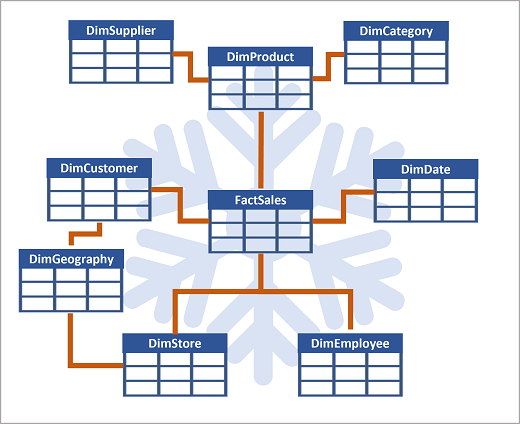
* + When data is stored in Data Lake Storage Gen2, the file size, number of files, and folder structure have an impact on performance.
  + If you store your data as many small files, this can negatively affect performance. In general, organize your data into larger-sized files for better performance (256 MB to 100 GB in size).
  + Some engines and applications might have trouble efficiently processing files that are greater than 100GB in size.
  + Sometimes, data pipelines have limited control over the raw data, which has lots of small files. It is recommended to have a "cooking" process that generates larger files to use for downstream applications.
* **Data Warehouse**: Like all relational databases, a data warehouse contains tables in which the data you want to analyze is stored.Most commonly, these tables are organized in a schema that is optimized for multidimensional modeling, in which numerical measures associated with events known as facts can be aggregated by the attributes of associated entities across multiple dimensions. For example, measures associated with a sales order (such as the amount paid or the quantity of items ordered) can be aggregated by attributes of the date on which the sale occurred, the customer, the store, and so on.

A common pattern for relational data warehouses is to define a schema that includes two kinds of table: dimension tables and fact tables.

* + **Dimension tables**: Dimension tables describe business entities, such as products, people, places, and dates. (like master tables). In addition to attribute columns, a dimension table contains a unique key column that uniquely identifies each row in the table. In fact, it's common for a dimension table to include two key columns:
    - **Surrogate key**: It is specific to the data warehouse and uniquely identifies each row in the dimension table - usually an incrementing integer number.It's often easiest to use an **IDENTITYcolumn** to auto-generate an incrementing surrogate key (otherwise you need to generate unique keys every time you load data).
    - **Alternate key**: It is used to identify a specific instance of an entity in the transactional source system- such as a product code or a customer ID. Often called as a natural or business key.
    - **Why have two keys?**:
      * The data warehouse may be populated with data from multiple source systems, which can lead to the risk of duplicate or incompatible business keys.
      * Simple numeric keys generally perform better in queries that join lots of tables - a common pattern in data warehouses.
      * Attributes of entities may change over time - for example, a customer might change their address. Since the data warehouse is used to support historic reporting, you may want to retain a record for each instance of an entity at multiple points in time; so that, for example, sales orders for a specific customer are counted for the city where they lived at the time the order was placed. In this case, multiple customer records would have the same business key associated with the customer, but different surrogate keys for each discrete address where the customer lived at various times.
    - t's common for a data warehouse to include a dimension table that represents time. This table enables data analysts to aggregate data over temporal intervals. Depending on the type of data you need to analyze, the lowest granularity (referred to as the grain) of a time dimension could represent times (to the hour, second, millisecond, nanosecond, or even lower), or dates.
  + **Fact tables**:Fact tables store details of observations or events; for example, sales orders, stock balances, exchange rates, or recorded temperatures. A fact table contains columns for numeric values that can be aggregated by dimensions.In addition to the numeric columns, a fact table contains key columns that reference unique keys in related dimension tables.
  + **Data warehouse schema designs**: In most transactional databases that are used in business applications, the data is normalized to reduce duplication. In a data warehouse however, the dimension data **is generally de-normalized to reduce the number of joins required to query** the data.Often, a data warehouse is organized as a **star schema**, in which a fact table is directly related to the dimension tables, as shown in this example:



The attributes of an entity can be used to aggregate measures in fact tables over multiple hierarchical levels - for example, to find total sales revenue by country or region, city, postal code, or individual customer.The attributes for each level can be stored in the same dimension table. However, when an entity has a large number of hierarchical attribute levels, or when some attributes can be shared by multiple dimensions (for example, both customers and stores have a geographical address), it can make sense to **apply some normalization to the dimension tables** and create a **snowflake schema (in which dimension tables are related to one another)**, as shown in the following example:



In this case, the DimProduct table has been normalized to create separate dimension tables for product categories and suppliers, and a DimGeography table has been added to represent geographical attributes for both customers and stores. Each row in the DimProduct table contains key values for the corresponding rows in the DimCategory and DimSupplier tables; and each row in the DimCustomer and DimStore tables contains a key value for the corresponding row in the DimGeography table.

* + **Staging tables**are used as temporary storage for data as it's being loaded into the data warehouseto ingest data from source systems.
  + When designing a star schema model for small or medium sized datasets you can use your preferred database, such as **Azure SQL**. For larger data sets you may benefit from implementing your data warehouse in **Azure Synapse Analytics instead of SQL Server**. It's important to understand **some key differences when creating tables** in Synapse Analytics:
    - **Data integrity constraints**:Dedicated SQL pools in Synapse Analytics don't support foreign key and unique constraints as found in other relational database systems like SQL Server. This means that jobs used to load data must maintain uniqueness and referential integrity for keys, without relying on the table definitions in the database to do so.
    - **Indexes**: While Synapse Analytics dedicated SQL pools support clustered indexes as found in SQL Server, the default index type is clustered columnstore. This index type offers a significant performance advantage when querying large quantities of data in a typical data warehouse schema and should be used where possible. However, some tables may include data types that can't be included in a clustered columnstore index (for example, VARBINARY(MAX)), in which case a clustered index can be used instead.
    - **Distribution**: Azure Synapse Analytics dedicated SQL pools use a massively parallel processing (MPP) architecture, as opposed to the symmetric multiprocessing (SMP) architecture used in most OLTP database systems. In an MPP system, the data in a table is distributed for processing across a pool of nodes. Synapse Analytics supports the following kinds of distribution:
      * **Hash**: A deterministic hash value is calculated for the specified column and used to assign the row to a compute node.
      * **Round-robin**: Rows are distributed evenly across all compute nodes.
      * **Replicated**: A copy of the table is stored on each compute node.

**Dimension table**: Use **replicated distribution** for smaller tables to avoid data shuffling when joining to distributed fact tables. If tables are too large to store on each compute node, use **hash distribution**

**Fact Table**: Use hash distribution with clustered columnstore index to distribute fact tables across compute nodes.

**Staging Table**: Use round-robin distribution for staging tables to evenly distribute data across compute nodes.

* **Notebook**:
  + You can create code cell for writing code and markup cell for writing notes.
  + Select the actions ellipses (...) on the top-right corner of the cell, and then select **Toggle parameter cell.**  After toggling this option, you will see the **Parameters tag** on the cell.
  + Under pipeline run, in Notebook activity run details,you can select Playback (1) to watch a playback of the progress through the jobs (2). At the bottom, you can view the Diagnostics and Logs with different filter options (3). To the right (4), we can view the run details, such as the duration, Livy ID, Spark pool details, and so on. Select the View details link on a job to view its details (5).

1. a
2. **Repos:**

* <https://github.com/MicrosoftDocs/mslearn-store-data-in-azure.git>

1. **Exercises:**

* [Exercise - use the Azure Storage SDK to list, write to and read from blobs using App Service Web App.](https://docs.microsoft.com/en-us/learn/modules/store-app-data-with-azure-blob-storage/4-create-storage-resources?pivots=csharp)
* [Create ADF using PowerShell](https://docs.microsoft.com/en-us/learn/modules/data-integration-azure-data-factory/7-set-up)
* [Linked Service JSON](https://docs.microsoft.com/en-us/learn/modules/data-integration-azure-data-factory/8-create-linked-services)
* [Dataset JSON](https://docs.microsoft.com/en-us/learn/modules/data-integration-azure-data-factory/9-create-datasets)
* [Pipeline and Activities JSON](https://docs.microsoft.com/en-us/learn/modules/data-integration-azure-data-factory/10-create-activities-pipelines)
* [Setup Self Hosted IR](https://docs.microsoft.com/en-us/learn/modules/petabyte-scale-ingestion-azure-data-factory/5-manage-self-hosted-integration-runtime)
* Exercise - [Create Window Transformation](https://docs.microsoft.com/en-us/learn/modules/code-free-transformation-scale/4-author-azure-data-factory-mapping-data-flow)
* [Exercise - wrangling data using PowerQuery](https://docs.microsoft.com/en-us/learn/modules/code-free-transformation-scale/6-use-wrangling-data)
* [Exercise - Integrate SQL server integration services (SSIS) packages within Azure Data Factory](https://docs.microsoft.com/en-us/learn/modules/code-free-transformation-scale/8-integrate-sql-server-integration-services-packages)
* [Delta - Slowly changing dimensions (SCD)](https://docs.microsoft.com/en-us/learn/modules/populate-slowly-changing-dimensions-azure-synapse-analytics-pipelines/4-exercise-design-implement-type-1-dimension)
* \* [Exercise - Integrate a Notebook within Azure Synapse Pipelines](https://docs.microsoft.com/en-us/learn/modules/orchestrate-data-movement-transformation-azure-data-factory/5a-exercise-integrate-notebook-azure-synapse-pipelines)
* \* [ADF Development using Python](https://docs.microsoft.com/en-us/learn/modules/operationalize-azure-data-factory-pipelines/2-understand-language-support-azure-data-factory)
* [Exercise - Identify modern data warehouse architecture components](https://docs.microsoft.com/en-us/learn/modules/design-modern-data-warehouse-using-azure-synapse-analytics/4-exercise-identify-architecture-components)
* [ML in spark Notebook](https://docs.microsoft.com/en-us/learn/modules/design-modern-data-warehouse-using-azure-synapse-analytics/8-serve-data-for-analysis)
* \* [Create data warehouse tables](https://docs.microsoft.com/en-us/learn/modules/design-multidimensional-schema-to-optimize-analytical-workloads/3-create-tables)
* \* [Use external tables with Synapse SQL](https://docs.microsoft.com/en-us/azure/synapse-analytics/sql/develop-tables-external-tables?tabs=hadoop)
* \* [Indexes on dedicated SQL pool tables](https://docs.microsoft.com/en-us/azure/synapse-analytics/sql-data-warehouse/sql-data-warehouse-tables-index)
* \* [Tutorial: Load the New York Taxicab dataset](https://docs.microsoft.com/en-us/azure/synapse-analytics/sql-data-warehouse/load-data-from-azure-blob-storage-using-copy#load-the-data-into-your-data-warehouse)

1. **Notebook Important Code:**
   1. Create temporary view in Python:

df.createOrReplaceTempView("top\_purchases")

* 1. Create temporary view in SQL and also refer temporary view in SQL created by Python:
* CREATEORREPLACETEMPORARYVIEW top\_5\_products
* AS
* selectUserId, ProductId, ItemsPurchasedLast12Months
* from (select \*,
* row\_number() over (partitionbyUserIdorderby ItemsPurchasedLast12Months desc) asseqnum
* from**top\_purchases**
* ) a
* whereseqnum<= 5andIsTopProduct == trueandIsPreferredProduct = true

orderbya.UserId

* 1. Refer temporary view in Python created by SQL:

top5Products = sqlContext.table("top\_5\_products")

* 1. Select from Select:

Graphical user interface, text, application

Description automatically generated

* 1. Extract Data from JSON in SQL using JSON functions, such as JSON\_VALUE and ISJSON

Table

Description automatically generated with medium confidence

Graphical user interface, text, application

Description automatically generated

* 1. The COPY command and PolyBase can be used to import data from various formats into the SQL pool, either through T-SQL scripts like we see here, or from orchestration pipelines.



* 1. You can set the following path pattern on the ADLS Gen2 output:tran/sensor/{datetime:yyyy}/{datetime:MM}/{datetime:dd}
  2. Graphical user interface, application

     Description automatically generatedCopy Streaming data from Event Hub to Synapse SQL pool:
  3. Create External Table:

-- External data source links to data lake location

CREATEEXTERNALDATASOURCEStagedFiles

WITH (

LOCATION = 'https://mydatalake.blob.core.windows.net/data/stagedfiles/'

);

GO

-- External format specifies file format

CREATEEXTERNALFILEFORMATParquetFormat

WITH (

FORMAT\_TYPE = PARQUET,

DATA\_COMPRESSION = 'org.apache.hadoop.io.compress.SnappyCodec'

);

GO

-- External table references files in external data source

CREATEEXTERNALTABLEdbo.ExternalStageProduct

(

ProductIDNVARCHAR(10) NOTNULL,

ProductName NVARCHAR(200) NOTNULL,

ProductCategoryNVARCHAR(200) NOTNULL,

Color NVARCHAR(10),

SizeNVARCHAR(10),

ListPriceDECIMALNOTNULL,

Discontinued BITNOTNULL

)

WITH

(

DATA\_SOURCE = StagedFiles,

LOCATION = 'products/\*.parquet',

FILE\_FORMAT = ParquetFormat

);

GO

1. S
2. S
3. S
4. S
5. S
6. S
7. S
8. S
9. S
10. **Powershell Code:**
    1. **To Stop all triggers before deployment**:

$triggersADF = Get-AzDataFactoryV2Trigger -DataFactoryName $DataFactoryName -ResourceGroupName $ResourceGroupName

$triggersADF | ForEach-Object { Stop-AzDataFactoryV2Trigger -ResourceGroupName $ResourceGroupName -DataFactoryName $DataFactoryName -Name $\_.name -Force }