Azure Mini Project End-to-end ETL Discussions

1..Why should one use Azure Key Vault when working in the Azure environment? What are the alternatives to using Azure Key Vault? What are the pros and cons of using Azure Key Vault?

Storing application secrets in Azure Key Vault allows one to control their distribution. Key Vault greatly reduces the chances that secrets may be accidentally leaked. When using Key Vault, application developers no longer need to store security information in their application

- Alternatives:

Azure Key Vault has many alternatives. Google Cloud KMS, and AWS KMS are two of them. Google KMS is a cloud service for managing encryption keys for other Google cloud services. AWS Key Management Service (KMS) is an Amazon Web Services product that allows administrators to create, delete and control keys that encrypt data stored in AWS databases and products.

Here is a comparison of them:

Feature	AWS KMS	Azure Key Vault	Google Cloud KMS
Key Storage	Appliance (Software + Hardware)	Appliance* (Software)	Appliance (Software + Hardware)
FIPS 140-2 Level	Level 2	Level 2	Level 1
Key Types	Symmetric and Asymmetric	Asymmetric	Symmetric and Asymmetric

Symmetric Key Length	256-bit AES	None	256-bit AES
Plain-text size limit	4KB	0.25KB	64KB
Key Capabilities	• AWS Managed Service • Encryption/Decryption • Sign/Verify • Auditing • REST APIs	• Support Customer Managed Keys • Support tokens, passwords, certificates, API keys, and other secrets • Encryption/Decryption • Sign/Verify • Key Vault logging • REST APIs	• Support Customer Managed Keys • Encryption/Decryption • Sign/Verify • Auditing • REST APIs

- Azure Key Vault has following benefits:
    - Increase security and control over keys and passwords. Applications have no direct access to keys
    - Easy to use and user friendly. . Create and import encryption keys in minutes
    - Can be easily integrated with Python, Node.js, Java, .NET and other popular tools

2. How do you achieve the loop functionality within an Azure Data Factory pipeline? Why would you need to use this functionality in a data pipeline?

We can use ForEach Activity. The ForEach Activity defines a repeating control flow in an Azure Data Factory . This activity is used to iterate over a collection and executes specified activities in a loop.

By using ForEach Activity, we can execute the same set of activities or pipelines multiple times, with different values each time. In this way, we don't need to write the same code repeatedly.

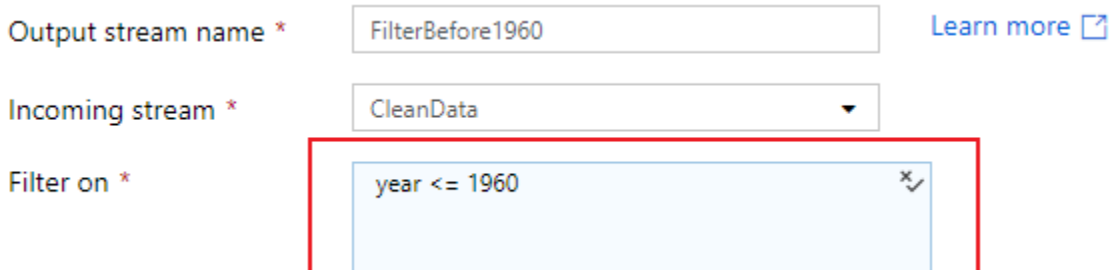
3. What are expressions in Azure Data Factory? How are they helpful when designing a data pipeline (please explain with an example)?

In Azure data factory, in mapping data flow of creating pipelines, many transformation properties are entered as expressions. These expressions are composed of column values, parameters, functions, operators, and literals that evaluate to a Spark data type at run time.

They can help in the following way:

- You can implement more complex business logic
- You can use parameters to pass external values into pipelines, datasets, linked services, and data flows. By parameterizing resources, you can reuse them with different values each time.

For example :we can use an expression to define an output stream which will be used to filter out records where year is equal or smaller than 1960:



4. What are the pros and cons of parametrizing a dataset in Azure Data Factory pipeline's activity?

Pros	By parameterizing a dataset, we can reuse them with different values each time, which provides flexibility and avoids creating multiple datasets.
Cons	We will need to provide values when using these datasets at run time.

:

5. What are the different supported file formats and compression codecs in Azure Data Factory? When will you use a Parquet file over an ORC file? Why would you choose an AVRO file format over a Parquet file format?

5.1 Azure Data Factory supports following formats

	- Avro format - Binary format - Delimited text format - JSON format - ORC format - Parquet format

Among them, Parquet and ORC are columnar formats which offer higher compression. Avro are row-based and has a lower compression rate.

5.2 Parquet vs ORC

ORC File Format

The Optimized Row Columnar (ORC) file format was designed to overcome the limitations of other file formats. It has many advantages such as:

- A single file as the output of each task.
- Hive type support including DateTime, decimal, and the complex types (struct, list, map, and union)
- Concurrent reads of the same file using separate RecordReaders
- Has the ability to split files without scanning for markers
- Metadata stored using Protocol Buffers, which allows the addition and removal of fields

ORC vs. PARQUET

- PARQUET is more capable of storing nested data.
- ORC is more capable of Predicate Pushdown.
- ORC supports ACID properties.
- ORC is more compression efficient
- One key difference between the Parquet file and an ORC file is that ORC is better optimized for Hive, whereas Parquet works really well with Apache Spark.

So I will use Parquet over ORC when writing and reading data in Apache Spark

5.3 Parquet vs Avro

Avro format is a row-based storage format for Hadoop, which is widely used as a serialization platform.

- Avro format stores the schema in JSON format, making it easy to read and interpret by any program. The data itself is stored in a binary format making it compact and efficient in Avro files.

- Avro format is a language-neutral data serialization system. It can be processed by many languages (currently C, C++, C#, Java, Python, and Ruby).

- A key feature of Avro format is the robust support for **data schema**s that changes over time, i.e., schema evolution. Avro handles schema changes like missing fields, added fields, and changed fields.

- The Avro format is the ideal candidate for storing data in a data lake landing zone because:

    1. Data from the landing zone is usually read as a whole for further processing by downstream systems (the row-based format is more efficient in this case).

    2. Downstream systems can easily retrieve table schemas from Avro files (there is no need to store the schemas separately in an external meta store).

    3. Any source schema change is easily handled (schema evolution).

Parquet file format is a columnar storage format.

- The columnar storage format is more efficient when you need to query a few columns from a table. It will read only the required columns since they are adjacent, thus minimizing IO.
- One of the unique features of Parquet is that it can store data with nested structures in a columnar fashion too. This means that in a Parquet file format, even the nested fields can be read individually without reading all the fields in the nested structure.

AVRO vs. PARQUET

1. AVRO is a row-based storage format, whereas PARQUET is a columnar-based storage format.
2. PARQUET is much better for analytical querying,
3. Writiing operations in AVRO are better than in PARQUET.
4. AVRO is much matured than PARQUET when it comes to schema evolution. PARQUET only supports schema append, whereas AVRO supports a much-featured schema evolution, i.e., adding or modifying columns.

Overall, PARQUET is ideal for querying a subset of columns in a multi-column table. AVRO is ideal in the case of ETL operations, where we need to query all the columns.

References

Expressions	How to use parameters and expressions in Azure Data Factory - Azure Data Factory | Microsoft Docs
Data Formats	Big Data File Formats (clairvoyant.ai) Choosing an HDFS data storage format- Avro vs. Parquet and more - Sta… (slideshare.net) How to choose between Parquet, ORC and AVRO for S3, Redshift and Snowflake? - BryteFlow