Member-only story

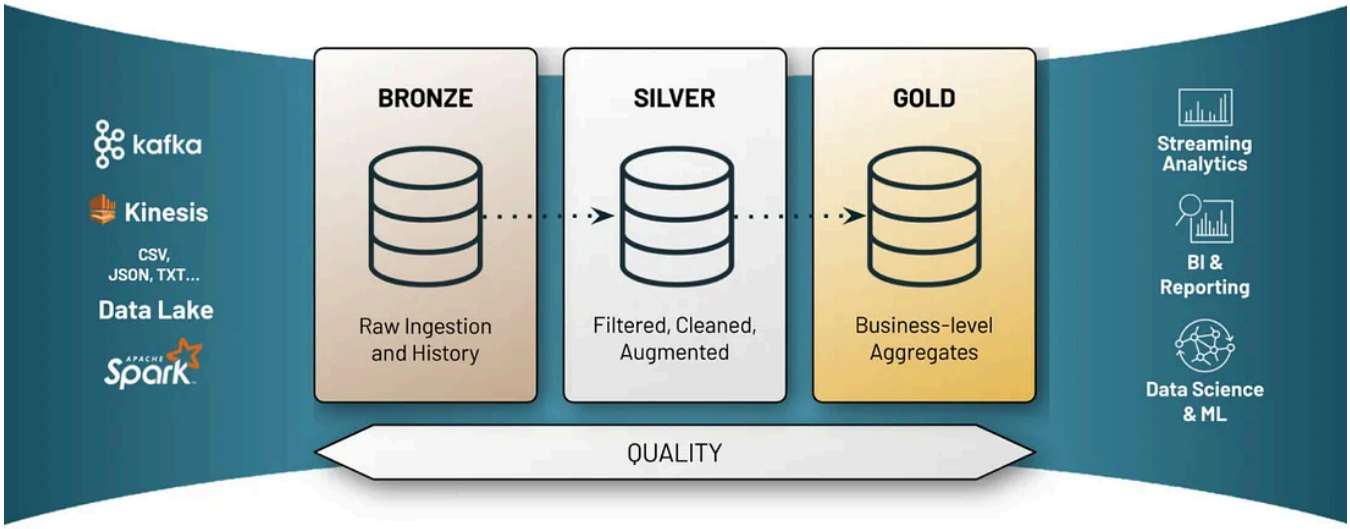
Medallion architecture: best practices for managing Bronze, Silver and Gold

Piethein Strengholt · Follow

10 min read · Jan 13, 2023

Listen Share More

Many of my clients employ a Medallion structure to logically arrange data in a Lakehouse. They process incoming data through various stages or layers. The most recognized layout, illustrated below, incorporates Bronze, Silver, and Gold layers, thus the term "Medallion architecture" is used.



Although the 3-layered design is common and well-known, I have witnessed many discussions on the scope, purpose, and best practices on each of these layers. I also observe that there's a huge difference between theory and practice. So, let me share my personal reflection on how the layering of your data architecture should be implemented.

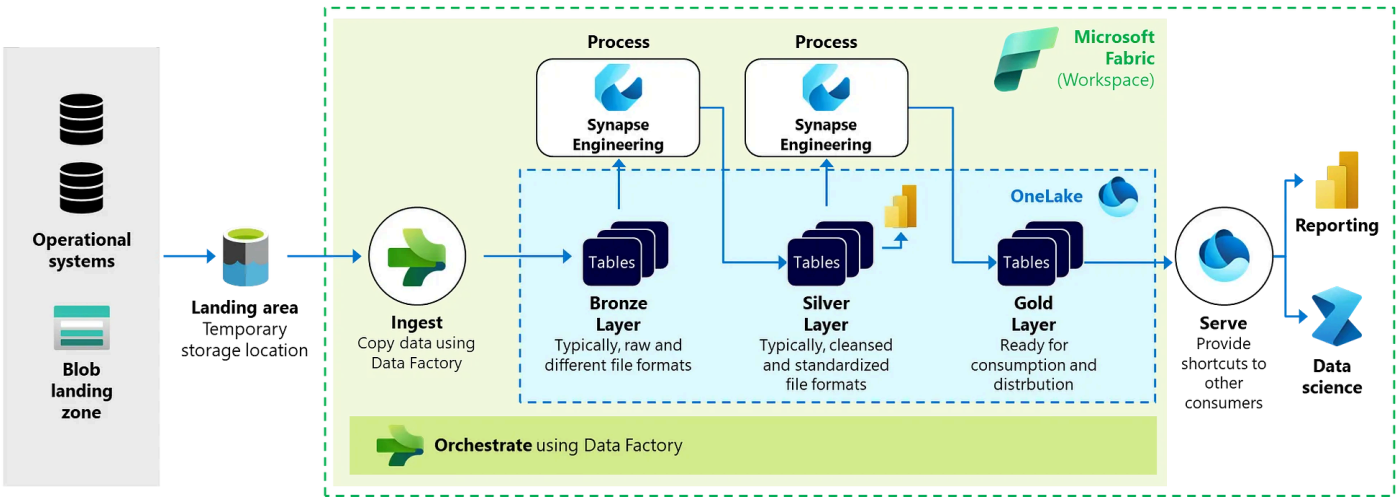
Data platform strategy

The first and most important consideration for layering your architecture is determining how your data platform is used. A centralized and shared data platform is expected to have quite a different structure than a federated multi-platform structure that is used by many domains. The layering also varies based on whether you align platform(s) with the source-system side or consuming side of your architecture. A source-system aligned platform is usually easier to standardize in terms of layering and structure than a consumer-aligned platform given the more diverse data usage characteristics on the consumption side.

With these considerations in mind, let's explore each layer after each layer. For each layer, I first provide some abstract and high-level objectives. After that, I'll make the layering more specific with observations from the field.

Landing area

A landing area, or landing zone, is an optional level frequently implemented by organizations setting up a data platform. It serves as a temporary storage location for data gathered from various sources before it is transferred into the Bronze layer. This layer becomes particularly necessary when extracting data from the target source system proves challenging, such as when dealing with external clients or SaaS vendors. In these instances, there may be a dependency, or data might be received in an unsuitable file format or structure.



Medallion Lakehouse reference architecture for Microsoft Fabric

The design of a landing zone can differ significantly among organizations. It's often a straightforward Blob storage account, but in some cases, it is integrated into the

data lake services, such as a container, bucket, or a specific folder where data ingestion occurs. The data housed in landing zones is frequently very diverse, with file formats ranging from CSV, JSON, XML, Parquet, Delta, and so forth.

Bronze layer

The bronze layer is usually a reservoir that stores data in its natural and original state. It contains unvalidated data (without having to first define schemas). In this layer you either get data using full loads or delta loads. Data that is stored in bronze has usually the following characteristics:

- Maintains the raw state of the data source in the structure “as-is”.
- Data is immutable (read-only).
- Managed using interval partitioned tables, for example, using a YYYYMMDD or datetime folder structure.
- Retains the full (unprocessed) history of each dataset in an efficient storage format, for example, Parquet or Delta.
- For transactional data: Can be appended incrementally and grow over time.
- Provides the ability to recreate any state of a given data system.
- Can be any combination of streaming and batch transactions.
- May include extra metadata, such as schema information, source file names or recording the time data was processed.

A common query I encounter is, “Which file format is superior? Should I opt for Delta or Parquet?” While Delta offers faster speed, its benefits aren’t as pronounced if your data is already versioned or historized using a folder structure. As such, maintaining a transaction log or applying versioning isn’t crucial. Bronze data is typically new or appended data. Therefore, selecting Parquet is perfectly acceptable. However, you could also choose Delta to remain consistent with other layers.

Some argue that Bronze data is beneficial for business users conducting queries or ad-hoc analyses. However, based on my experience with customers, it’s rare for raw data to be utilized as input for these tasks. Raw data is challenging to handle as it necessitates a deep understanding of the source system’s design. You’ll also need to decipher the intricate business logic embedded within the data. Due to the frequent

presence of numerous small tables, securing it is virtually impossible. In conclusion, Bronze serves as a staging layer and a source for other layers, primarily accessed by technical accounts.

Silver layer

The Silver layer provides a refined structure over data that has been ingested. It represents a validated, enriched version of our data that can be trusted for downstream workloads, both operational and analytical. In addition to that, Silver may have the following characteristics:

- Uses data quality rules for validating and processing data.
- Typically contains only functional data. So, technical data or irrelevant data from Bronze is filtered out.
- Historization is usually applied by merging all data. Data is processed using slowly changing dimensions (SCD), either type 2 or type 4. This means additional columns are added, such as start, end and current columns.
- Data is stored in an efficient storage format; preferably Delta, alternatively Parquet.
- Uses versioning for rolling back processing errors.
- Handles missing data, standardizes clean or empty fields.
- Data is usually enriched with reference and/or master data.
- Data is often cluttered around certain subject areas.
- Data is often still source-system aligned and organized. Thus, it has not been integrated with other domain data yet.

For Silver there are a couple of attention points:

Certain individuals propose that Silver can function as a transient storage layer, enabling the deletion of outdated data or the spontaneous creation of storage accounts. Customers often inquire if I share this perspective. Well, it depends. If you aren't planning on using data in the original context for operational reporting or operational analytics, then Silver can be a temporal layer. However, if you aim to preserve historical data and employ Silver for operational reporting and analytics, then I would recommend establishing Silver as a permanent layer.

If you're handling Silver data that requires querying, it's advisable to utilize a denormalized data model. This approach eliminates the necessity for extensive joins and aligns better with the distributed column-based storage architecture. Does this imply you should abandon a 3rd-normal form or Data Vault-style data model? There doesn't appear to be strong reasons to do so. Regarding historization, the delta file format already versions Parquet files for isolation and safety. Plus, you have a history in your Bronze layer from which you can reload data. For automation and adaptability, a metadata-driven strategy is recommended for managing tables and notebooks. As for data redundancy (storing the same data multiple times), storage in a lake is less expensive compared to computational processing and data joining. In conclusion, unless you're dealing with significant daily schema changes, there seems to be no strong justification to add the complexity of a data vault. For further exploration of this topic, Simon Whiteley's video provides numerous considerations and is highly recommended.

I often discuss whether one should already be integrating data between applications and source systems. This issue is slightly more complex. My advice is to, if possible, separate things for easier management and isolation of concerns. This suggests that, for situations where you're using Silver for operational reporting or analytics, it's advised not to prematurely merge and integrate data from source systems. Doing so could lead to unnecessary connection points between applications. For instance, a user interested only in data from a single source would also be linked to other systems, as the data is first combined in a harmonized layer before being provided. Consequently, these data users are more likely to experience potential impacts from other systems. If you're striving for such an isolated design, the integration or combination of data from different sources should be moved up to the Gold layer.

The aforementioned reasoning applies equally to aligning your lake houses with the source-system side of your architecture. If you have intentions of constructing data products and are adamant about aligning data ownership, then I advise against your engineers prematurely cross-joining data from applications in other domains.

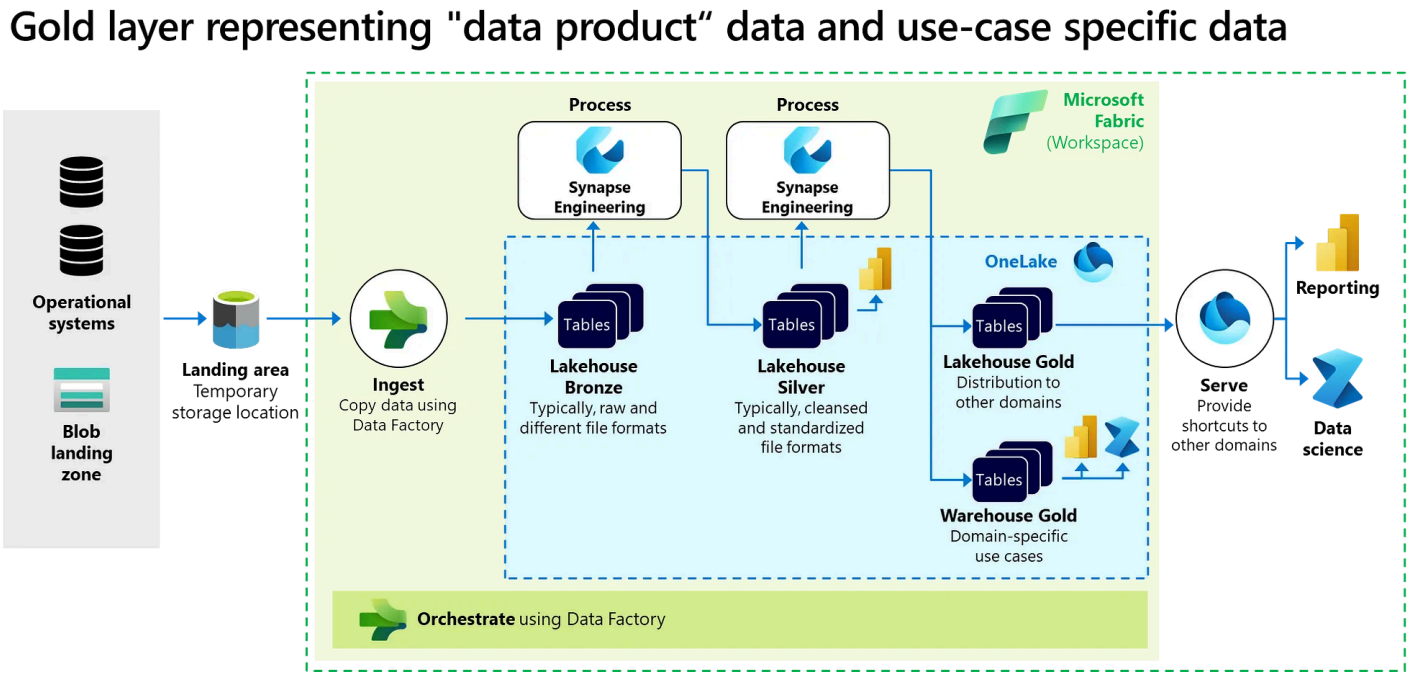
When considering enrichments, such as calculations, there are certain factors to bear in mind. If your goal is to enable operational reporting and this requires enrichments, I suggest you start enriching your data in the Silver layer. This might lead to some extra calibration when merging data later in the Gold stage. While this may require additional effort, the flexibility it offers makes it a worthwhile endeavor.

Gold layer

In a Lakehouse architecture, the Gold layer houses data that is structured in "project-specific" databases, making it readily available for consumption. This integration of data from various sources may result in a shift in data ownership. As for the Gold layer, I suggest utilizing a denormalized and read-optimized data model with fewer joins, such as a Kimball-style star schema, depending on your specific use cases. In addition to that, the Gold layer is expected having the following characteristics:

- Gold tables represent data that has been transformed for consumption or use cases.
- Data is stored in an efficient storage format, preferably Delta.
- Gold uses versioning for rolling back processing errors.
- Historization is applied only for the set of use cases or consumers. So, Gold can be a selection or aggregation of data that's found in Silver.
- In Gold you apply complex business rules. So, it uses many post-processing activities, calculations, enrichments, use-case specific optimizations, etc.
- Data is highly governed and well-documented.

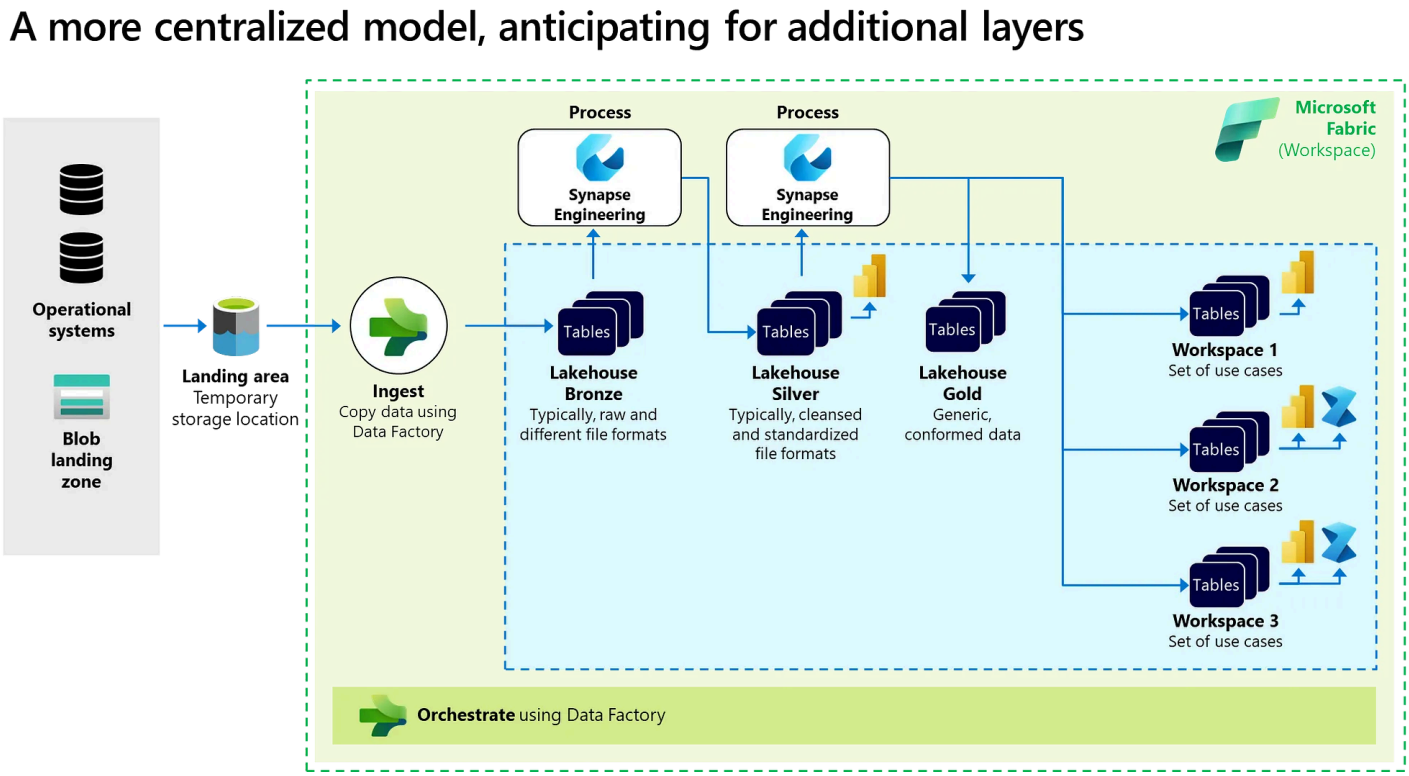
Gold is frequently the most intricate layer due to its design being dependent on the breadth of your architecture. In the most basic setup, your Lakehouses are solely in line with the source-system side. In this case, data in the Gold layer represents "data product" data, making it general and user-friendly, suitable for wide distribution to numerous other domains. Following this distribution, it's expected for the data to be housed in another platform, possibly another Lakehouse.



Depending on the breadth of your architecture, your Gold layer could be a conglomerate of data product data and use-case specific data

If this happens, it's anticipated that the Gold Layer in these platforms would comply with the exact needs of analytic consumers. Consequently, the data is modeled, having a shape and structure tailored specifically for the use case in progress. This method could foster principles where you only allow domains to function on internal data that isn't directly distributed to other domains. Therefore, some tables may be flexible, adjustable at any given time, while other tables with a formal status are consumed by other entities. A reflection of such a design is seen in the image above.

If the scope of your Lakehouse is broad and encompasses both the provider and consumer ends, anticipate additional layers. These extra layers are often referred to as workspace or presentation layers. In this configuration, the data in Gold is more universal, being integrated and ready for various use cases. These workspace or presentation layers then contain subsets of data. This model is much like the typical data modeling within data warehousing. In essence, Gold serves as a universal integration layer from which data marts or subsets can be filled.



Architecture showing a more centralized operating model. The final presentation layer could represent areas such as data warehouses, data lakes or data marts for different types of use cases.

Certain organizations utilize these workspace or presentation layers to disseminate data to other platforms or teams. They do this by selecting and/or pre-filtering data for specific scenarios. Some other organizations implement tokenization, which involves substituting sensitive data with randomized strings. Others employ extra services for data anonymization. This data can be seen as behaving similarly to “data product” data.

The gold layer may stand out among other layers due to its adherence to enterprise-wide standards. While enterprise data modeling can be seen as complex and time-consuming, many organizations still utilize it to ensure data reusability and standardization. In the Lakehouse approach, the enterprise data model provides an abstract framework for organizing data within teams’ lakehouses. The enterprise model typically only outlines certain data and requires teams to follow specific reference values or include master identifiers when sharing datasets with other teams. An alternative approach to setting enterprise-wide standards is gathering data, harmonizing, and distributing data in another Lakehouse architecture. From there, data can be consumed downstream by other domains. This approach is similar to master data management. Alternatively, you could mandate teams to take responsibility for specific harmonized entities. Other teams, then, must conform by

always using these entities. In most of the above cases, standardization and harmonization occur in the Gold layer.

When it comes to technology architecture, particularly for Gold, it's crucial to understand that choosing a database or service is a complex procedure. This process requires consideration of various factors and involves making numerous compromises. Beyond data structure, you'll need to assess your requirements in terms of consistency, availability, caching, timeliness, and indexing for enhanced performance. There are a variety of methods for storing and retrieving data, such as small or large chunks, sorted chunks, etc. Contrary to what some enthusiasts may suggest, no single service can perfectly cater to all aspects simultaneously. A typical Lakehouse architecture therefore usually comprises a blend of diverse technology services such as a serverless SQL service for ad-hoc queries, a columnar store for fast reporting, a relational database for more complicated queries, a timeseries store for IoT and stream analysis, among others.

I hope you found this a helpful article. And to all of you building a next generation data platform: enjoy the trip!

Data | Data Lake | Delta Lake | Databricks | Azure Synapse Analytics

Follow

Written by Piethein Strengholt

5.7K Followers · 25 Following

Hands-on Chief Data Officer. Working @Microsoft.

Responses (9)

Tharasp

What are your thoughts?

Amit Borole
Oct 4, 2023

Very good article .

13 Reply

Richard Kooijman
Jul 19, 2023

did someone actually proof read this?

"Data in Silver is queryable. From a data modeling standpoint, this means that for Silver it's recommended to follow a more denormalized data model. Why? Because such a design better utilizes the distributed... more

8 2 replies Reply

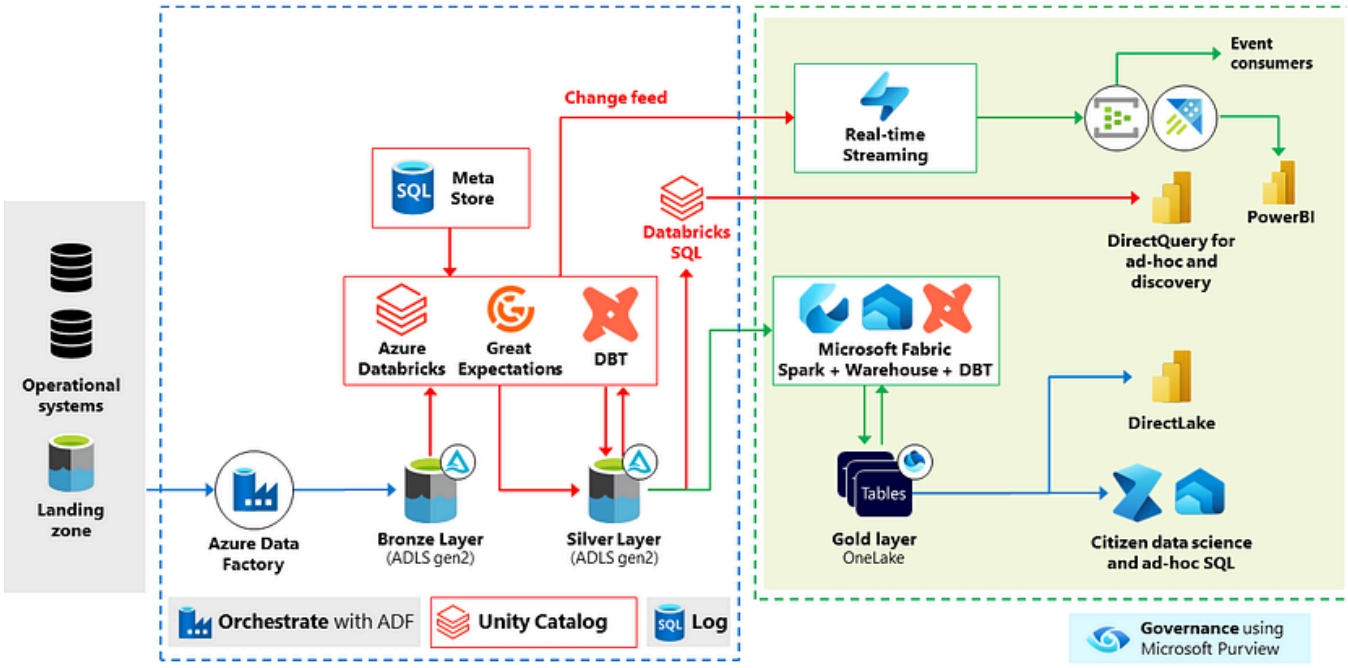
Balachandar Ganesan
Apr 5, 2023

Nice explanation and references

1 Reply

See all responses

More from Piethein Strengholt

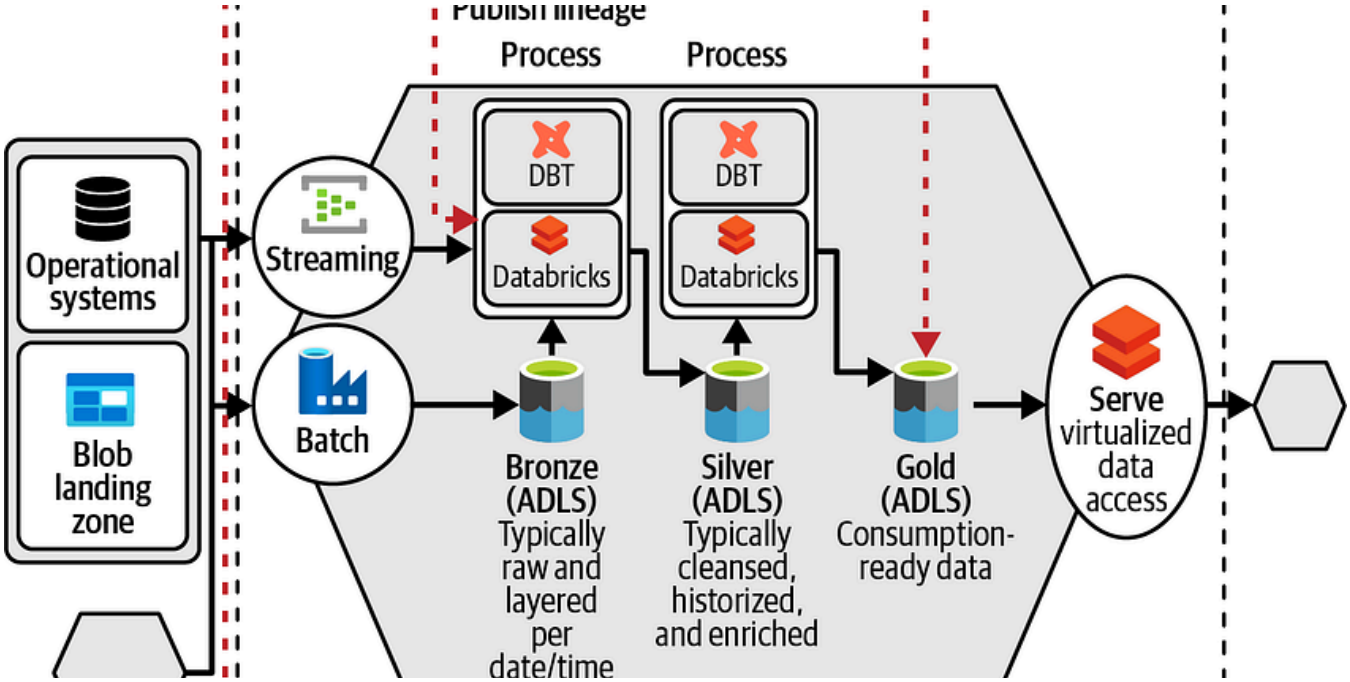


Piethein Strengholt

Integrating Azure Databricks and Microsoft Fabric

The article discusses the integration of Azure Databricks and Microsoft Fabric, presenting several architectural design options.

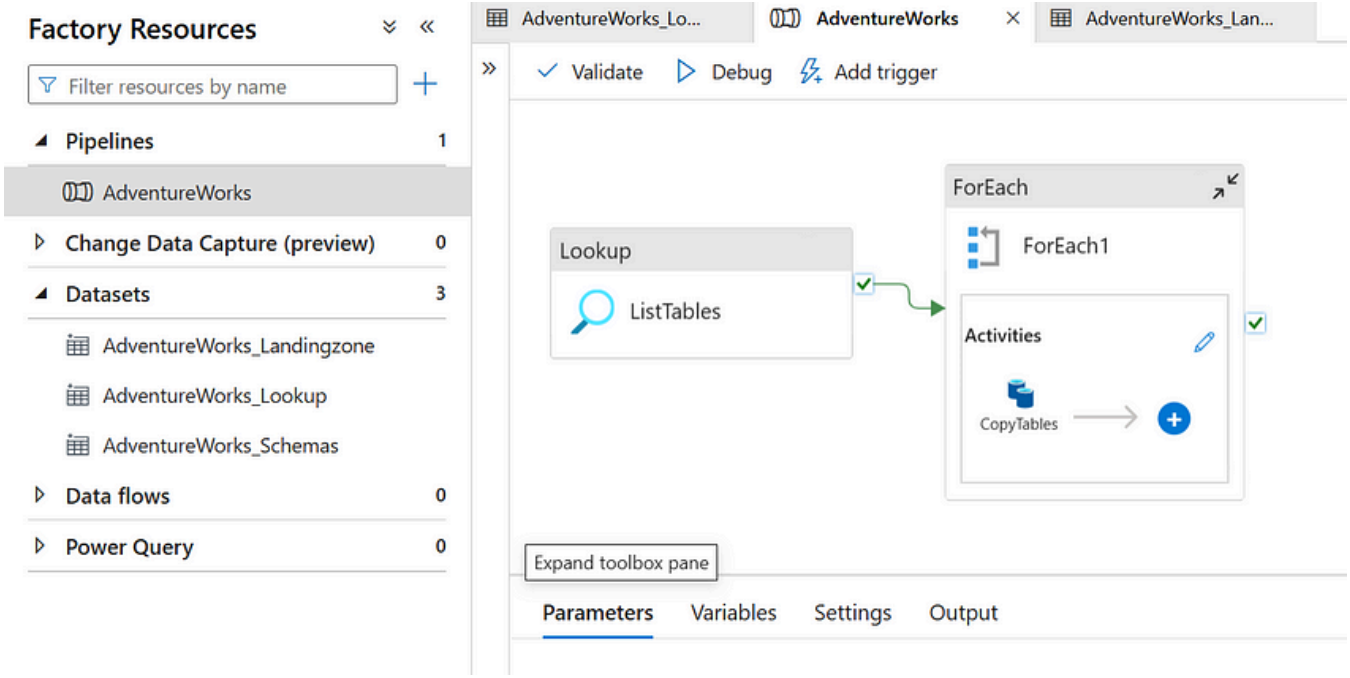
Jun 4, 2024 457 10



Piethein Strengholt

Data Management at Scale

Jul 11, 2023 538 4

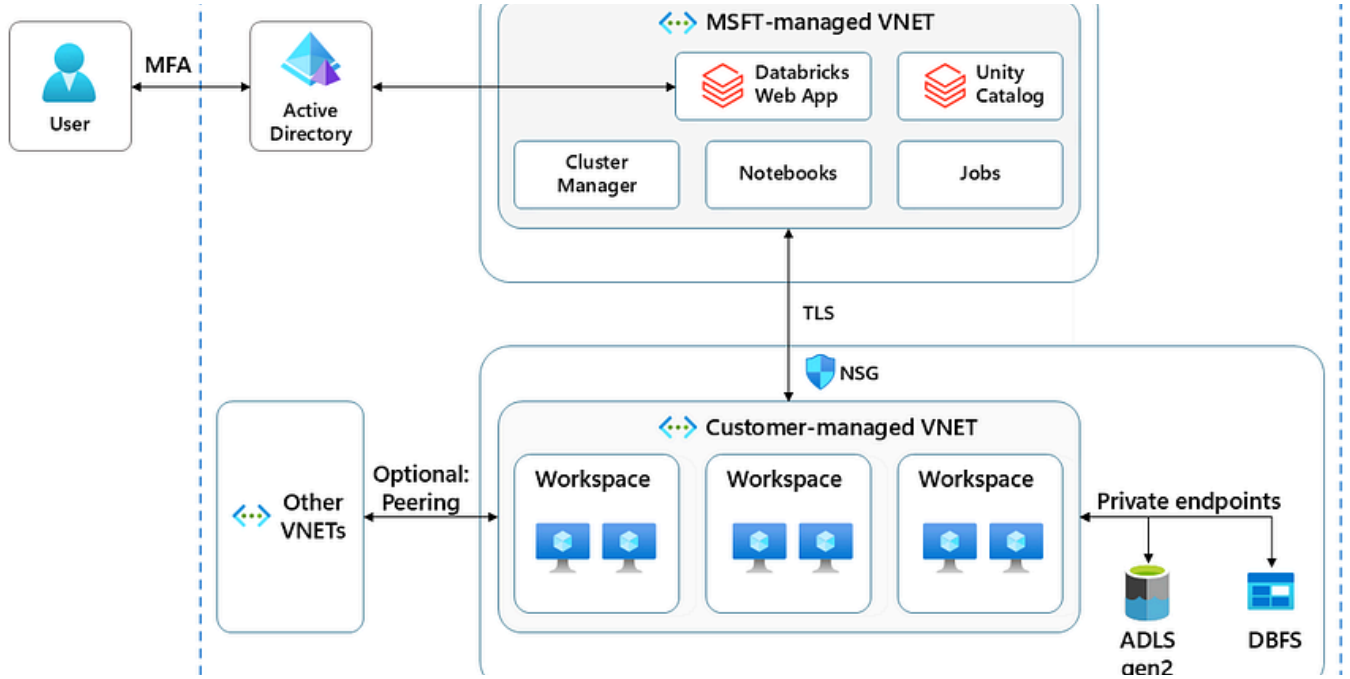


Piethein Strengholt

Construct the Bronze Layer with Azure Databricks

In the upcoming book, Building Medallion Architectures, there is an exercise demonstrating how to build a Medallion Architecture from start...

Dec 17, 2024 25 1



Piethein Strengholt

Building the Medallion foundation with Azure Databricks and Unity Catalog

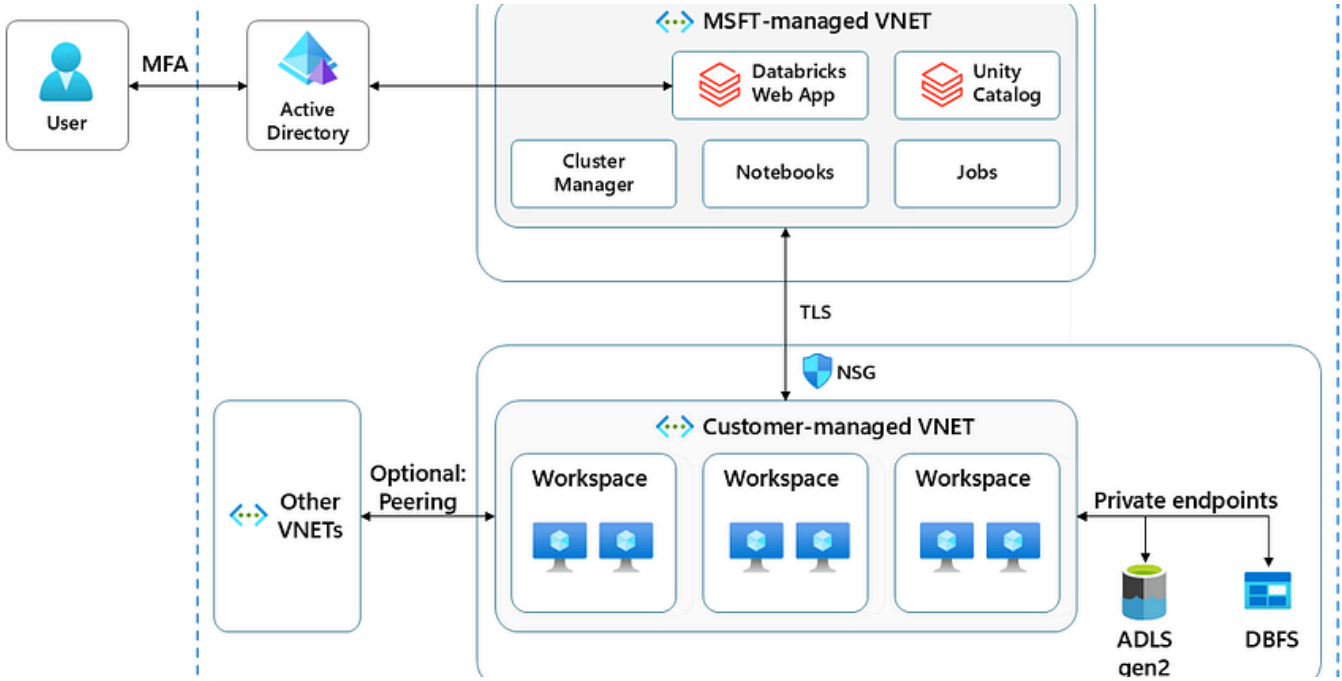
In the upcoming book, Building Medallion Architectures, I initially planned to extensively explore the process of building a Medallion...

Nov 25, 2024 73 2

See all from Piethein Strengholt

Recommended from Medium

Feb 15 143

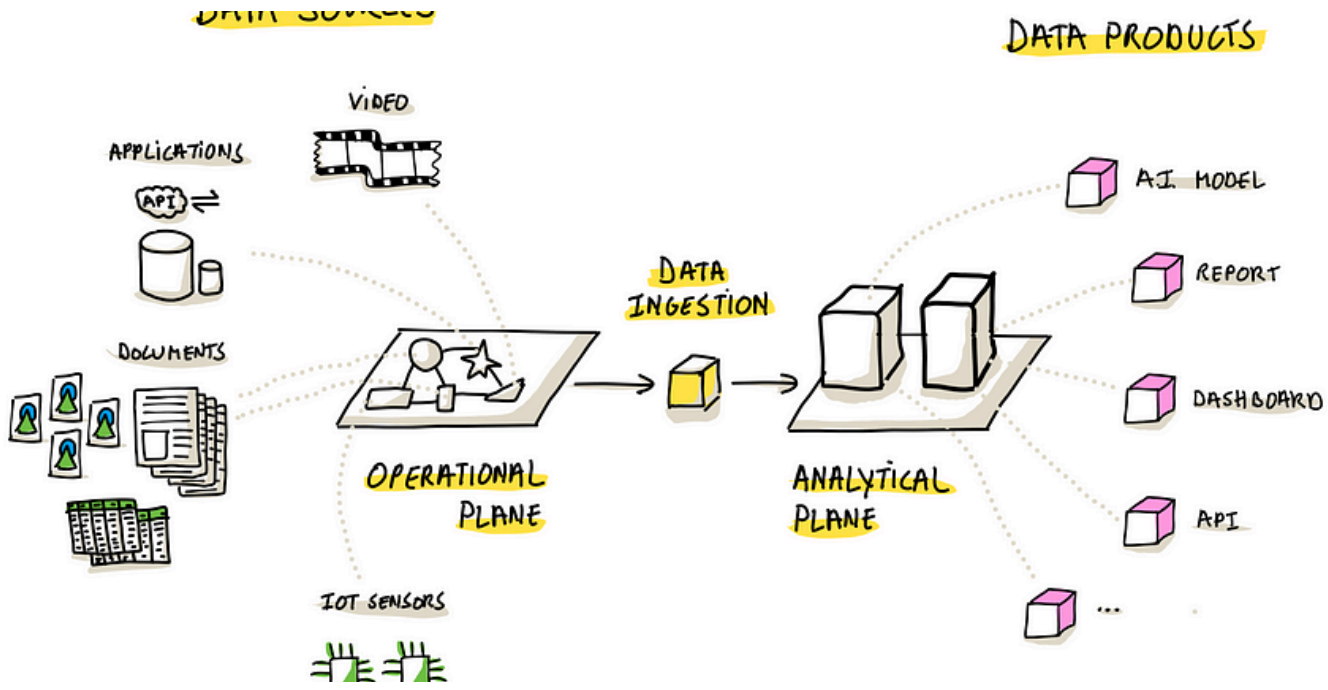


Piethein Strengholt

Building the Medallion foundation with Azure Databricks and Unity Catalog

In the upcoming book, Building Medallion Architectures, I initially planned to extensively explore the process of building a Medallion...

Nov 25, 2024 73 2

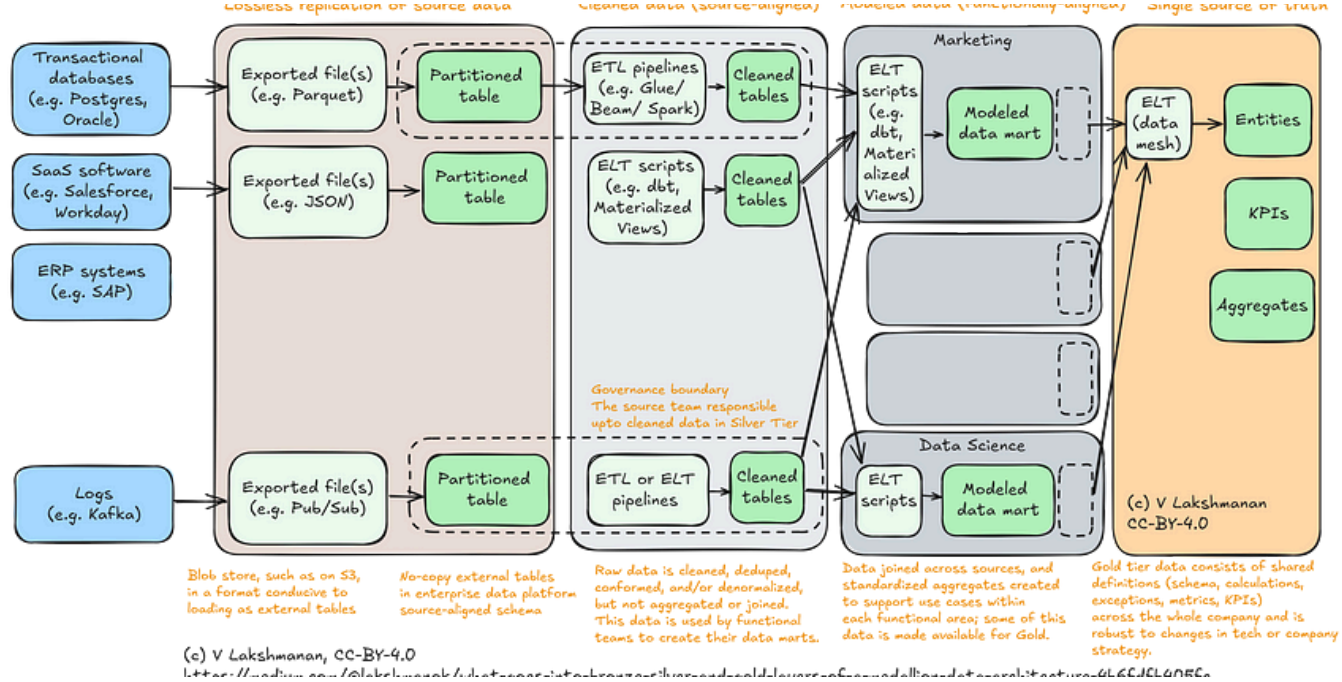


In The Modern Scientist by janmeskens

Data Ingestion — Part 1: Architectural Patterns

Over the course of two articles, I will thoroughly explore data ingestion, a fundamental process that bridges the operational and...

Nov 28, 2023 2.7K 26

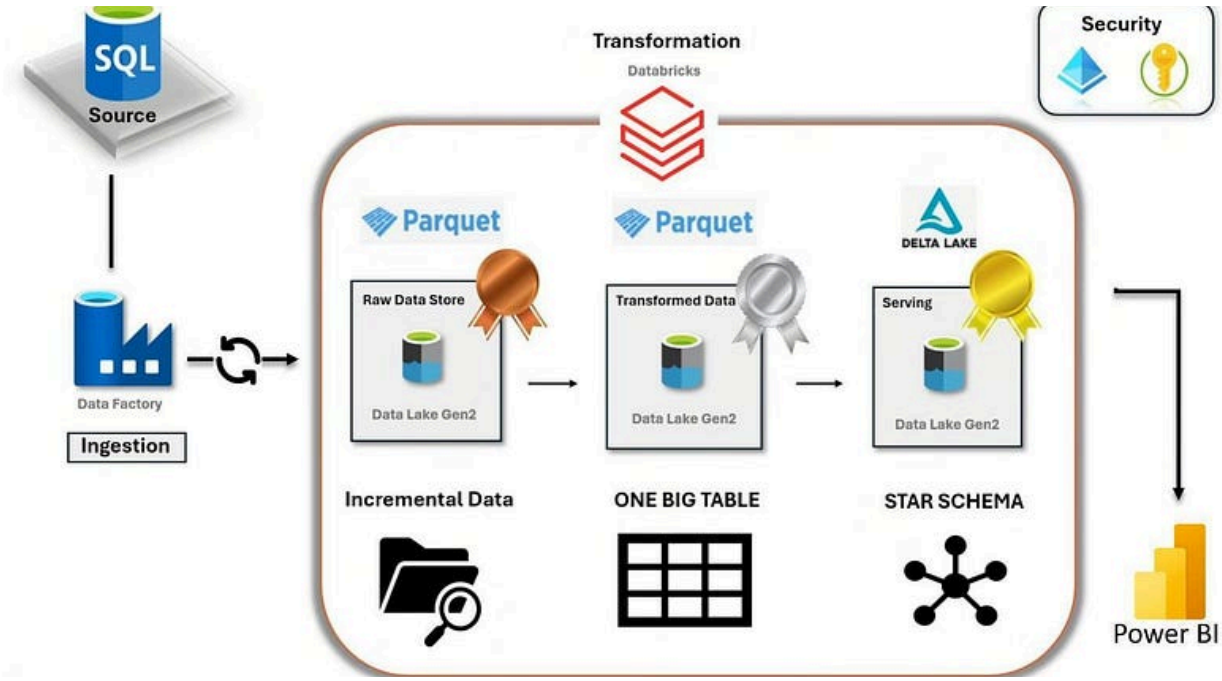


Lak Lakshmanan

What goes into bronze, silver, and gold layers of a medallion data architecture?

Here's a four-layer medallion architecture that explicitly addresses data governance and separation-of-responsibility

Sep 18, 2024 506 9



Rihab Feki

Azure End-to-End Data Engineering Project: Medallion Architecture with Databricks [Part 2]

Using Azure SQL DB, Azure Data Factory, Databricks, Delta Lake, Power BI

Jan 19 325 5

Lewis Gavin

How to Become a World-Class Data Architect

Tips and advice after 10 years of experience

Jan 30 108 6

See more recommendations