Data Lake Architecture -

A Comprehensive Design Document

Medical Data Processing Company

# Tracker

## Revision, Sign off Sheet and Key Contacts

## Change Record

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Author | Version | Change Reference |
| 06/04/2020 | Sohaib Zafar | 0.1 | Initial draft |

## Reviewers / Approval

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Version Approved | Position | Date |
|  | 1.0 | Udacity Reviewer  Enterprise Data Lake Architect |  |

## Key Contacts

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Role | Team | email |
| Sohaib Zafar | Data Architect | Medical Data Processing | zafarsohaib@gmail.com |

# **Table of Contents**

1. Purpose
2. Requirements
3. Data Lake Architecture design principles
4. Assumptions
5. Data Lake Architecture for Medical Data Processing Company
6. Design Considerations and Rationale
7. Conclusion
8. References

# **Purpose**

# Summary

“Medical Data Processing” company’s current technology stack can’t keep up with their hyper growth. As the volume of data continues to grow, the existing single node SQL Server is not able to scale. SQL Server has become a single point of failure, hosting critical customer data. The CTO is interested in a data lake solution to deal with the company’s current and future challenges. This is the detailed document with proposed data lake architecture design.

# Target Audience

* Company’s management
* Data engineers
* Data scientists
* Data and business analysts

# In-scope items

* Requirements, design, assumptions and rationale behind the proposed data lake architecture.

# Out-of-scope items

* Data Lake architecture implementation steps, pricing, data governance and machine learning.

# **Requirements**

# Summary of requirements for Data Lake

* Design a system which is highly available, scalable, resilient and performant
* Maintain one source of truth and break down data silos
* Easy integration with Machine Learning frameworks
* Can support near-real time dashboards for reporting

# Existing Technical Environment

* 1 Master SQL DB Server
* 1 Stage SQL DB Server
* 64 core vCPU
* 512 GB RAM
* 12 TB disk space (70% full, ~8.4 TB)
* 70+ ETL jobs running to manage over 100 tables
* 3 other smaller servers for Data Ingestion (FTP Server, data and API extract agents)
* Series of web and application servers (32 GB RAM Each, 16 core vCPU)

# Current Data Volume

* Data coming from over 8K facilities
* 99% zip files size ranges from 20 KB to 1.5 MB
* Edge cases - some large zip files are as large as 40 MB
* Each zip files when unzipped will provide either CSV, TXT, XML records
* In case of XML zip files, each zip file can contain anywhere from 20-300 individual XML files, each XML file with one record
* **Average zip files per day:** 77,000
* **Average data files per day:** 15,000,000
* **Average zip files per hour:** 3500
* **Average data files per hour:** 700,000
* **Data Volume Growth rate:** 15-20% YoY

# Business Requirements

* Improve uptime of overall system
* Reduce latency of SQL queries and reports
* System should be reliable and fault tolerant
* Architecture should scale as data volume and velocity increases
* Improve business agility and speed of innovation through automation and ability to experiment with new frameworks
* Embrace open-source tools, avoid proprietary solutions which can lead to vendor lock-in
* Metadata driven design - a set of common scripts should be used to process different types of incoming data sets rather than building custom scripts to process each type of data source.
* Centrally store all of the enterprise data and enable easy access

# Technical Requirements

* Ability to process incoming files on the fly (instead of nightly batch loads today)
* Separate the metadata, data and compute/processing layers
* Ability to keep unlimited historical data
* Ability to scale up processing speed with increase in data volume
* System should sustain small number of individual node failures without any downtime
* Ability to perform change data capture (CDC), UPSERT support on a certain number of tables
* Ad-hoc data analytics, interactive querying capability using SQL
* Ability to drive multiple use cases from same dataset, without the need to move the data or extract the data
  + Ability to integrate with different ML frameworks such as TensorFlow
  + Ability to create dashboards using tools such as PowerBI, Tableau, or Microstrategy
  + Generate daily, weekly, nightly reports using scripts or SQL

# Data Lake Architecture design principles

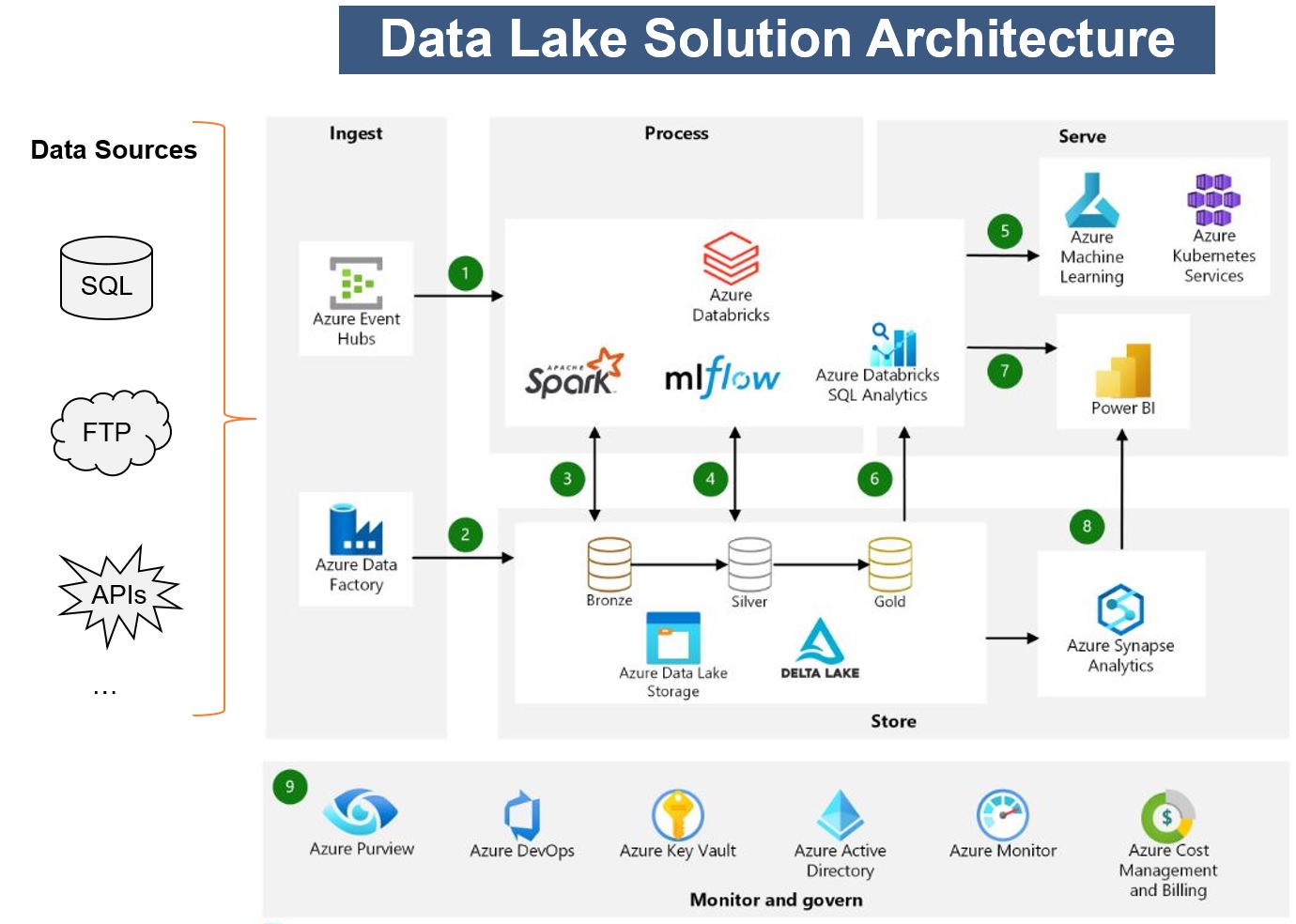
* **Open**: The solution supports open-source code, open standards and open frameworks. It also works with popular integrated development environments (IDEs), libraries, and programming languages. The solution also works with a wide range of other services through native connectors and APIs
* **Collaborative**: Data engineers, data scientists, and analysts work together on this solution. You can use collaborative notebooks, IDEs, dashboards, and other tools to access and analyze shared underlying data.
* **Simple**: Unified analytics, data science, and machine learning simplify data architecture
* **Event Sourcing**: Use “store now, think later” approach by maintaining an immutable log of all incoming events on storage. To be able to "travel back in time" and retrace your actions to discover the exact transformation that was performed on your raw data, right down to the event level. If your ETL code had a problem, you could easily correct it and run the updated code on the original, unaltered data.

# Assumptions

* It is expected that there will be hyper growth of the company in future, but it is difficult to exactly predict the capacity requirements in advance. That’s why it is better to choose a cloud-based solution which can be easily scaled as per demand rather than an on-premises solution.
* All the existing data will be migrated to Cloud.
* There will be sufficient budget and resource allocation to support the migration efforts.
* Transitioning to Data Lake should happen as soon as possible.

# Data Lake Architecture for Medical Data Processing Company

I recommend using the Azure Data Lake approach. As shown in the diagram below:



# Design Considerations and Rationale

## Ingestion Layer

In the ingest layer, we have **Azure Event Hubs** and **Azure Data Factory**.

**Event Hubs** is a big data streaming platform. It is a fully managed, real-time data ingestion service that’s simple, trusted, and scalable. Stream millions of events per second from any source to build dynamic data pipelines and immediately respond to business challenges. Keep processing data during emergencies using the geo-disaster recovery and geo-replication features.

Integrate seamlessly with other Azure services to unlock valuable insights. Allow existing Apache Kafka clients and applications to talk to Event Hubs without any code changes—you get a managed Kafka experience without having to manage your own clusters. Experience real-time data ingestion and micro-batching on the same stream.

**Why choose Event Hubs?**

* **Simple**: Build real-time data pipelines with just a couple clicks. Seamlessly integrate with Azure data services to uncover insights faster.
* **Secure**: Protect your real-time data. Event Hubs is certified by CSA STAR, ISO, SOC, GxP, HIPAA, HITRUST, and PCI.
* **Scalable**: Adjust throughput dynamically based on your usage needs and pay only for what you use.
* **Open**: Ingest data from anywhere and develop across platforms with support for popular protocols, including AMQP, HTTPS, and Apache Kafka.

List of all data formats supported by Azure Event Hubs can be found here:

<https://learn.microsoft.com/en-us/azure/data-explorer/ingest-data-event-hub-overview#data-format>

Additionally, I choose **Azure Data Factory**. It is a hybrid data integration service, which is fully managed, serverless solution enables the creation, scheduling, and orchestration of data transformation workflows. Visually integrate data sources with more than 90 built-in, maintenance-free connectors at no added cost. Easily construct ETL and ELT processes code-free in an intuitive environment or write your own code.

**Why choose Azure Data Factory?**

* **Easy-to-use:** Rehost SQL Server Integration Services (SSIS) in a few clicks and build ETL and ELT pipelines code-free, with built-in Git and CI/CD support.
* **Scalable and Cost-effective**: Enjoy a pay-as-you-go, fully managed serverless cloud service that scales on demand.
* **Powerful**: Ingest all your on-premises, and software as a service (SaaS) data with more than 90 built-in connectors. Orchestrate and monitor at scale.
* **Intelligent**: Use autonomous ETL to unlock operational efficiencies and enable citizen integrators.

List of all data stores and formats supported by Azure Data Factory can be found here:

<https://learn.microsoft.com/en-us/azure/data-factory/copy-activity-overview#supported-data-stores-and-formats>

To sum it up, Event Hubs is to create dynamic data pipelines for ingesting the streaming data (e.g., from APIs). These data streams can be fed to the processing layer if further data transformation (ETL or ELT) is needed. While Azure Data Factory can be used to easily integrate and ingest data from different data stores (for example SQL database or FTP server). It enables you to easily build code-free ETL and ELT pipelines and the data is then sent to the storage layer.

Other third party open-source tools were also considered, but at the end I decided to go with fully managed and easy to use Azure solutions. Here are few examples:

* **Apache NiFi** is an integrated data logistics platform for automating the movement of data between disparate systems. It provides real-time control that makes it easy to manage the movement of data between any source and any destination.
* **Apache Kafka** is a distributed data store optimized for ingesting and processing streaming data in real-time. Streaming data is data that is continuously generated by thousands of data sources, which typically send the data records in simultaneously.
* **Apache Flume** is a distributed, reliable, and available software for efficiently collecting, aggregating, and moving large amounts of log data. It has a simple and flexible architecture based on streaming data flows.
* **Apache Pulsar** has a reliable, easy-to-use stream processor built right in. Called Pulsar Functions, it can handle most use cases, including event-based services, filtering, real-time aggregation, routing, enrichment, and simple ETL (extract, transform, and load) operations.

## Storage Layer

In the storage layer, we choose **Azure Data Lake Storage Gen2** and **Azure Synapse Analytics**.

**Azure Data Lake Storage Gen2** is a scalable and secure data lake for high-performance analytics workloads. This service can manage multiple petabytes of information while maintaining hundreds of gigabytes of throughput. The data can be structured, semi-structured or unstructured. It usually comes from multiple heterogeneous sources such as logs, files, and media. As it is built on **Azure Blob Storage**, you'll also get low-cost, tiered storage, with high availability/disaster recovery capabilities.

**Why choose ADLS Gen2?**

* **Scalability:** Azure Storage is scalable by design whether you access via Data Lake Storage Gen2 or Blob storage interfaces. It can store and serve many exabytes of data. This amount of storage is available with throughput measured in gigabits per second (Gbps) at high levels of input/output operations per second (IOPS). Processing is executed at near-constant per-request latencies that are measured at the service, account, and file levels.
* **Performance** is optimized because you do not need to copy or transform data as a prerequisite for analysis. Compared to the flat namespace on Blob storage, the hierarchical namespace greatly improves the performance of directory management operations, which improves overall job performance.
* **Management** is easier because you can organize and manipulate files through directories and subdirectories.
* **Security** is enforceable because you can define POSIX permissions on directories or individual files.
* **Cost effective** because it is built on top of the low-cost Azure Blob Storage. The additional features further lower the total cost of ownership for running big data analytics on Azure.
* **Hadoop compatible access:** Data Lake Storage Gen2 allows you to manage and access data just as you would with a Hadoop Distributed File System (HDFS). The new ABFS driver (used to access data) is available within all Apache Hadoop environments. These environments include Azure HDInsight, Azure Databricks, and Azure Synapse Analytics.

On top of Data Lake, **Delta Lake** is an open-source storage layer built on **Spark**. Its key functions ensure data integrity with **ACID** Transactions while also allowing reading and writing from/to the same directory/table, bringing reliability to massive data lakes. ACID stands for Atomicity, Consistency, Isolation, and Durability.

Delta Lake is stored in an open data format, so you avoid data lock-in from proprietary formats and gain access to a vast open-source ecosystem. Today, thousands of companies are processing exabytes of data per month with Delta Lake.

Delta Lake forms the curated layer of the Data Lake. It stores the refined data in an open-source format. “**Delta**" is a data format based on **Apache Parquet**. Apache Parquet is an open source, column-oriented data file format designed for efficient data storage and retrieval. It provides efficient data compression and encoding schemes with enhanced performance to handle complex data in bulk.

Azure Databricks works well with a medallion architecture that organizes data into layers:

* **Bronze**: Contains raw data.
* **Silver**: Contains cleaned, filtered data.
* **Gold**: Stores aggregated data useful for business analysis.

**Key Features of Delta Lake**

* **Scalable Metadata Handling**: Delta Lakes are capable of handling even petabytes of data with ease. It stores metadata just as it stores data and users can access it using Describe Detail feature.
* **Schema Enforcement**: Delta Lakes are widely used by companies because it enforces the schema. It reads the Schema as a part of the metadata and looks at every column, data type, etc.
* **Unified Batch and Streaming**: Delta Lakes provides a single architecture for reading stream data and batch data as well.
* **Upserts and Deletes**: Delta allows you to make upserts easily. These upserts or merges are like SQL Merges into the Delta table. It allows you to merge data from another data frame into your table and apply updates, inserts, and deletes.

Additionally, we can have **Azure Synapse** **Analytics** service for data warehouses and big data systems. This service is integrated with Power BI, Machine Learning and other Azure services.

Users can export gold datasets from the delta lake to Azure Synapse using the streamlined Synapse connector. **SQL pools** in Azure Synapse provide a data warehousing and compute environment. Azure Synapse connectors provide a way to access Azure Synapse through Azure Databricks. These connectors efficiently transfer large amounts of data between Azure Databricks clusters and Azure Synapse instances.

Azure Data Lake Storage (ADLS) **backup and recovery strategy** uses a combination of techniques and features to get the best functionality and performance for an acceptable cost:

* Soft delete for containers
* Soft delete for blobs
* Resource lock on the storage account
* Delta Lake time travel
* Self-built automated backup process (copying a part of the Data Lake data to a secondary location)
* Backup ‘vault’ to store subsets of the Data Lake indefinitely
* Lifecycle management policies
* Storage account access tiers (hot/cool/archive)

For further information, about the back-up strategies go to the following link:

<https://www.moderndata.ai/2021/10/how-to-automatically-backup-your-azure-data-lakehouse/>

ADLS provides six different layers of **security**:

* Authentication
* Access control
* Network isolation
* Data protection
* Advanced threat protection
* Auditing

For further information about the security features, go to the following link:

<https://learn.microsoft.com/en-us/answers/questions/91722/azure-data-lake-storage-gen2-security-features-and.html>

Other third party open-source tools were also considered, but at the end I decided to go with ADLS Gen 2 Azure solution because of its seamless experience with Databricks. Here are few examples:

* **Amazon S3** provides an optimal foundation for a data lake because of its virtually unlimited scalability and high durability. You can seamlessly and non-disruptively increase storage from gigabytes to petabytes of content, paying only for what you use. Amazon S3 is designed to provide 99.999999999% durability.
* **Google Cloud Storage** provides globally unified, scalable and highly durable object storage. It can store data in its native format and process any variety of it, ignoring size limits.

## Processing Layer

The processing layer consists of **Azure Databricks**.

**Azure Databricks** is a data analytics platform. The platform's fully managed **Spark** clusters process large streams of data from multiple sources. Azure Databricks cleans and transforms amorphous datasets. The processed data is combined with structured data from operational databases or data warehouses. Azure Databricks also trains and deploys scalable machine learning and deep learning models.

The analytical platform collects the data from the various batch and streaming sources. Data analysts use different tools for these tasks.

* data preparation
* data exploration
* model preparation
* model training

**Managed MLflow** is built on top of MLflow, an open-source platform developed by Databricks to help manage the complete machine learning lifecycle with enterprise reliability, security and scale. Its components monitor machine learning models during training and execution and saves models and loads them in production. It manages parameter, metric, and model tracking in data science code runs. The coding options are flexible:

* Code can be in SQL, Python, R and Scala.
* The code can use popular open-source libraries and frameworks like Koalas, Pandas, and scikit-learn, which are pre-installed and optimized.
* Practitioners can optimize performance and cost with single-node and multi-node compute options.

Machine learning models are available in different formats. Azure Databricks stores information about models in the MLflow model Registry. The registry exposes models via batch, streaming, and REST APIs.

**Azure Databricks SQL Analytics** is an environment provided by Databricks, which provides a simple experience for SQL users who want to run quick ad-hoc queries on their data lake, create multiple visualization types to explore query results from different perspectives, and build and share dashboards.

Azure Databricks SQL Analytics services allow the us to have a single underlying data source to ensure consistency. This service:

* Provides a query editor and gallery, query history, basic dashboarding, and alerts.
* Uses built-in security that includes row and column level permissions.
* Uses a Photon-powered Delta engine to speed up performance.

Other tools were also considered, for example , WS offers **Elastic Map Reduce (EMR)**, which easily run and scale Apache Spark, Hive, Presto, and other big data workloads. Databricks is also available in **AWS** and **Google** cloud. But as a rule, the integrations to the rest of the Azure platform are deeper on Azure Databricks, compared to how even Databricks on AWS integrates with other AWS services. Overall, this builds a more seamless and streamlined experience for building out your data estate with Databricks.

## Serving Layer

The primary function of the serving layer is to expose the views produced by the process and analysis layer accessible to other systems or users for querying. Data or models that is easily served to consumer apps makes up the serving layer. So, the majority of this is processed data.

Databricks **Managed-MLflow model registry** exposes models via batch, streaming, and REST APIs. The solution can also deploy models to **Azure Machine Learning Web Services** or **Azure Kubernetes Service (AKS)**.

**Azure Machine Learning** is a cloud-based environment that you can use to build, deploy, and manage predictive analytics solutions.

**Power BI** generates analytical and historical reports and dashboards from the unified data platform. When working with Azure Databricks, this service uses the following features:

* A built-in Azure Databricks connector to visualize the underlying data.
* Optimized Java Database Connectivity (JDBC) drivers and ODBC (Open Database Connectivity) drivers.

## Monitoring and Governance Layer

The solution uses Azure services for collaboration, performance, reliability, governance and security:

* **Microsoft Purview** provides data discovery services, confidential data classification, and governance insights across the data estate.
* **Azure DevOps** offers continuous integration and continuous deployment (CI/CD) and other built-in version control features.
* **Azure Key Vault** securely manages secrets, keys, and certificates.
* **Azure Active Directory (Azure AD)** provides single sign-on (SSO) for Azure Databricks users. Azure Databricks supports automated user provisioning with Azure AD for these tasks:
  + Creating new users.
  + Assign an access level for each user.
  + Remove users and deny access.
* **Azure Monitor** collects and analyzes Azure resource telemetry. By proactively identifying problems, this service maximizes performance and reliability.
* **Azure Cost Management and Billing** provides financial governance services for Azure workloads.

# Conclusion

To summarize, the data lake architecture:

* The ingestion layer consists of **Azure Event Hubs** for streaming data pipelines and **Azure Data Factory** for easy data transformation (ETL/ELT) and integration with different data sources.
* The storage layer consists of **Azure Data Lake Storage (ADLS) Gen 2**, which is scalable, secure and suitable for high-performance analytical workloads. On top of it, we have **Delta Lake**, to give ACID transaction capabilities to our data lake solution. The golden dataset can be easily exported to **Azure Synapse Analytics** data warehouse for specific use-cases.
* The processing layer consists of **Azure Databricks**, which processes data using fully managed **Spark** clusters. Databricks notebooks allow you to write code in SQL, Python, R and Scala, to perform data transformation, data preparation and machine learning. **Managed MLFlow** is very well integrated inside the Databricks environment, which can be used for machine learning lifecycle management, tracking model training, saves and loading models. Additionally, **Azure Databricks SQL Analytics** provides a familiar experience to SQL user to write ad hoc queries, on the same underlying delta tables.
* The service layer consists of **Azure Machine Learning** and **Azure Kubernetes Services (AKS)** to serve the ML models. **Power BI** can be used to generate and visualize processed data reports and dashboards.
* The monitoring and governance layer consists of different Azure tools and services, like **Azure DevOps, Azure Monitor, Azure Cost Management and Billing, Azure Key Vault, Azure Active Directory, Microsoft Purview**, etc.

The next step is to determine the use cases before implementing the data lake. Prioritizing a few key use cases for the data lake over choosing the underlying technology should be the first goal. The defined use cases will assist in demonstrating some immediate benefits and the business impact of the data lake, which will be essential to sustaining project momentum and support from those higher up the chain of command.

# References

1. [Azure Event Hubs](https://azure.microsoft.com/en-us/services/event-hubs/)
2. [Azure Data Factory](https://azure.microsoft.com/services/data-factory)
3. [Azure Data Lake Storage Gen 2](https://azure.microsoft.com/services/storage/data-lake-storage)
4. [Azure Storage access tiers](https://learn.microsoft.com/en-us/azure/storage/blobs/access-tiers-overview)
5. [Delta Lake](https://delta.io/)
6. [Medallion Architecture](https://techcommunity.microsoft.com/t5/analytics-on-azure/how-to-reduce-infrastructure-costs-by-up-to-80-with-azure/ba-p/1820280)
7. [Best Practices for High Performance Data Lakes](https://www.upsolver.com/blog/best-practices-high-performance-data-lake)
8. [Guiding Principles of Modern Data Lake](https://www.upsolver.com/blog/four-principles-data-lake-architecture)
9. [Azure Modern Data Lakehouse Architecture](https://learn.microsoft.com/en-us/azure/architecture/solution-ideas/articles/azure-databricks-modern-analytics-architecture)
10. [Design Patterns for Data Lake](https://medium.com/@lackshub/design-patterns-for-data-lakes-d6da14a0af1f)
11. [Azure Synapse Analytics](https://learn.microsoft.com/de-de/azure/databricks/sql)
12. [Azure Databricks](https://azure.microsoft.com/services/databricks)
13. [Databricks Lakehouse](https://www.databricks.com/blog/2020/01/30/what-is-a-data-lakehouse.html)
14. [Spark](https://spark.apache.org/)
15. [Manage MLFlow](https://www.databricks.com/product/managed-mlflow)
16. [Azure Databricks SQL Analytics](https://learn.microsoft.com/de-de/azure/databricks/sql)
17. [Azure Machine Learning](https://azure.microsoft.com/services/machine-learning)
18. [Azure Kubernetes Services (AKS)](https://azure.microsoft.com/services/kubernetes-service)
19. [Power BI](https://learn.microsoft.com/de-de/power-bi/fundamentals/power-bi-overview)
20. [Azure DevOps](https://azure.microsoft.com/solutions/devops)
21. [Microsoft Purview](https://learn.microsoft.com/de-de/azure/purview/overview)
22. [Azure Key Vault](https://azure.microsoft.com/services/key-vault)
23. [Azure Active Directory](https://azure.microsoft.com/services/active-directory)
24. [Azure Monitor](https://azure.microsoft.com/services/monitor)
25. [Azure Cost Management and Billing](https://azure.microsoft.com/services/cost-management)
26. [ADLS Backup and Recovery](https://www.moderndata.ai/2021/10/how-to-automatically-backup-your-azure-data-lakehouse/)
27. [ADLS Security Features](https://learn.microsoft.com/en-us/answers/questions/91722/azure-data-lake-storage-gen2-security-features-and.html)
28. [Apache Nifi](https://nifi.apache.org/)
29. [Apache Kafka](https://kafka.apache.org/)
30. [Apache Pulsar](https://pulsar.apache.org/)
31. [Apache Flume](https://flume.apache.org/)
32. [Amazon S3](https://aws.amazon.com/s3/)
33. [Google Cloud Storage](https://cloud.google.com/storage)
34. [Amazon EMR](https://aws.amazon.com/emr/)