Agenda

Azure Data Lake: What, Why, and How

• Data Lake Overview & Use Cases
• Big Data in Azure
• Data Storage in Azure
• Compute in Azure
• Integrating Azure Data Lake in a Multi-Platform Architecture
• Suggestions for Getting Started with a Data Lake Project

As of Jan 2019:

Things are changing rapidly. Azure Data Lake Storage Gen2 is in public preview.
Data Lake Overview & Use Cases
What is a Data Lake?

A repository for storing large quantities of disparate sources of data in its native format.

One architectural platform to house all types of data:

- Machine-generated data (ex: IoT, logs)
- Human-generated data (ex: tweets, e-mail)
- Traditional operational data (ex: sales, inventory)
Objectives of a Data Lake

- Reduce up-front effort to ingest data
- Defer work to ‘schematize’ until value is known
- Allow time for defining business value of the data
- Store low latency data
- Access to new data types
- Facilitate advanced analytics scenarios & new use cases
- Store large volumes of data cost efficiently
Data Lake Use Cases

Ingestion of New File Types

- Preparatory file storage for multi-structured data
- Exploratory analysis to determine value of new data types & sources
- Affords additional time for longer-term planning while accumulating data or handling an influx of data

- Spatial, GPS
- Devices & Sensors
- Social Media Data
- Web Logs
- Images, Audio, Video

Data Lake

Raw Data

Exploratory Analysis

Data Science Sandbox
Data Lake Use Cases

Data Science Experimentation | Hadoop Integration

- Big data clusters
- SQL-on-Hadoop solutions
- Integrate with open source projects such as Hive, Spark, Storm, Kafka, etc.
- Sandbox solutions for initial data prep, experimentation, and analysis
- Migrate from proof of concept to operationalized solution
Data Lake Use Cases

Data Warehouse Staging Area

- ELT strategy (extract>load>transform)
- Reduce storage needs in relational platform by using the data lake as landing area
- Practical use for data stored in the data lake
- Potentially also handle data transformations in the data lake
**Data Lake Use Cases**

**Active Archiving**

- Offload aged data from data warehouse back to the data lake
- An “active archive” available for querying when needed
- Federated queries to access: current data in the DW + archive data in the data lake
Data Lake Use Cases

Lambda Architecture

- **Devices & Sensors**
- **Social Media Data**
- **Corporate Data**

**Speed Layer**
- Data Ingestion
- Data Processing

**Batch Layer**
- Data Lake: Raw Data
- Data Lake: Curated Data

**Batch Layer**
- Data Warehouse

**Serving Layer**
- Cubes & Semantic Models
- Analytics & Reporting

✓ Support for low-latency, high-velocity data in near real time
✓ Support for batch-oriented operations
Big Data in Azure
Big Data in Azure

Compute
- Azure HDInsight
- Azure Databricks
- Azure Data Lake Analytics

Storage
- Azure Storage
- Azure Data Lake Storage (Gen1)
- Azure Data Lake Storage (Gen2)*

Hadoop on a cluster of Azure virtual machines (IaaS)

*In Public Preview
Azure Data Lake

Azure Data Lake is a collection of the following services:

**Compute**
- Azure HDInsight
- Azure Data Lake Analytics

*Clusters as-a-service*
*Queries as-a-service*

**Storage**
- Azure Data Lake Storage (Gen1)
- Azure Data Lake Storage (Gen2)*

*Storage-as-a-service*

*In Public Preview*
Data Storage in Azure
Big Data in Azure: Storage

Optimized for analytics workloads

General purpose files & workloads

- Azure Storage
  - Object storage
  - Hierarchical file system
  - Multi-modal storage

- Azure Data Lake Storage (Gen1)
- Azure Data Lake Storage (Gen2)*

(Excluding relational and NoSQL data storage options)
Object-based storage manages data as discrete units.

Folders are part of the URI, but they’re merely simulated. There is no folder-level security, nor folder-specific performance optimizations.
Hierarchical file-based storage supports nesting of files within folders.

Folder-level security can be implemented, as well as certain performance optimizations.
Previously – An Either/Or Decision

Object store:
- /ATMMachineData/RawData/2018/08/CustomerATMTransactions_201808_1.csv
- /ATMMachineData/RawData/2018/08/CustomerATMTransactions_201808_2.csv

Hierarchical storage:
- ATMMachineData
  - Raw
  - 2018
    - 08
      - ‘CustomerATMTransactions_201808_1.csv’
      - ‘CustomerATMTransactions_201808_2.csv’
Deciding Between Storage Services

**Azure Storage**

- General purpose object store (containers > blobs)

**ADLS (Gen 1)**

- Hierarchical file system (folders > files)
- Optimized for analytics workloads
  - Hadoop and big data optimizations
  - Parallelized reads and writes
  - Scaled out over multiple nodes
  - Low latency writes with I/O throughput
  - Fine-grained security via access control lists

**ADLS (Gen 2)**

- Multi-modal combining features from both of the above
- Not a separate service: Azure Storage with new features
- Enable the “hierarchical namespace” (HNS) to use

Additional features not available in ADLS Gen1
- Data replication and redundancy options
- Available in all regions globally
- Hot/cold/archive tiers
- Lifecycle management (in preview at this time)
- Metadata (key/value pairs)
New Multi-Modal Storage Option: ADLS Gen 2

**The long-term vision:**
The data is stored once, and accessed through either endpoint based on use case / data access pattern. Files & folders are ‘first class citizens.’

- **ATM Machine Data**
  - **Raw**
  - **2018**
  - **08**
  - ‘CustomerATMTransactions_201808_1.csv’
  - ‘CustomerATMTransactions_201808_2.csv’

**Endpoint:**
- **object store access**
- **file system access**

---

**Azure Data Lake Storage Gen 2**
Object Store Endpoint: wasb[s]

ATM Machine Data

- Raw
- 2018
- 08

Endpoint: object store access

wasb[s]://containernname@accountname.blob.core.windows.net/raw/2018/08/CustomerATMTransactions_201808_1.csv

wasb[s]://containernname@accountname.blob.core.windows.net/raw/2018/08/CustomerATMTransactions_201808_2.csv
File System Endpoint: abfs[s]

ATM Machine Data

Filesystem:
- Raw
- 2018
- 08

Files:
- CustomerATMTransactions_201808_1.csv
- CustomerATMTransactions_201808_2.csv

Endpoints:
- abfs[s]://filesystemname@accountname.dfs.core.windows.net/raw/2018/08/CustomerATMTransactions_2018_1.csv

Abbreviations:
- afbs = Azure Blob File System
- abfss = SSL
- dfs = endpoint
Leverage partition scans & partition pruning to improve query performance:

Endpoint: file system access

Security at the folder level

Queries
Select...
From...
Where Month=01

Raw

2018

01

02

...

08

‘CustomerATMTransactions_201808_1.csv’

‘CustomerATMTransactions_201808_2.csv’
Multi-Modal Advantages with ADLS Gen 2 – Example 2

Metadata-only changes with significantly better performance using the file system endpoint:

Data ingestion

Endpoint of choice: object store access -or- file system access

Current status: This scenario is supported if everything uses file system endpoint – full interoperability is still evolving
Use different endpoints for ingestion vs. data processing:

- **Data ingestion**
  - **Endpoint:** object store access
  - **Raw**
    - 2018
      - ‘CustomerATMTransactions_201808_1.csv’
      - ‘CustomerATMTransactions_201808_2.csv’

- **Curated**

- **Data processing**
  - **Endpoint:** file system access
    - ‘CustomerATMTransactionsSummary.csv’

Current status: This is the vision from the product team, but using both interchangeably is not yet supported.
Multi-Modal Advantages with ADLS Gen 2 – Example 4

Use as the basis for Dataflows in Power BI:

ADLS Gen 2

- CDM source/subject
  - ‘CustomerEntity.csv’
  - ‘ProductEntity.csv’

CDM-compliant tables in ADLS Gen 2

Power BI Service

- Dataflow
- Dataset
- Report(s)
- Power Query Online
- Dataset
- Report(s)

Current status: Both services are in public preview so capabilities are evolving

CDM = Common Data Model
Azure Data Lake Storage Gen 2

ADLS Gen 2 = Azure Storage with the Hierarchical Namespace (HNS) enabled.

When to **Disable** the Hierarchical Namespace?

- General purpose file storage such as backups and VHDs
- Classic object store use cases which do not benefit from hierarchical storage or a high degree of organization (ex: image storage)
- Custom apps, APIs, or legacy systems which only use the Blob API and/or are unaware of file system semantics

The HNS is enabled at the storage account level. Product team says there’s no harm or performance difference (even for raw I/O) if the HNS is enabled but not used. However, you’ll pay extra cost (~30% extra) on every transaction if HNS is enabled.
Summary: Goals of ADLS Gen 2

- **Unify** the data lake story on Azure
- Take advantage of the *best of both feature sets* (object storage & hierarchical storage)
- **Multiple protocol endpoints** to allow flexibility for use cases
- **Avoid duplicating data** for specific use cases or tools (‘islands of data’)
- **Overcome limitations of object storage** (ex: metadata only operations)
- Improved *performance for big data analytics*
- Implement *full data lifecycle* and *data policies*
- **Low cost** with high-performing *throughput*
- Integrate with the new ‘*dataflows*’ functionality in Power BI
Summary: Current State of the Storage Options

Azure Storage
✓ Still a very valid option for object store workloads

ADLS (Gen 1)
✓ Fully supported in existing regions
✓ No new features

ADLS (Gen 2)
✓ In public preview
✓ Feature support is evolving over time
Demo: ADLS Gen 2
Compute in Azure
Compute Services and Data Management

Cloud Systems

- Devices & Sensors
- Social Media Data
- Third Party Data, Flat Files
- Corporate Data

Data Lake Storage

- Azure Portal
- Visual Studio Code
- Visual Studio Extensions
- Azure Data Factory
- Azure Data Lake Analytics
- Azure Databricks
- Azure Cloud Shell
- Azure SDKs (.NET, Java, Node.js, Python, etc)
- Azure CLI
- Azure Mobile App
- Azure Storage Explorer

*In Public Preview*
Big Data in Azure: Compute

- Hadoop on a cluster of Azure virtual machines (IaaS)
- Azure HDInsight (PaaS)
- Azure Databricks (PaaS)
- Azure Data Lake Analytics (Serverless)

**Higher level of complexity, control, & customization**

**Greater integration with various Apache projects**

**Easiest entry point to get started**

**Less administrative effort**

**Less integration with various Apache projects**

**Greater administrative effort**
U-SQL: Unified SQL

Blends the **declarative nature of SQL dialects + imperative nature of C#**.
Works on structured as well as **semi-structured data** without reformatting.
Distributed query support which scales and **parallelizes** across nodes.

Supported:
- Batch processing queries
- File output to ADLS Gen 1 or Blob Storage

Current status:
No integration with ADLS Gen 2 at this time (it requires the flat object store endpoint)
Azure Databricks

A unified platform for data engineering and data science activities

Cloud Systems

Devices & Sensors

Social Media Data

Third Party Data, Flat Files

Corporate Data

Azure Data Lake Storage

Azure Data Blob Storage

Spark SQL

R

Python

Scala

Azure Machine Learning

Azure Cognitive Services

Azure Data Factory

Azure SQL DB

Azure CosmosDB

SQL Server in Azure VM

Azure SQL DW
Azure HDInsight

Managed big data clusters for running open source frameworks

Cloud Systems

Devices & Sensors

Social Media Data

Third Party Data, Flat Files

Corporate Data

Azure Data Lake Storage

Azure Blob Storage

kafka

HIVE

STORM

HBASE

Spark

Azure Machine Learning

Azure Cognitive Services

Azure Data Factory

Azure SQL DB

Azure Cosmos DB

SQL Server in Azure VM

Azure SQL DW

Managed big data clusters for running open source frameworks

Cloud Systems

Devices & Sensors

Social Media Data

Third Party Data, Flat Files

Corporate Data

Azure Data Lake Storage

Azure Blob Storage

kafka

HIVE

STORM

HBASE

Spark

Azure Machine Learning

Azure Cognitive Services

Azure Data Factory

Azure SQL DB

Azure Cosmos DB

SQL Server in Azure VM

Azure SQL DW
## Deciding Between Compute Services

<table>
<thead>
<tr>
<th>Type</th>
<th>Purpose</th>
<th>Suitable for</th>
</tr>
</thead>
<tbody>
<tr>
<td>IaaS</td>
<td>Running your own cluster of Hadoop virtual machines</td>
<td>Full control over everything; investment in distributions such as Hortonworks, Cloudera, MapR</td>
</tr>
<tr>
<td>PaaS</td>
<td>Running a managed big data cluster</td>
<td>Integration with open source Apache projects and/or greater control over clusters</td>
</tr>
<tr>
<td>PaaS</td>
<td>Running a managed, optimized Spark framework</td>
<td>Collaborative notebooks; easier deployments; utilizing Spark in a variety of ways</td>
</tr>
<tr>
<td>Serverless</td>
<td>Running U-SQL batch jobs</td>
<td>Focus on running individual jobs (scripts) rather than managing a cluster</td>
</tr>
</tbody>
</table>

**1st choice**: ADLA

**Caution**
Interacting with ADLS Gen 2

- Azure Data Lake Analytics
- Custom apps, APIs, legacy systems
- Windows Azure Storage Blob Driver (wasb:)
- Current status: Blob endpoint is disabled

![Diagram of Azure Data Lake Storage (Gen 2)*](image)

- Hadoop
- Kafka
- R
- HBase
- Hive LLAP
- Spark
- Storm
- Azure HDInsight
- Azure Blob File System Driver (abfs[s]:)
- REST API interface
- Azure Data Lake Storage (Gen 2)*
- File (Blob)

- SparkSQL
- DataFrames
- MLlib
- GraphX
- SparkR
- Azure Databricks
- Azure Data Factory
- Azure Storage Explorer

*In Public Preview
Integrating Azure Data Lake in a Multi-Platform Architecture
Multi-Platform Architecture

- Handle a variety of data types & sources
- Larger data volumes at lower latency
- Bimodal: self-service + corporate BI to support all types of users
- Newer cloud services
- Advanced analytics scenarios
- Balance data integration & data virtualization
Azure Data Lake Implementation Options

Modern data warehouse

INGEST
- Logs (unstructured)
- Media (unstructured)
- Files (unstructured)
- Business/custom apps (structured)

STORE
- Azure Data Factory
- Azure Data Lake Storage

PREP & TRAIN
- Azure Databricks
- PolyBase

MODEL & SERVE
- Azure SQL Data Warehouse
- Azure Analysis Services
- Power BI

Azure Data Lake Implementation Options

Advanced analytics on big data

INGEST
- Logs (unstructured)
- Media (unstructured)
- Files (unstructured)
- Business/custom apps (structured)

STORE
- Azure Data Factory
- Azure Data Lake Storage

PREP & TRAIN
- Azure Databricks
- PolyBase

MODEL & SERVE
- Cosmos DB
- Real-time apps
- Azure SQL Data Warehouse
- Azure Analysis Services
- Power BI

Azure Data Lake Implementation Options

Real time analytics

INGEST
- Sensors and IoT (unstructured)
  - Apache Kafka for HDInsight

STORE
- Media (unstructured)
  - Logs (unstructured)
  - Files (unstructured)

PREP & TRAIN
- Business/custom apps (structured)
  - Azure Data Factory
  - Azure Data Lake Storage

MODEL & SERVE
- Azure Databricks
  - Azure Data Warehouse
  - Azure SQL
  - Cosmos DB

- Real-time apps

Suggestions for Getting Started with a Data Lake Project
Getting Started With a Data Lake Project

Is a Data Lake the Right Choice?

Do you have non-relational data?

Do you have IoT type of data?

Do you have advanced analytics scenarios on unusual datasets?

Do you need to offload ETL processing (ELT) and/or archive data from a data warehouse or other systems to low-cost storage?

Readiness:

Are you ready willing to learn different development patterns and/or new technologies?

Are you ready to handle the trade-offs of ‘schema on read’ vs ‘schema on write’?
Getting Started With a Data Lake Project

Data

What types of data ingestion pipelines do you have, at what frequency?
- Batch
- Micro-batch
- Streaming

What are the current + anticipated data size volumes, and in what format?
- Structured data
- Semi-structured data
- Unstructured data
- Geospatial data

To what extent does semi-structured data need to be integrated with the structured data?
Getting Started With a Data Lake Project

Data Movement & Storage

What level of data integration (ETL or ELT) vs. data virtualization provides optimal data access?

• Data movement can be expensive
• Data might be too large to practically move
• Time window for data processing may be small
• Latency (freshness) of data varies

Which do you value more?

• Polyglot persistence strategy (“best fit engineering”)
• Architectural simplicity

How much data movement are you willing to do?

A multi-platform architecture is more appealing if you subscribe to a polyglot persistence strategy. Success very much depends on staff skills.
Getting Started With a Data Lake Project

Information Delivery

What are the expectations + needs of your user population?
• Casual users
• Data analysts
• Data scientists
• IT, BI specialists, big data engineers

What type of data consumption do you support?
• Centralized reporting & analytics
• Decentralized self-service models
• Departmental or subject-specific data marts
• Application integration

The user base translates into expectations for how the information is to be delivered, which translates into technology choices.

The more diverse your user population is, the more likely you will have a multi-platform architecture with both schema-on-read and schema-on-write.
Getting Started With a Data Lake Project

Organizing the Data Lake
✓ Based on optimal data retrieval & security boundaries
✓ Avoid a chaotic, unorganized data swamp
✓ Take advantage of data pruning optimizations (esp year/month/day) when running queries

Common ways to organize and/or tag the data:

- **Time Partitioning**
  - Year/Month/Day/Hour/Minute

- **Subject Area**
- **Security Boundaries**
  - Department
  - Business unit
  etc...

- **Data Retention Policy**
  - Temporary data
  - Permanent data
  - Applicable period (ex: project lifetime)
  etc...

- **Business Impact / Criticality**
  - High (HBI)
  - Medium (MBI)
  - Low (LBI)
  etc...

- **Owner / Steward / SME**

- **Probability of Data Access**
  - Recent/current data
  - Historical data
  etc...

- **Confidential Classification**
  - Public information
  - Internal use only
  - Supplier/partner confidential
  - Personally identifiable information (PII)
  - Sensitive – financial
  - Sensitive – intellectual property
  etc...
Organizing the Data Lake

Although a data lake emphasizes getting started quickly, there is still up-front planning.
# Getting Started With a Data Lake Project

## Data Lake Challenges

<table>
<thead>
<tr>
<th>Technology</th>
<th>Process</th>
<th>People</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>✓ Complex, multi-layered architecture</td>
<td>✓ Right balance of deferred work vs. up-front work to minimize chaos</td>
<td>✓ Expectations &amp; trust</td>
<td>✓ Data volumes</td>
</tr>
<tr>
<td>✓ Unknown storage &amp; scalability</td>
<td>✓ Ignoring established best practices for data mgmt</td>
<td>✓ Data stewardship</td>
<td>✓ Read &amp; write performance</td>
</tr>
<tr>
<td>✓ Data retrieval</td>
<td>✓ Data quality</td>
<td>✓ Redundant effort</td>
<td>✓ Relating disparate data</td>
</tr>
<tr>
<td>✓ Working with uncurated data</td>
<td>✓ Governance</td>
<td>✓ Data engineering skillsets</td>
<td>✓ Schema changes over time</td>
</tr>
<tr>
<td>✓ Performance</td>
<td>✓ Security</td>
<td>✓ Ownership changes between teams to operationalize solutions</td>
<td>✓ Diversity of file formats &amp; types</td>
</tr>
<tr>
<td>✓ Change management</td>
<td>✓ Disaster recovery for large solutions</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Getting Started With a Data Lake Project

Always do a **proof of concept** before making a big commitment, including **file management**, **data access** & **security**.

**Data tagging & cataloging** is critical. Capture **metadata** whenever possible.

Consider the **experience level** of your staff, and the ability to support a complex solution.

Assess the impact of **open source** technologies vs. **proprietary** technologies.
Getting Started With a Data Lake Project

Cloud offerings/features/functionality are constantly changing. It is very challenging to keep up.

Your goals for going to the cloud will consistently involve trade-offs: cost, complexity, security.

Though traditional data warehousing is evolving, the concept of curated, cleansed, user-friendly data structures are still extremely relevant & needed.
Getting Started With a Data Lake Project

Organize data by *time series* whenever possible.

Consider *data access patterns* when designing the folder structure. “Pruning” of data can happen when data is set up in a well-designed hierarchy.

Make careful decisions about *file formats* you select. Every format has tradeoffs.

**File sizes:**
- There is a 4.77 TB file limit in Gen 2 (there was no specific file limit in Gen1).
- The traditional Hadoop ‘small file size’ problem still exists (though technology is evolving to help).
- The ideal, practical file size is still ~250MB - ~2GB.
Azure Data Lake: What, Why, and How

Melissa Coates
Solution Architect, BlueGranite

Download a copy of this presentation:
SQLChick.com > Presentations & Downloads page

Creative Commons License 3.0
Attribute to me as original author if you share this material
No usage of this material for commercial purposes
No derivatives or changes to this material
Appendix A

Suggestions for Continued Learning
Suggestions for Continued Learning

Azure Data Lake Developer Center: [http://azure.github.io/AzureDataLake/](http://azure.github.io/AzureDataLake/) ➤ tons of helpful links
U-SQL Center: [http://usql.io/](http://usql.io/)
U-SQL Tutorial: [https://saveenr.gitbooks.io/usql-tutorial/content/](https://saveenr.gitbooks.io/usql-tutorial/content/)

Azure Data Lake Samples & Documentation: [https://github.com/Azure/AzureDataLake/](https://github.com/Azure/AzureDataLake/)
U-SQL Samples & Documentation: [https://github.com/Azure/USQL](https://github.com/Azure/USQL)

Azure Data Lake Release Notes: [https://github.com/Azure/AzureDataLake/tree/master/docs/Release_Notes](https://github.com/Azure/AzureDataLake/tree/master/docs/Release_Notes)
Contains new features, fixes, deprecations, and breaking changes


ADL Twitter account: [https://twitter.com/azuredatalake](https://twitter.com/azuredatalake)
ADL team member Twitter accounts: [https://twitter.com/MikeDoesBigData](https://twitter.com/MikeDoesBigData) + [https://twitter.com/saveenr](https://twitter.com/saveenr)
Suggestions for Continued Learning (2/4)


ADLA Saving Money & Controlling Costs: https://onedrive.live.com/?authkey=%21AHceDeuGX5PKbVw&id=3BDE3286AB2E59F7%211005&cid=3BDE3286AB2E59F7


Creating External Data Sources for PolyBase and Elastic Queries: https://docs.microsoft.com/en-us/sql/t-sql/statements/create-external-data-source-transact-sql

Suggestions for Continued Learning

**Blog Posts**


Suggestions for Continued Learning

**Video**

What’s New with Azure Data Lake Storage Gen 2
https://www.youtube.com/watch?v=DJkFSpis2B0

**E-Book**

Data Lakes in a Modern Data Architecture
Appendix B
Terms & Definitions
<table>
<thead>
<tr>
<th><strong>Definitions</strong></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Warehouse</td>
<td>Repository of data from multiple sources, cleansed &amp; enriched for reporting; generally ‘schema on write’</td>
</tr>
<tr>
<td>Data Lake</td>
<td>Repository of data for multi-structured data; generally ‘schema on read’</td>
</tr>
<tr>
<td>Hadoop</td>
<td>(1) Data storage via HDFS (Hadoop Distributed File System), and (2) Set of Apache projects for data processing and analytics</td>
</tr>
<tr>
<td>Lambda Architecture</td>
<td>Data processing &amp; storage with batch, speed, and serving layers</td>
</tr>
<tr>
<td>ETL</td>
<td>Extract &gt; Transform &gt; Load: traditional paradigm associated with data warehousing and ‘schema on write’</td>
</tr>
<tr>
<td>ELT</td>
<td>Extract &gt; Load &gt; Transform: newer paradigm associated with data lakes &amp; ‘schema on read’</td>
</tr>
<tr>
<td>Semantic Model</td>
<td>User-friendly interface for users on top of a data warehouse and/or data lake</td>
</tr>
</tbody>
</table>
### Definitions

<table>
<thead>
<tr>
<th>Term</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Data Integration</strong></td>
<td>Physically moving data to integrate multiple sources together</td>
</tr>
<tr>
<td><strong>Data Virtualization</strong></td>
<td>Access to one or more distributed data sources without requiring the data to be physically materialized in another data structure</td>
</tr>
<tr>
<td><strong>Federated Query</strong></td>
<td>A type of data virtualization: access &amp; consolidate data from multiple distributed data sources</td>
</tr>
<tr>
<td><strong>Polyglot Persistence</strong></td>
<td>A multi-platform strategy which values using the most effective technology based on the data itself (“best fit engineering”)</td>
</tr>
<tr>
<td><strong>Schema on Write</strong></td>
<td>Data structure is applied at design time, requiring additional up-front effort to formulate a data model (relational DBs)</td>
</tr>
<tr>
<td><strong>Schema on Read</strong></td>
<td>Data structure is applied at query time rather than when the data is initially stored (data lakes, NoSQL)</td>
</tr>
</tbody>
</table>

Definitions

- **Shared Infrastructure (Lower Cost)**
  - **Infrastructure as a Service (IaaS)**
    - SQL Server in a VM, Hadoop in a VM
  - **Platform as a Service (Paas)**
    - Azure SQL DB, Azure SQL DW, Azure HDInsight, Azure Data Lake Storage
  - **Software as a Service (SaaS)**
    - Power BI, Office 365, Azure Data Lake Analytics

- **Dedicated Infrastructure (Higher Cost of Ownership)**
  - **On-Premises**
    - **Physical Server**
    - **Virtual Server**
  - **Azure Stack (Private Cloud)**

- **Easier to Scale**
- **More Difficult to Scale**

- **More Control (Higher Administration Effort)**
- **Less Control (Lower Administration Effort)**