[**Azure Data Factory**](https://intellipaat.com/blog/azure-data-factory-tutorial/) is a cloud-based Microsoft tool that collects raw business data and further transforms it into usable information. It is a data integration ETL (extract, transform, and load) service that automates the transformation of the given raw data.

# **Execution and triggers in Azure Data Factory for Historical Data in Azure Data Factory**

Triggers are like job schedulers for the execution of pipeline runs in the Azure Data Factory. Presently there are three types of triggers that are supported in ADF.

1. **Schedule trigger**: A trigger that executes a pipeline on an absolute schedule
2. **Tumbling window trigger**: A trigger that operates at periodic intervals and also retains state
3. **Event-based trigger**: A trigger that responds based on events

In this blog, we will discuss the tumbling window trigger and how it supports fetching historical data in the Azure Data Factory.

### Tumbling Window Trigger

A typical ETL Package is built to compute data from a point in time forward. So, historical data will not be loaded from source to target and a separate load is required between the datasets. The process of adding missing data from the past to the target is termed as Historical Data Collection or Data Backfilling. In traditional ETL, backfilling of data requires enormous amounts of manual work and time to build effective SQL scripts. Microsoft has provided a feature named Tumbling Window Trigger, which is primarily designed for fetching historical data using Azure Data Factory. A tumbling window trigger will fire in a sequence of non-overlapping and contiguous periodic time intervals from a specified start time while also retaining state.

**1. Why do we need Azure Data Factory?**

* The amount of data generated these days is huge and this data comes from different sources. When we move this particular data to the cloud, there are few things needed to be taken care of.
* Data can be in any form as it comes from different sources and these different sources will transfer or channelize the data in different ways and it can be in a different format. When we bring this data to the cloud or particular storage we need to make sure that this data is well managed. i.e you need to transform the data, delete unnecessary parts. As per moving the data is concerned, we need to make sure that data is picked from different sources and bring it at one common place then store it and if required we should transform into more meaningful.
* This can be also done by traditional data warehouse as well but there are certain disadvantages. Sometimes we are forced to go ahead and have custom applications that deal with all these processes individually which is time-consuming and integrating all these sources is a huge pain. we need to figure out a way to automate this process or create proper workflows.
* Data factory helps to orchestrate this complete process into more manageable or organizable manner.

# Integration runtime in Azure Data Factory

The Integration Runtime (IR) is the compute infrastructure used by Azure Data Factory to provide the following data integration capabilities across different network environments:

* **Data Flow**: Execute a [Data Flow](https://docs.microsoft.com/en-us/azure/data-factory/concepts-data-flow-overview) 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.

In Data Factory, an activity defines the action to be performed. A linked service defines a target data store or a compute service. An integration runtime provides the bridge between the activity and linked Services. It's referenced by the linked service or activity, 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 to the target data store or compute service in the most performant way while meeting security and compliance needs.

**2. What is Azure Data Factory?**

Cloud-based integration service that allows creating data-driven workflows in the cloud for orchestrating and automating data movement and data transformation.

* Using Azure data factory, you can create and schedule the data-driven workflows(called pipelines) that can ingest data from disparate data stores.
* It can process and transform the data by using compute services such as HDInsight Hadoop, Spark, Azure Data Lake Analytics, and Azure Machine Learning.

**3. What is the integration runtime?**

* The integration runtime is the compute infrastructure that Azure Data Factory uses to provide the following data integration capabilities across various network environments.
* 3 Types of integration runtimes:
* **Azure Integration Run Time:** Azure Integration Run Time can copy data between cloud data stores and it can dispatch the activity to a variety of compute services such as Azure HDinsight or SQL server where the transformation takes place
* **Self Hosted Integration Run Time:**Self Hosted Integration Run Time is software with essentially the same code as Azure Integration Run Time. But you install it on an on-premise machine or a virtual machine in a virtual network. A Self Hosted IR can run copy activities between a public cloud data store and a data store in a private network. It can also dispatch transformation activities against compute resources in a private network. We use Self Hosted IR because Data factory will not be able to directly access on-primitive data sources as they sit behind a firewall.It is sometimes possible to establish a direct connection between Azure and on-premises data sources by configuring the firewall in a specific way if we do that we don’t need to use a self-hosted IR.
* **Azure SSIS Integration Run Time:**With SSIS Integration Run Time, you can natively execute SSIS packages in a managed environment. So when we lift and shift the SSIS packages to data factory, we use Azure SSIS Integration Run TIme.

## What are mapping data flows?

Mapping data flows are visually designed data transformations in Azure Data Factory. Data flows allow data engineers to develop data transformation logic without writing code. The resulting data flows are executed as activities within Azure Data Factory pipelines that use scaled-out Apache Spark clusters. Data flow activities can be operationalized using existing Azure Data Factory scheduling, control, flow, and monitoring capabilities.

Mapping data flows provide an entirely visual experience with no coding required. Your data flows run on ADF-managed execution clusters for scaled-out data processing. Azure Data Factory handles all the code translation, path optimization, and execution of your data flow jobs.

**4. What is the limit on the number of integration runtimes?**

There is no hard limit on the number of integration runtime instances you can have in a data factory. There is, however, a limit on the number of VM cores that the integration runtime can use per subscription for SSIS package execution.

**5. What is the difference between Azure Data Lake and Azure Data Warehouse?**

Data Warehouse is a traditional way of storing data which is still used widely. Data Lake is complementary to Data Warehouse i.e if you have your data at a data lake that can be stored in data warehouse as well but there are certain rules that need to be followed.

|  |  |
| --- | --- |
| **DATA LAKE** | **DATA WAREHOUSE** |
| Complementary to data warehouse | Maybe sourced to the data lake |
| Data is Detailed data or Raw data. It can be in any particular form.you just need to take the data and dump it into your data lake | Data is filtered, summarised,refined |
| Schema on read (not structured, you can define your schema in n number of ways) | Schema on write(data is written in Structured form or in a particular schema) |
| One language to process data of any format(USQL) | It uses SQL |

**Intermediate Interview Questions**

**6. What is blob storage in Azure?**

[Azure Blob Storage](https://azure.microsoft.com/en-us/services/storage/blobs/) is a service for storing large amounts of unstructured object data, such as text or binary data. You can use Blob Storage to expose data publicly to the world or to store application data privately. Common uses of Blob Storage include:

* Serving images or documents directly to a browser
* Storing files for distributed access
* Streaming video and audio
* Storing data for backup and restore disaster recovery, and archiving
* Storing data for analysis by an on-premises or Azure-hosted service

**7. What is the difference between Azure Data Lake store and Blob storage?**

|  |  |  |
| --- | --- | --- |
|  | **Azure Data Lake Storage Gen1** | **Azure Blob Storage** |
| **Purpose** | Optimized storage for big data analytics workloads | General purpose object store for a wide variety of storage scenarios, including big data analytics |
| **Structure** | Hierarchical file system | Object store with flat namespace |
| **Key Concepts** | Data Lake Storage Gen1 account contains folders, which in turn contains data stored as files | Storage account has containers, which in turn has data in the form of blobs |
| **Use Cases** | Batch, interactive, streaming analytics and machine learning data such as log files, IoT data, click streams, large datasets | Any type of text or binary data, such as application back end, backup data, media storage for streaming and general purpose data. Additionally, full support for analytics workloads; batch, interactive, streaming analytics and machine learning data such as log files, IoT data, click streams, large datasets |
| **Server-side API** | [WebHDFS-compatible REST API](https://msdn.microsoft.com/library/azure/mt693424.aspx) | [Azure Blob Storage REST API](https://msdn.microsoft.com/library/azure/dd135733.aspx) |
| **Data Operations – Authentication** | Based on [Azure Active Directory Identities](https://docs.microsoft.com/en-us/azure/active-directory/develop/authentication-scenarios) | Based on shared secrets – [Account Access Keys](https://docs.microsoft.com/en-us/azure/storage/common/storage-account-manage#access-keys) and [Shared Access Signature Keys](https://docs.microsoft.com/en-us/azure/storage/common/storage-dotnet-shared-access-signature-part-1). |

**8. What are the steps for creating ETL process in Azure Data Factory?**

While we are trying to extract some data from Azure SQL server database, if something has to be processed, then it will be processed and is stored in the Data Lake Store.

**Steps for Creating ETL**

* Create a Linked Service for source data store which is SQL Server Database
* Assume that we have a cars dataset
* Create a Linked Service for destination data store which is Azure Data Lake Store
* Create a dataset for Data Saving
* Create the pipeline and add copy activity
* Schedule the pipeline by adding a trigger

**9. What is the difference between HDinsight & Azure Data Lake Analytics?**

|  |  |
| --- | --- |
| **HDInsight(PaaS)** | **ADLA(SaaS)** |
| HDInsight is Platform as a service | Azure Data Lake Analytics is Software as a service. |
| If we want to process a data set, first of all, we have to configure the cluster with predefined nodes and then we use a language like pig or hive for processing data | It is all about passing query, written for processing data and Azure Data Lake Analytics will create necessary compute nodes as per our instruction on demand and process the data set |
| Since we configure the cluster with HD insight, we can create as we want and we can control it as we want. All Hadoop subprojects such as spark, kafka can be used without any limitation. | With azure data lake analytics, it does not give much flexibility in terms of the provision in the cluster, but Azure takes care of it. We don’t need to worry about cluster creation. The assignment of nodes will be done based on the instruction we pass. In addition to that, we can make use of USQL taking advantage of dotnet for processing data. |

**10. What are the top-level concepts of Azure Data Factory?**

* **Pipeline:**It acts as a carrier in which we have various processes taking place.

This individual process is an activity.

* **Activities:**Activities represent the processing steps in a pipeline. A pipeline can have one or multiple activities. It can be anything i.e process like querying a data set or moving the dataset from one source to another.
* **Datasets:**Sources of data. In simple words, it is a data structure that holds our data.
* **Linked services**: These store information that is very important when it comes to connecting an external source.

For example: Consider SQL server, you need a connection string that you can connect to an external device. you need to mention the source and the destination of your dat

**11. How can I schedule a pipeline?**

* You can use the scheduler trigger or time window trigger to schedule a pipeline.
* The trigger uses a wall-clock calendar schedule, which can schedule pipelines periodically or in calendar-based recurrent patterns (for example, on Mondays at 6:00 PM and Thursdays at 9:00 PM).

**12. Can I pass parameters to a pipeline run?**

* Yes, parameters are a first-class, top-level concept in Data Factory.
* You can define parameters at the pipeline level and pass arguments as you execute the pipeline run on demand or by using a trigger.

**13. Can I define default values for the pipeline parameters?**

You can define default values for the parameters in the pipelines.

**14. Can an activity in a pipeline consume arguments that are passed to a pipeline run?**

Each activity within the pipeline can consume the parameter value that’s passed to the pipeline and run with the @parameter construct.

**15. Can an activity output property be consumed in another activity?**

An activity output can be consumed in a subsequent activity with the @activity construct.

**16. How do I gracefully handle null values in an activity output?**

You can use the @coalesce construct in the expressions to handle the null values gracefully.

**17. Which Data Factory version do I use to create data flows?**

Use the Data Factory V2 version to create data flows.

**18. What has changed from private preview to limited public preview in regard to data flows?**

* You will no longer have to bring your own Azure Databricks clusters.
* Data Factory will manage cluster creation and tear-down.
* Blob datasets and Azure Data Lake Storage Gen2 datasets are separated into delimited text and Apache Parquet datasets.
* You can still use Data Lake Storage Gen2 and Blob storage to store those files. Use the appropriate linked service for those storage engines.

**19. How do I access data by using the other 80 dataset types in Data Factory?**

* The Mapping Data Flow feature currently allows Azure SQL Database, Azure SQL Data Warehouse, delimited text files from Azure Blob storage or Azure Data Lake Storage Gen2, and Parquet files from Blob storage or Data Lake Storage Gen2 natively for source and sink.
* Use the Copy activity to stage data from any of the other connectors, and then execute a Data Flow activity to transform data after it’s been staged. For example, your pipeline will first copy into Blob storage, and then a Data Flow activity will use a dataset in source to transform that data.

**20. Explain the two levels of security in ADLS Gen2?**

The two levels of security applicable to ADLS Gen2 were also in effect for ADLS Gen1. Even though this is not new, it is worth calling out the two levels of security because it’s a very fundamental piece to getting started with the data lake and it is confusing for many people just getting started.

* Role-Based Access Control (RBAC). RBAC includes built-in Azure roles such as reader, contributor, owner or custom roles. Typically, RBAC is assigned for two reasons. One is to specify who can manage the service itself (i.e., update settings and properties for the storage account). Another reason is to permit the use of built-in data explorer tools, which require reader permissions.
* Access Control Lists (ACLs). Access control lists specify exactly which data objects a user may read, write, or execute (execute is required to browse the directory structure). ACLs are POSIX-compliant, thus familiar to those with a Unix or Linux background.

POSIX does not operate on a security inheritance model, which means that access ACLs are specified for every object. The concept of default ACLs is critical for new files within a directory to obtain the correct security settings, but it should not be thought of as inheritance. Because of the overhead assigning ACLs to every object, and because there is a limit of 32 ACLs for every object, it is extremely important to manage data-level security in ADLS Gen1 or Gen2 via Azure Active Directory groups

6. What is the difference between SaaS, PaaS, and IaaS?

This is one of the most common Azure interview questions. Cloud Computing has three types of service models, that are IaaS, PaaS, and SaaS

|  |  |  |
| --- | --- | --- |
| Infrastructure as a Service(IaaS) | Platform as a Service(PaaS) | Software as a Service(SaaS) |
| It provides users with components such as OS, networking capabilities, etc. This is a paid service, based on usage and can be used to host applications. | It enables developers to build and work with applications without having to worry about the infrastructure or management of the hosting environment. | It involves applications being consumed and used by organizations. Usually, organizations pay for their use of the application |
| Example -  Azure Virtual Machine, Azure VNET | Example -  Azure SQL, Azure Storage | Example -  Office 365, Salesforce |

7. What are the instance types offered by Azure?

Azure offers a number of different instance types based on what needs they fulfill.

* General Purpose - CPU to memory ratio is balanced. Provides low to medium traffic web servers, small to medium databases and is ideal for testing and development

Largest instance size: Standard\_D64\_v3

256 GB Memory and 1600 GB SSD Temp Storage

* Compute Optimized - High CPU to memory ratio. Best suited for medium traffic web servers, application servers, batch processes, and network appliances

Largest instance size: Standard\_F72s\_V2

144 GB Memory and 576 GB SSD Temp Storage

* Memory-Optimized - High memory to CPU ratio. Best suited for relational database servers, in-memory analytics, and medium to large caches

Largest instance size: Standard\_M128m

3892 GB Memory and 14,336 GB SSD Temp Storage

* Storage Optimized - Provides high disk IO and throughput. Best suited for Big Data, NoSQL and SQL Databases

Largest instance size: Standard\_L32s

256 GB Memory and 5630 GB SSD Temp Storage

* GPU - Virtual Machines that specialize in heavy graphic rendering and video editing. It also helps with model training and inferencing with deep learning

Largest instance size: Standard\_ND24rs

448 GB Memory and 2948 GB SSD Temp Storage  
4 GPUs and 96 GB Memory

* High-Performance Compute - Provides Azure’s fastest and powerful CPU virtual machines with optional high throughput interfaces

Largest instance size: Standard\_L32s

224 GB Memory and 2000 GB SSD Temp Storage

**Azure Blob storage/ ADLS Gen1 /ADLS gen 2**

# **Azure Blob storage**

[Azure Blob storage](https://azure.microsoft.com/services/storage/blobs/) is a service for storing large amounts of unstructured object data, such as text or binary data. You can use Blob storage to expose data publicly to the world, or to store application data privately. Common uses of Blob storage include:

* Serving images or documents directly to a browser
* Storing files for distributed access
* Streaming video and audio
* Storing data for backup and restore, disaster recovery, and archiving
* Storing data for analysis by an on-premises or Azure-hosted service

### Mount an Azure Blob storage container

To mount a Blob storage container or a folder inside a container, use the following command:

dbutils.fs.mount(

source = "wasbs://<container-name>@<storage-account-name>.blob.core.windows.net",

mount\_point = "/mnt/<mount-name>",

extra\_configs = {"<conf-key>":dbutils.secrets.get(scope = "<scope-name>", key = "<key-name>")})

where

* + <storage-account-name> is the name of your Azure Blob storage account.
  + <container-name> is the name of a container in your Azure Blob storage account.
  + <mount-name> is a DBFS path representing where the Blob storage container or a folder inside the container (specified in source) will be mounted in DBFS.
  + <conf-key> can be either fs.azure.account.key.<storage-account-name>.blob.core.windows.net or fs.azure.sas.<container-name>.<storage-account-name>.blob.core.windows.net
  + dbutils.secrets.get(scope = "<scope-name>", key = "<key-name>") gets the key that has been stored as a [secret](https://docs.databricks.com/security/secrets/secrets.html) in a [secret scope](https://docs.databricks.com/security/secrets/secret-scopes.html).

1. Access files in your container as if they were local files, for example:

df = spark.read.text("/mnt/<mount-name>/...")

df = spark.read.text("dbfs:/<mount-name>/...")

### Unmount a mount point

dbutils.fs.unmount("/mnt/<mount-name>")

## Access Azure Blob storage directly

This section explains how to access Azure Blob storage using the Spark DataFrame API, the RDD API, and the Hive client.

### Access Azure Blob storage using the DataFrame API

### You need to configure credentials before you can access data in Azure Blob storage, either as session credentials or cluster credentials.

* Set up an account access key:

spark.conf.set(

"fs.azure.account.key.<storage-account-name>.blob.core.windows.net",

"<storage-account-access-key>")

**val** df **=** spark.read.parquet("wasbs://<container-name>@<storage-account-name>.blob.core.windows.net/<directory-name>")

# **Azure Data Lake Storage Gen**

# [Azure Data Lake Storage Gen1](https://docs.microsoft.com/azure/data-lake-store/) (formerly Azure Data Lake Store, also known as ADLS) is an enterprise-wide hyper-scale repository for big data analytic workloads. Azure Data Lake Storage Gen1 enables you to capture data of any size, type, and ingestion speed in a single place for operational and exploratory analytics. Azure Data Lake Storage Gen1 is specifically designed to enable analytics on the stored data and is tuned for performance for data analytics scenarios.

There are two ways of accessing Azure Data Lake Storage Gen1:

1. Mount an Azure Data Lake Storage Gen1 filesystem to DBFS using a service principal and OAuth 2.0.
2. Use a service principal directly.

Create and grant permissions to service principal

If your selected access method requires a service principal with adequate permissions, and you do not have one, follow these steps:

1. [Create an Azure AD application and service principal that can access resources](https://docs.microsoft.com/azure/azure-resource-manager/resource-group-create-service-principal-portal). Note the following properties:
   * application-id: An ID that uniquely identifies the client application.
   * directory-id: An ID that uniquely identifies the Azure AD instance.
   * service-credential: A string that the application uses to prove its identity.
2. Register the service principal, granting the correct [role assignment](https://docs.microsoft.com/azure/data-lake-store/data-lake-store-secure-data), such as Contributor, on the Azure Data Lake Storage Gen1 account.

## Mount Azure Data Lake Storage Gen1 resource using a service principal and OAuth 2.0

You can mount an Azure Data Lake Storage Gen1 resource or a folder inside it to [Databricks File System (DBFS)](https://docs.databricks.com/data/databricks-file-system.html). The mount is a pointer to data lake storage, so the data is never synced locally.

### Mount Azure Data Lake Storage Gen1 resource or folder

configs = {"<prefix>.oauth2.access.token.provider.type": "ClientCredential",

"<prefix>.oauth2.client.id": "<application-id>",

"<prefix>.oauth2.credential": dbutils.secrets.get(scope = "<scope-name>", key = "<key-name-for-service-credential>"),

"<prefix>.oauth2.refresh.url": "https://login.microsoftonline.com/<directory-id>/oauth2/token"}

# Optionally, you can add <directory-name> to the source URI of your mount point.

dbutils.fs.mount(

source = "adl://<storage-resource>.azuredatalakestore.net/<directory-name>",

mount\_point = "/mnt/<mount-name>",

extra\_configs = configs)

where

* <prefix> is fs.adl for Databricks Runtime 6.0 and above and dfs.adls for Databricks Runtime 5.5 LTS.

**Warning**

For Databricks Runtime 6.0 and above, the dfs.adls. prefix for Azure Data Lake Storage Gen1 configuration keys has been deprecated in favor of the prefix fs.adl.

Access files in your container as if they were local files, for examle

df = spark.read.text("/mnt/**%s**/...." % <mount-name>)

df = spark.read.text("dbfs:/<mount-name>/....")

### Unmount a mount point

dbutils.fs.unmount("/mnt/<mount-name>")

## Access directly with Spark APIs using a service principal and OAuth 2.0

You can access an Azure Data Lake Storage Gen1 storage account directly (as opposed to mounting with DBFS) with OAuth 2.0 using the service principal.

### Access using the DataFrame API

To read from your Azure Data Lake Storage Gen1 account, you can configure Spark to use service credentials with the following snippet in your notebook:

Python

Copy

spark.conf.set("<prefix>.oauth2.access.token.provider.type", "ClientCredential")

spark.conf.set("<prefix>.oauth2.client.id", "<application-id>")

spark.conf.set("<prefix>.oauth2.credential", dbutils.secrets.get(scope = "<scope-name>", key = "<key-name-for-service-credential>"))

spark.conf.set("<prefix>.oauth2.refresh.url", "https://login.microsoftonline.com/<directory-id>/oauth2/token")

where

* <prefix> is fs.adl for Databricks Runtime 6.0 and above and dfs.adls for Databricks Runtime 5.5 LTS.

**Warning**

For Databricks Runtime 6.0 and above, the dfs.adls. prefix for Azure Data Lake Storage Gen1 configuration keys has been deprecated in favor of the prefix fs.adl.

* dbutils.secrets.get(scope = "<scope-name>", key = "<key-name>") retrieves your storage account access key that has been stored as a [secret](https://docs.databricks.com/security/secrets/secrets.html) in a [secret scope](https://docs.databricks.com/security/secrets/secret-scopes.html).

Once your credentials are set up, you can use standard Spark and Databricks APIs to read from the resource. For example:

Scala

Copy

**val** df **=** spark.read.parquet("adl://<storage-resource>.azuredatalakestore.net/<directory-name>")

dbutils.fs.ls("adl://<storage-resource>.azuredatalakestore.net/<directory-name>")

# **Azure Data Lake Storage Gen2**

[Azure Data Lake Storage Gen2](https://docs.microsoft.com/azure/storage/data-lake-storage/introduction) (also known as ADLS Gen2) is a next-generation [data lake](https://databricks.com/discover/data-lakes/introduction) solution for big data analytics. Azure Data Lake Storage Gen2 builds Azure Data Lake Storage Gen1 capabilities—file system semantics, file-level security, and scale—into Azure Blob storage, with its low-cost tiered storage, high availability, and disaster recovery features.

There are several ways of accessing Azure Data Lake Storage Gen2:

* Mount an Azure Data Lake Storage Gen2 filesystem to DBFS using a service principal and OAuth 2.0.
* Use a service principal directly.
* Use an Azure storage shared access signature (SAS) token provider.
* Use the Azure Data Lake Storage Gen2 storage account access key directly.

[Create an Azure AD application and service principal that can access resources](https://docs.microsoft.com/azure/azure-resource-manager/resource-group-create-service-principal-portal). Note the following properties:

* + application-id: An ID that uniquely identifies the application.
  + directory-id: An ID that uniquely identifies the Azure AD instance.
  + storage-account-name: The name of the storage account.
  + service-credential: A string that the application uses to prove its identity.

1. Register the service principal, granting the correct [role assignment](https://docs.microsoft.com/azure/storage/common/storage-auth-aad-rbac-portal?toc=%2fazure%2fstorage%2fblobs%2ftoc.json), such as Storage Blob Data Contributor, on the Azure Data Lake Storage Gen2 account.

## Mount an Azure Data Lake Storage Gen2 account using a service principal and OAuth 2.0

You can mount an Azure Data Lake Storage Gen2 account to DBFS, authenticating using a service principal and OAuth 2.0. The mount is a pointer to data lake storage, so the data is never synced locally.

**Important**

* Mounting an Azure Data Lake Storage Gen2 is supported only using OAuth credentials. Mounting with an account access key is not supported. **Mount Azure Data Lake Storage Gen2 filesystem**

1. To mount an Azure Data Lake Storage Gen2 filesystem or a folder inside it, use the following command:

configs = {"fs.azure.account.auth.type": "OAuth",

"fs.azure.account.oauth.provider.type": "org.apache.hadoop.fs.azurebfs.oauth2.ClientCredsTokenProvider",

"fs.azure.account.oauth2.client.id": "<application-id>",

"fs.azure.account.oauth2.client.secret": dbutils.secrets.get(scope="<scope-name>",key="<service-credential-key-name>"),

"fs.azure.account.oauth2.client.endpoint": "https://login.microsoftonline.com/<directory-id>/oauth2/token"}

# Optionally, you can add <directory-name> to the source URI of your mount point.

dbutils.fs.mount(

source = "abfss://<file-system-name>@<storage-account-name>.dfs.core.windows.net/",

mount\_point = "/mnt/<mount-name>",

extra\_configs = configs)

1. Access files in your Azure Data Lake Storage Gen2 filesystem as if they were files in DBFS; for example:

df = spark.read.text("/mnt/**%s**/...." % <mount-name>)

df = spark.read.text("dbfs:/mnt/<mount-name>/....")

### Unmount a mount point

dbutils.fs.unmount("/mnt/<mount-name>")

## Access directly with service principal and OAuth 2.0

You can access an Azure Data Lake Storage Gen2 storage account directly (as opposed to mounting with DBFS) with OAuth 2.0 using the service principal. You can directly access any Azure Data Lake Storage Gen2 storage account that the service principal has permissions on. You can add multiple storage accounts and service principals in the same Spark session.

### Set credentials

The way you set credentials depends on which API you plan to use when accessing Azure Data Lake Storage Gen2: DataFrame, Dataset, or RDD.

#### **DataFrame or DataSet API**

If you are using Spark DataFrame or Dataset APIs, we recommend that you set your account credentials in your notebook’s session configs:

Scala

Copy

spark.conf.set("fs.azure.account.auth.type.<storage-account-name>.dfs.core.windows.net", "OAuth")

spark.conf.set("fs.azure.account.oauth.provider.type.<storage-account-name>.dfs.core.windows.net", "org.apache.hadoop.fs.azurebfs.oauth2.ClientCredsTokenProvider")

spark.conf.set("fs.azure.account.oauth2.client.id.<storage-account-name>.dfs.core.windows.net", "<application-id>")

spark.conf.set("fs.azure.account.oauth2.client.secret.<storage-account-name>.dfs.core.windows.net", dbutils.secrets.get(scope="<scope-name>",key="<service-credential-key-name>"))

spark.conf.set("fs.azure.account.oauth2.client.endpoint.<storage-account-name>.dfs.core.windows.net", "https://login.microsoftonline.com/<directory-id>/oauth2/token")

where dbutils.secrets.get(scope="<scope-name>",key="<service-credential-key-name>") retrieves your service credential that has been stored as a [secret](https://docs.databricks.com/security/secrets/secrets.html) in a [secret scope](https://docs.databricks.com/security/secrets/secret-scopes.html).

Once your credentials are set up, you can use standard Spark and Databricks APIs to read from the storage account. For example:

**val** df **=** spark.read.parquet("abfss://<file-system-name>@<storage-account-name>.dfs.core.windows.net/<directory-name>")

dbutils.fs.ls("abfss://<file-system-name>@<storage-account-name>.dfs.core.windows.net/<directory-name>")

# **Azure Key Vault basic**

Azure Key Vault is a cloud service for securely storing and accessing secrets. A secret is anything that you want to tightly control access to, such as API keys, passwords, certificates, or cryptographic keys. Key Vault service supports two types of containers

Azure Key Vault helps solve the following problems:

* **Secrets Management** - Azure Key Vault can be used to Securely store and tightly control access to tokens, passwords, certificates, API keys, and other secrets
* **Key Management** - Azure Key Vault can also be used as a Key Management solution. Azure Key Vault makes it easy to create and control the encryption keys used to encrypt your data.
* **Certificate Management** - Azure Key Vault is also a service that lets you easily provision, manage, and deploy public and private Transport Layer Security/Secure Sockets Layer (TLS/SSL) certificates for use with Azure and your internal connected resources.

# **Azure Data Explorer (Kusto) when touse or not**

Let’s talk about [Azure Data Explorer](https://docs.microsoft.com/en-us/azure/data-explorer/data-explorer-overview) (ADX ) also known as Kusto.

If you ask me that is the best kept secret in Azure.

Well, it isn’t exactly a secret but most people do not know about it or if they do, they just think of it as the back-end engine behind [Azure Monitor](https://docs.microsoft.com/en-us/azure/azure-monitor/overview).

ADX is an [Azure Analytics Service](https://azure.microsoft.com/en-us/services/#analytics). It is great at analyzing large volume of near real time telemetry such as logs and IoT.

Isn’t that what [Azure Datawarehouse](https://vincentlauzon.com/2016/07/31/how-does-azure-data-warehouse-scale/) is supposed to do? Or [Azure Databricks](https://vincentlauzon.com/2017/12/18/azure-databricks-getting-started/)?In this article, I’ll go around characteristics of the service: what its strength are and where it is complemented by other services.

I started with a huge essay trying to cover every aspects but I was bored writing it so I guess it wouldn’t have been very exciting to reading material. I went with a much lighter version. I’ll explore it further in future articles.

## Scale & Performance

The [online documentation](https://docs.microsoft.com/en-us/azure/data-explorer/data-explorer-overview#what-makes-azure-data-explorer-unique) says it scales to terabytes of data in minutes.

That is true but it is also true of many distributed data services.

The uniqueness comes in what we can do at that scale.

At heart Azure Data Explorer (ADX) is about… Data Exploration. It is a real challenge to explore data at the Terabyte scale with little data preparation, i.e. no defined indexes & no pre-computed aggregations.

* ADX is a very fast engine that can perform **ad hoc queries**, such as aggregations on high volume of data in a few seconds, often in less than a second. This enables a truly interactive experience where each query teaches us something about the data and leads us to the next question we want to ask.
* Aggregations are fine but aggregating over huge data set often result in losing information. It is akin to looking at a forest and be told it is made of wood. Accurate but not insightful. One of ADX’ strength is [Time Series](https://en.wikipedia.org/wiki/Time_series) [analysis](https://docs.microsoft.com/en-us/azure/data-explorer/time-series-analysis). This allows us to take a use data set and split it in multiple series and analyse those series separately or as a whole.
* ADX can perform some [Machine Learning](https://docs.microsoft.com/en-us/azure/data-explorer/machine-learning-clustering) (ML) algorithm through a big data sets or use a pre-trained model (e.g. in Python) and use the prediction of a model in the queries.
* ADX can process **structured, semi-structured & unstructured** at great speed. Although other solutions can go through small sets of JSON or text fields, they usually take minutes to process large data sets. ADX answers in seconds.
* [Visualisation](https://docs.microsoft.com/en-us/azure/data-explorer/viz-overview) is part of the exploration and can be challenging in huge data sets. Visualization is part of the query language. It seems odd at first but turns out to be very productive.
* The query language, [Kusto](https://docs.microsoft.com/en-us/azure/data-explorer/write-queries), is unique to ADX. This also seems odd at first: isn’t SQL the perfect language to query data? It turns out Kusto is way more productive than SQL for analytics. For someone proficient in SQL it takes 1-3 hours to become a Kusto query language expert. In addition, ADX also supports [TSQL](https://docs.microsoft.com/en-us/azure/kusto/api/tds/t-sql).

## Near real time

* ADX operates in the **near-real time window**. Unlike at Datawarehouse updated hourly (or less), ADX provides latency of less than a minute using batch ingestion and a few seconds latency using [streaming ingestion](https://docs.microsoft.com/en-us/azure/data-explorer/ingest-data-streaming).
* In order to have this low latency of data “freshness”, ADX can **ingest data by itself**, without relying on external services (such as Azure Data Factory). For instance it can [ingest data from Event Hub directly](https://docs.microsoft.com/en-us/azure/data-explorer/data-connection-event-hub-python).
* Since it can ingest data by itself, it can also **transform the data as it is ingested** (cf [update policy](https://docs.microsoft.com/en-us/azure/kusto/management/update-policy)).

## Integration

ADX has an impressive gallery of integration for such a young service:

* Azure Data Factory
* Spark
* Jupyter Notebooks
* Azure Pipelines
* Event Hub
* Event Grid
* IoT Hub
* Kafka
* Logstash
* Power BI
* Excel
* Grafana
* Tableau
* ODBC connector
* Sisense

The list is growing and doesn’t contain only Azure technology. ADX can therefore easily be part of a bigger solution.

## What ADX isn’t optimal for / stretch scenarios

The public cloud brought a lot of fragmentation in the Data services. Although part of the reasons for that is the youth of the public cloud technologies, it is also due to inherent characteristics of big data analytics in the cloud:

*Since we do not own the hardware the workloads are running on, we do not have to get married with one technology and run everything on it to amortise the cost of said hardware / licence. We can use the best tool for the job.*

This is a balancing act as we need to take the skill set of people into account.

Most of the scenarios we are citing here can be done with ADX but it wouldn’t be the best platform to do so.

| Scenario | Why | Azure PaaS Alternatives |
| --- | --- | --- |
| [Data warehouse](https://en.wikipedia.org/wiki/Data_warehouse) | For starter, ADX is mostly an append-only store. It isn’t transactional, doesn’t have log journals, etc. . This is part of the reasons it is so fast, but also part of the reasons it is a poor fit for a Datawarehouse. Also, although it is very fast, pre-computed aggregations would be better for dashboards. For the sceptics, [the rumors of data warehousing’s dead have been greatly exaggerated](https://www.jamesserra.com/archive/2017/12/is-the-traditional-data-warehouse-dead/). | [Azure Synapse](https://docs.microsoft.com/en-us/azure/sql-data-warehouse/sql-data-warehouse-overview-what-is) & [Power BI Premium](https://docs.microsoft.com/en-us/power-bi/service-premium-what-is) |
| Application Back end | Similar to Data warehousing, ADX isn’t built as a transactional workload. | [Cosmos DB](https://docs.microsoft.com/en-us/azure/cosmos-db/introduction), [Azure SQL DB](https://docs.microsoft.com/en-us/azure/sql-database/sql-database-technical-overview), [Azure PostgreSQL](https://docs.microsoft.com/en-us/azure/postgresql/), [Azure MySQL](https://docs.microsoft.com/en-us/azure/mysql/overview), [Azure MariaDB](https://docs.microsoft.com/en-us/azure/mariadb/overview) |
| [Machine Learning](https://en.wikipedia.org/wiki/Machine_learning) (ML) Training | ADX supports some built-in [ML algorithms](https://docs.microsoft.com/en-us/azure/data-explorer/machine-learning-clustering) (mostly clustering algorithms and statistical tools at the time of this writing, i.e. February 2020), it isn’t an ML training platform. It is excellent for running prediction on a pre-training model though. | [Azure ML](https://docs.microsoft.com/en-us/azure/machine-learning/overview-what-is-azure-ml), Spark ([Azure Databricks](https://docs.microsoft.com/en-us/azure/databricks/getting-started/spark/machine-learning) or [Azure HD Insight](https://docs.microsoft.com/en-us/azure/hdinsight/hdinsight-overview)), [Azure Batch](https://docs.microsoft.com/en-us/azure/batch/batch-technical-overview) & [Data Science Virtual Machine](https://docs.microsoft.com/en-us/azure/machine-learning/data-science-virtual-machine/overview) (DSVM) |
| Sub-second streaming | ADX can go as low as seconds of latency in ingesting data and be able to do analytics (i.e. events are still indexed and can be queried). Most “near real time” scenarios fall comfortably within that window. But it isn’t a sub-second streaming platform (e.g. for low-latency-trading). | [Structured Streaming in Continuous Mode in Spark](https://databricks.com/blog/2018/03/20/low-latency-continuous-processing-mode-in-structured-streaming-in-apache-spark-2-3-0.html) ([Azure Databricks](https://docs.microsoft.com/en-us/azure/databricks/getting-started/spark/streaming) or [Azure HD Insight](https://docs.microsoft.com/en-us/azure/hdinsight/hdinsight-overview)), [Kafka Streams](https://kafka.apache.org/documentation/streams/) on [Azure HD Insight](https://docs.microsoft.com/en-us/azure/hdinsight/hdinsight-overview), [Flink](https://flink.apache.org/) on [Azure HD Insight](https://docs.microsoft.com/en-us/azure/hdinsight/hdinsight-overview) |

## Concrete scenarios

Here are some scenarios we’ve seen in different industries. This is by no mean an exhaustive list but the popular scenarios.

Quite a few customers are using ADX / Kusto to analyze unified logs, i.e. logs from on-premise systems and different clouds. This is typical log analysis, so it could be for security, reliability engineering, forecasting, etc. .

IoT telemetry analysis is quite popular. As customers capture telemetry, they want to mine that data.

We see different businesses using it to analyze transactions (sales) to understand customer behaviours, predict trends or spike and optimize go-to-market strategy. What if in days of deploying a new product we could figure out what customer segment is having traction and which ones are lagging?

In general, we see customers starting with historical analysis and then move to more and more real time analysis as the teams are getting more comfortable with the service.

## Summary

We hope we manage to give a good idea of what ADX can do.

It is also important to note that it is the data platform for other Azure Services:

* [Azure Monitor logs](https://docs.microsoft.com/en-us/azure/azure-monitor/overview)
* [Application Insights](https://docs.microsoft.com/en-us/azure/azure-monitor/app/app-insights-overview)
* [Azure Sentinel](https://docs.microsoft.com/en-us/azure/sentinel/overview)
* [Time Series Insights](https://docs.microsoft.com/en-us/azure/time-series-insights/time-series-insights-update-overview)

# **Azure HDinsight versus Azure Databricks**

Azure has multiple analytical tools nowadays. In this blog, I wanted to talk about Azure HDinsight and Azure Databricks and give a bit of background on them. One of the main questions is when would you choose one over the other.

## Azure HDInsight

First, let’s call it what it is: it’s Apache Hadoop running on Microsoft Azure. This means that we now have a cluster available in the cloud. Starting with some background on Hadoop:

**Hadoop:** An open-source framework for storing data and running apps on clusters. It offers massive storage for any data, lots of processing power. It can handle virtually “limitless” concurrent tasks. Hadoop has been declared open source and is now named Apache Hadoop.

In Azure, we can pick the following clusters that we may need in certain circumstances:

* **Hadoop:** Petabyte-scale processing.
* **Spark:** Fast data analytics and cluster computing using in-memory processing.
* **Kafka:** High throughput, low-latency, a real-time streaming platform using a publish-subscribe messaging system.
* **HBase:** Fast and Scalable NoSQL database.
* **Interactive Query:** Uses Hive (SQL on Hadoop) and LLAP (Low Latency Analytical Processing).
* **Storm:** Real-time streams of data through reliable processes.
* **ML Services:** A server for hosting and managing parallel distributed R processes.

We can only select one type of cluster during the configuration of the HDInsight. The HDinsight cluster cannot be turned off, so this can result in high costs during low use situations. For Active Directory integration with HDinsight, we need a few components to make it work. You will need the **Enterpise security package (ESP)**. For this, you will also need to deploy **Azure Active Directory Domain Services**. There is a high availability guarantee from Microsoft.

In short, Azure HDInsight provides the most popular open-source frameworks that are easily accessible from the portal. If you need a combination of multiple clusters for example: HDinsight Kafka for your streaming with Interactive Query, this would be a great choice.

## Azure Databricks

Azure Databricks is a newer service provided by Microsoft. Let’s start with some background information about Spark and Databricks:

**Spark:** General purpose distributed data processing engine. It can be used for a wide range of circumstances. It uses a lot of libraries that can be used. For example: SQL, machine learning, graph computing, and streaming processing. Spark does not provide storage, only a computation engine. Spark extends the Hadoop MapReduce framework to work in an optimized way.

**Databricks:** Databricks was founded by the creator of Spark. The team behind databricks keeps the Apache Spark engine optimized to run faster and faster. The databricks platform provides around five times more performance than an open-source Apache Spark. With Databricks, you have collaborative notebooks, integrated workflows, and enterprise security. This will be in a fully managed cloud platform.

Azure Databricks works on a premium Spark cluster. This one is faster than the open-source Spark. Azure Databricks is a PaaS solution. It doesn’t require a lot of admin work after the initial setup. It is providing security thanks to the **Azure Active Directory** integration without any need for custom configuration. It brings you all the pros that Databricks brings to you only then in Azure.

## Conclusion

The choice between Azure HDInsight and Azure Databricks depends on the use case that you want to solve. The biggest one is how are the data scientists going to work? Are they going to work without collaborating then it could be wiser to choose Azure HDInsight. Will, there be a lot of collaborating, then Azure Databricks can bring you the extra mile due to the shared notebooks and readily available workflows.

If you only need a spark cluster, then Azure Databricks will bring you that as it has better performance then an open-source Spark cluster.

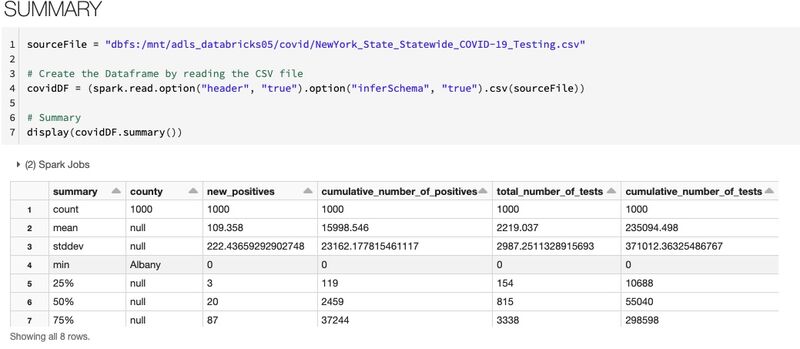
If you would like a Kafka based streaming service that is connected to a transformation tool, then the combination of HDinsight Kafka and Azure Databricks is the right solution.

If you have a lot of long running jobs that need high power then Azure HDInsight could be better then Azure Databricks.

[#Question369](https://www.linkedin.com/feed/hashtag/?keywords=question369&highlightedUpdateUrns=urn%3Ali%3Aactivity%3A6765557882684604416): **How can we DROP Nulls from a DataFrame in Pyspark ?**

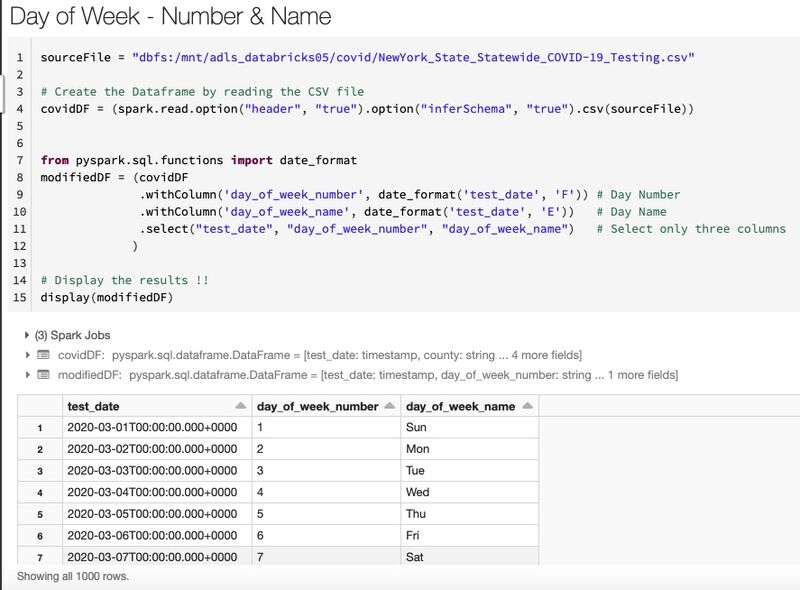
By using the na.drop() method  
Step1: Create the Dataframe with Nulls  
Step2: Apply the na.drop() method  
Step3: This method is having optional three parameters  
drop(how='any', thresh=None, subset=None)  
[#Parameters](https://www.linkedin.com/feed/hashtag/?keywords=parameters&highlightedUpdateUrns=urn%3Ali%3Aactivity%3A6765557882684604416):  
[#how](https://www.linkedin.com/feed/hashtag/?keywords=how&highlightedUpdateUrns=urn%3Ali%3Aactivity%3A6765557882684604416) – ‘any’ or ‘all’. If ‘any’, drop a row if it contains any nulls. If ‘all’, drop a row only if all its values are null.  
[#thresh](https://www.linkedin.com/feed/hashtag/?keywords=thresh&highlightedUpdateUrns=urn%3Ali%3Aactivity%3A6765557882684604416) – int, default None. If specified, drop rows that have less than thresh non-null values. This overwrites the how parameter.  
[#subset](https://www.linkedin.com/feed/hashtag/?keywords=subset&highlightedUpdateUrns=urn%3Ali%3Aactivity%3A6765557882684604416) – optional list of column names to consider.  
Step4: apply na.drop() method with different options based on your use case the way we see in the below code snippet.  
Step5: Display the results ( This time I am using show() method )  
  
[#apachespark](https://www.linkedin.com/feed/hashtag/?keywords=apachespark&highlightedUpdateUrns=urn%3Ali%3Aactivity%3A6765557882684604416) [#pyspark](https://www.linkedin.com/feed/hashtag/?keywords=pyspark&highlightedUpdateUrns=urn%3Ali%3Aactivity%3A6765557882684604416) [#spark](https://www.linkedin.com/feed/hashtag/?keywords=spark&highlightedUpdateUrns=urn%3Ali%3Aactivity%3A6765557882684604416) [#dataengineering](https://www.linkedin.com/feed/hashtag/?keywords=dataengineering&highlightedUpdateUrns=urn%3Ali%3Aactivity%3A6765557882684604416) [#justenoughspark](https://www.linkedin.com/feed/hashtag/?keywords=justenoughspark&highlightedUpdateUrns=urn%3Ali%3Aactivity%3A6765557882684604416) [#bytesizelearning](https://www.linkedin.com/feed/hashtag/?keywords=bytesizelearning&highlightedUpdateUrns=urn%3Ali%3Aactivity%3A6765557882684604416)



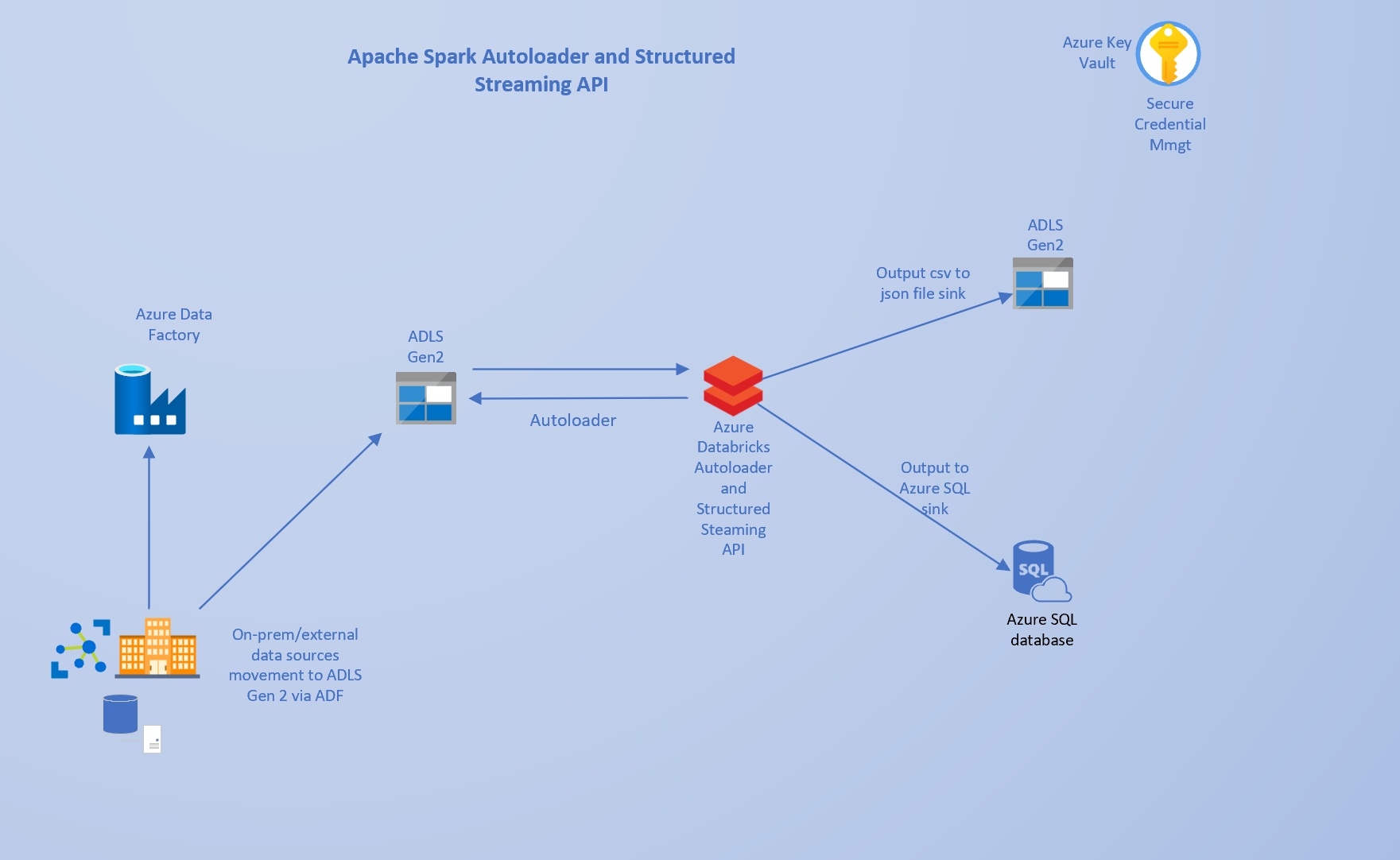
[#Question368](https://www.linkedin.com/feed/hashtag/?keywords=question368&highlightedUpdateUrns=urn%3Ali%3Aactivity%3A6765276942607482880): What is the SUMMARY() method in Pyspark ?  
  
it computes specified statistics for numeric and string columns.  
Available statistics are: - count - mean - stddev - min - max - arbitrary approximate percentiles specified as a percentage (eg, 75%)  
If no statistics are given, this function computes count, mean, stddev, min, approximate quartiles (percentiles at 25%, 50%, and 75%), and max.  
Step1: Create the Dataframe  
Step2: Apply the summary method  
Step3: We can limit the statistics by passing them as argument i.e. df.summary("count", "min", "25%", "75%", "max")  
Step4: Display the results  
  


.

**Question: How to get the Day Number of a week & the Day Name of a Week in Pyspark ?**  
  
By using date\_format  
Step1: Create the Dataframe which is having the date / timestamp  
Step2: use date\_format function to get the Day number & Day name  
Step3: use .withColumn method to create 2 additional columns.  
Step4: for Day number provide the option - "F & for Day name provide the option - "E"  
Step5: Display the results



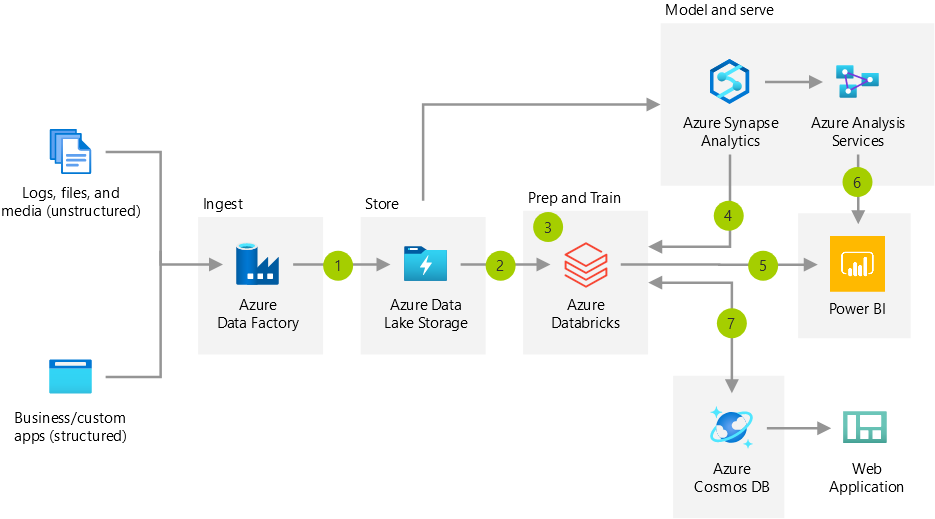
Architrecture 1:



# Advanced Analytics Architecture

Transform your data into actionable insights using the best-in-class machine learning tools. This architecture allows you to combine any data at any scale, and to build and deploy custom machine learning models at scale.

## Architecture

 Download an [*SVG*](https://docs.microsoft.com/en-us/azure/architecture/solution-ideas/media/advanced-analytics-on-big-data.svg) of this architecture.

## Data Flow

1. Bring together all your structured, unstructured and semi-structured data (logs, files, and media) using Azure Data Factory to Azure Data Lake Storage.
2. Use Azure Databricks to clean and transform the structureless datasets and combine them with structured data from operational databases or data warehouses.
3. Use scalable machine learning/deep learning techniques, to derive deeper insights from this data using Python, R or Scala, with inbuilt notebook experiences in Azure Databricks.
4. Leverage native connectors between Azure Databricks and Azure Synapse Analytics to access and move data at scale.
5. Power users take advantage of the inbuilt capabilities of Azure Databricks to perform root cause determination and raw data analysis.
6. Query and report on data in [Power BI](https://docs.microsoft.com/en-us/azure/analysis-services/analysis-services-connect-pbi).
7. Take the insights from Azure Databricks to Cosmos DB to make them accessible through web and mobile apps.

## Components

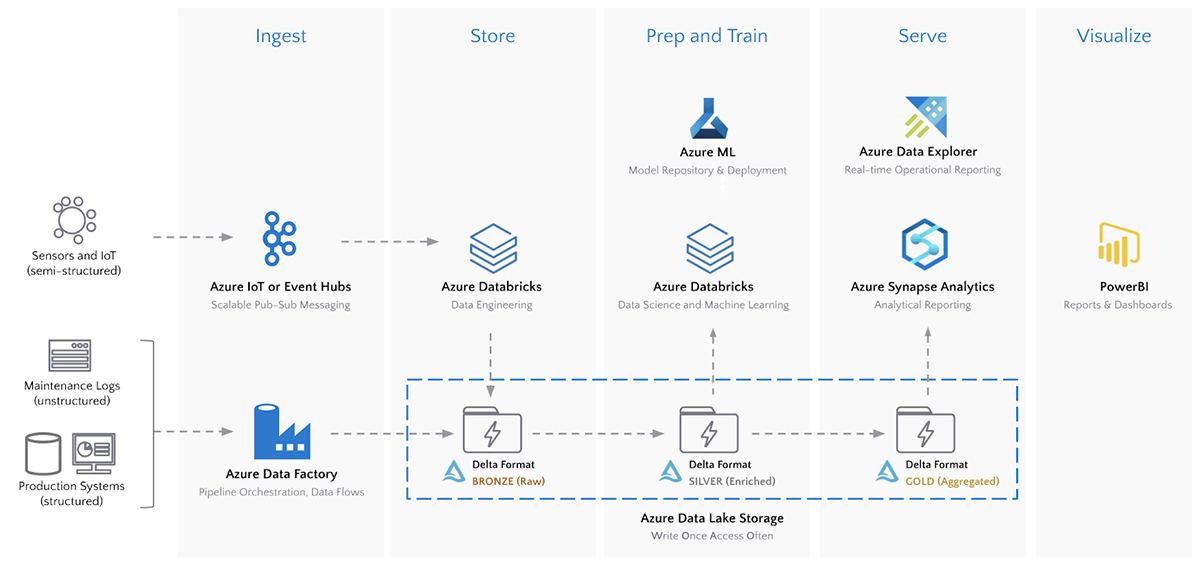
* [Azure Synapse Analytics](https://azure.microsoft.com/services/synapse-analytics) is the fast, flexible and trusted cloud data warehouse that lets you scale, compute and store elastically and independently, with a massively parallel processing architecture.
* Azure [Data Factory](https://azure.microsoft.com/services/data-factory) is a hybrid data integration service that allows you to create, schedule and orchestrate your ETL/ELT workflows.
* [Azure Blob storage](https://azure.microsoft.com/services/storage/blobs) is a Massively scalable object storage for any type of unstructured data-images, videos, audio, documents, and more-easily and cost-effectively.
* [Azure Databricks](https://azure.microsoft.com/services/databricks) is a fast, easy, and collaborative Apache Spark-based analytics platform.
* [Azure Cosmos DB](https://azure.microsoft.com/services/cosmos-db) is a globally distributed, multi-model database service. Learn how to replicate your data across any number of Azure regions and scale your throughput independent from your storage.
* [Azure Analysis Services](https://azure.microsoft.com/services/analysis-services) is an enterprise grade analytics as a service that lets you govern, deploy, test, and deliver your BI solution with confidence.
* [Power BI](https://powerbi.microsoft.com/) is a suite of business analytics tools that deliver insights throughout your organization. Connect to hundreds of data sources, simplify data prep, and drive ad hoc analysis. Produce beautiful reports, then publish them for your organization to consume on the web and across mobile devices.

Architectire 2:

The Architecture – Ingest, Store, Prep, Train, Serve, Visualize

Part 1:

The architecture below illustrates a modern, best-of-breed platform used by many organizations that leverages all that Azure has to offer for IIoT analytics.

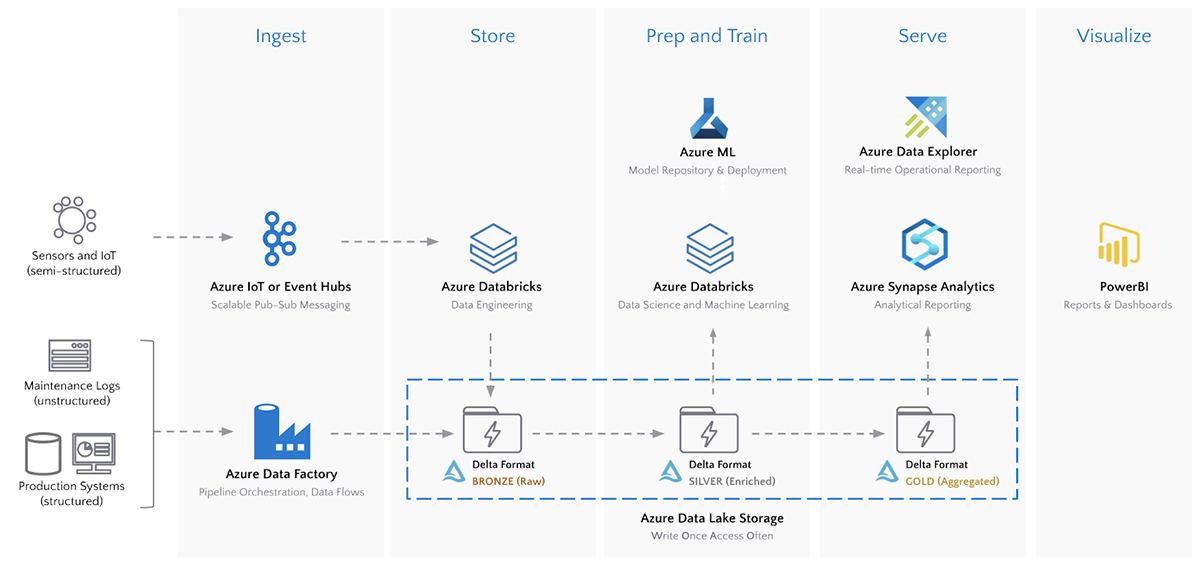
[](https://databricks.com/wp-content/uploads/2020/07/blog-iot-part-1-3.png)

* A key component of this architecture is the Azure Data Lake Store (ADLS), which enables the write-once, access-often analytics pattern in Azure. However, Data Lakes alone do not solve the real-world challenges that come with time-series streaming data. The Delta storage format provides a layer of resiliency and performance on all data sources stored in ADLS. Specifically for time-series data, Delta provides the following advantages over other storage formats on ADLS:

|  |  |  |
| --- | --- | --- |
| **Required Capability** | **Other formats on ADLS Gen 2** | **Delta Format on ADLS Gen 2** |
| Unified batch & streaming | Data Lakes are often used in conjunction with a streaming store like CosmosDB, resulting in a complex architecture | ACID-compliant transactions enable data engineers to perform streaming ingest and historically batch loads into the same locations on ADLS |
| Schema enforcement and evolution | Data Lakes do not enforce schema, requiring all data to be pushed into a relational database for reliability | Schema is enforced by default. As new IoT devices are added to the data stream, schemas can be evolved safely so downstream applications don’t fail |
| Efficient Upserts | Data Lakes do not support in-line updates and merges, requiring deletion and insertions of entire partitions to perform updates | MERGE commands are effective for situations handling delayed IoT readings, modified dimension tables used for real-time enrichment, or if data needs to be reprocessed. |
| File Compaction | Streaming time-series data into Data Lakes generates hundreds or even thousands of tiny files. | Auto-compaction in Delta optimizes the file sizes to increase throughput and parallelism. |
| Multi-dimensional clustering | Data Lakes provide push-down filtering on partitions only | ZORDERing time-series on fields like timestamp or sensor ID allows Databricks to filter and join on those columns up to 100x faster than simple partitioning techniques. |

* Summary
* In this post we reviewed a number of different challenges facing traditional IIoT systems.  We walked through the use case and the goals for modern IIoT analytics, shared a repeatable architecture that organizations are already deploying at scale and explored the benefits of Delta format for each of the required capabilities.

In [**part 1**](https://databricks.com/blog/2020/08/03/modern-industrial-iot-analytics-on-azure-part-1.html)  of the series on Modern Industrial Internet of Things (IoT) Analytics on Azure,  we walked through the big data use case and the goals for modern IIoT analytics, shared a real-world repeatable architecture in use by organizations to deploy IIoT at scale and explored the benefits of Delta format for each of the data lake capabilities required for modern IIoT analytics.

**[](https://databricks.com/wp-content/uploads/2020/08/blog-iiot-part-2-1.png)**

## The Deployment

We use Azure’s Raspberry PI IoT Simulator to simulate real-time machine-to-machine sensor readings and send them to Azure IoT Hub.

### Data Ingest: Azure IoT Hub to Data Lake

Our deployment has sensor readings for weather (wind speed & direction, temperature, humidity) and wind turbine telematics (angle and RPM) sent to an IoT cloud computing hub. Azure Databricks can natively stream data from IoT Hubs directly into a Delta table on ADLS and display the input vs. processing rates of the data.

# Read directly from IoT Hubs using the EventHubs library for Azure Databricks

iot\_stream = (

spark.readStream.format("eventhubs") # Read from IoT Hubs directly

.options(\*\*ehConf) # Use the Event-Hub-enabled connect string

.load() # Load the data

.withColumn('reading', F.from\_json(F.col('body').cast('string'), schema)) # Extract the payload from the messages

.select('reading.\*', F.to\_date('reading.timestamp').alias('date')) # Create a "date" field for partitioning

)

# Split our IoT Hubs stream into separate streams and write them both into their own Delta locations

write\_turbine\_to\_delta = (

iot\_stream.filter('temperature is null') # Filter out turbine telemetry from other streams

.select('date','timestamp','deviceId','rpm','angle') # Extract the fields of interest

.writeStream.format('delta') # Write our stream to the Delta format

.partitionBy('date') # Partition our data by Date for performance

.option("checkpointLocation", ROOT\_PATH + "/bronze/cp/turbine") # Checkpoint so we can restart streams gracefully

.start(ROOT\_PATH + "/bronze/data/turbine\_raw") # Stream the data into an ADLS Path

)

Delta allows our IoT data to be queried within seconds of it being captured in IoT Hub.

%sql

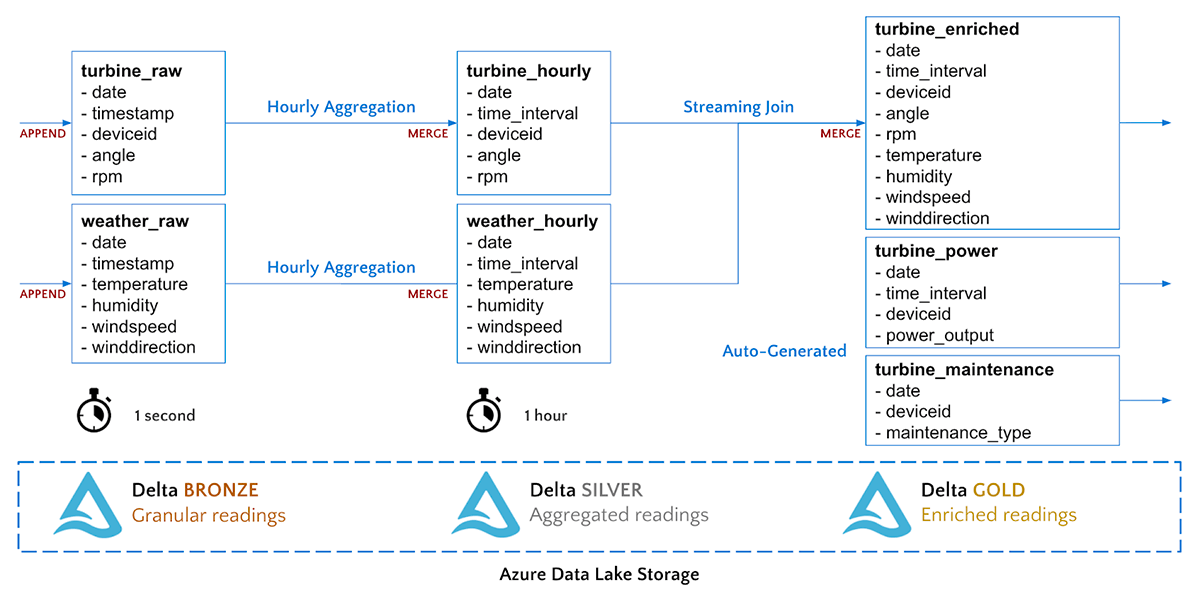
-- We can query the data directly from storage immediately as it streams into Delta

SELECT \* FROM delta.`/tmp/iiot/bronze/data/turbine\_raw` WHERE deviceid = 'WindTurbine-1'

We can now build a downstream pipeline that enriches and aggregates our IIoT applications data for data analytics.

### Data Storage and Processing: Azure Databricks and Delta Lake

Delta supports a multi-hop pipeline approach to data engineering, where data quality and aggregation increases as it streams through the pipeline. Our time-series data will flow through the following Bronze, Silver and Gold data levels.

**[](https://databricks.com/wp-content/uploads/2020/08/blog-iiot-part-2-3.png)**

Our pipeline from Bronze to Silver will simply aggregate our turbine sensor data to 1 hour intervals. We will perform a streaming MERGE command to upsert the aggregated records into our Silver Delta tables.

# Create functions to merge turbine and weather data into their target Delta tables

def merge\_records(incremental, target\_path):

incremental.createOrReplaceTempView("incremental")

# MERGE consists of a target table, a source table (incremental),

# a join key to identify matches (deviceid, time\_interval), and operations to perform

# (UPDATE, INSERT, DELETE) when a match occurs or not

incremental.\_jdf.sparkSession().sql(f"""

MERGE INTO turbine\_hourly t

USING incremental i

ON i.date=t.date AND i.deviceId = t.deviceid AND i.time\_interval = t.time\_interval

WHEN MATCHED THEN UPDATE SET \*

WHEN NOT MATCHED THEN INSERT \*

""")

# Perform the streaming merge into our data stream

turbine\_stream = (

spark.readStream.format('delta').table('turbine\_raw') # Read data as a stream from our source Delta table

.groupBy('deviceId','date',F.window('timestamp','1 hour')) # Aggregate readings to hourly intervals

.agg({"rpm":"avg","angle":"avg"})

.writeStream

.foreachBatch(merge\_records) # Pass each micro-batch to a function

.outputMode("update") # Merge works with update mod

.start()

)

Our pipeline from Silver to Gold will join the two streams together into a single table for hourly weather and turbine measurements.

# Read streams from Delta Silver tables

turbine\_hourly = spark.readStream.format('delta').option("ignoreChanges", True).table("turbine\_hourly")

weather\_hourly = spark.readStream.format('delta').option("ignoreChanges", True).table("weather\_hourly")

# Perform a streaming join to enrich the data

turbine\_enriched = turbine\_hourly.join(weather\_hourly, ['date','time\_interval'])

# Perform a streaming merge into our Gold data stream

merge\_gold\_stream = (

turbine\_enriched.writeStream

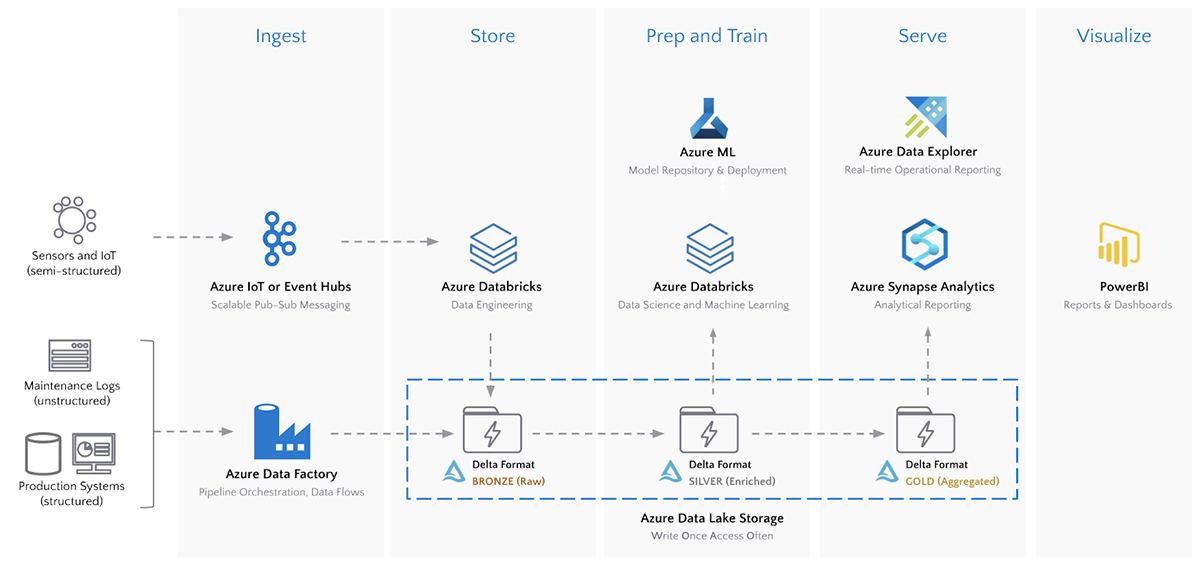
.foreachBatch(merge\_records)

.start()

)

We can query our Gold Delta table immediately.

In [**part 1**](https://databricks.com/blog/2020/08/03/modern-industrial-iot-analytics-on-azure-part-1.html)  of the series on Modern Industrial Internet of Things (IoT) Analytics on Azure,  we walked through the big data use case and the goals for modern IIoT analytics, shared a real-world repeatable architecture in use by organizations to deploy IIoT at scale and explored the benefits of Delta format for each of the data lake capabilities required for modern IIoT analytics.

**[](https://databricks.com/wp-content/uploads/2020/08/blog-iiot-part-2-1.png)**

## The Deployment

We use Azure’s Raspberry PI IoT Simulator to simulate real-time machine-to-machine sensor readings and send them to Azure IoT Hub.

### Data Ingest: Azure IoT Hub to Data Lake

Our deployment has sensor readings for weather (wind speed & direction, temperature, humidity) and wind turbine telematics (angle and RPM) sent to an IoT cloud computing hub. Azure Databricks can natively stream data from IoT Hubs directly into a Delta table on ADLS and display the input vs. processing rates of the data.

# Read directly from IoT Hubs using the EventHubs library for Azure Databricks

iot\_stream = (

spark.readStream.format("eventhubs") # Read from IoT Hubs directly

.options(\*\*ehConf) # Use the Event-Hub-enabled connect string

.load() # Load the data

.withColumn('reading', F.from\_json(F.col('body').cast('string'), schema)) # Extract the payload from the messages

.select('reading.\*', F.to\_date('reading.timestamp').alias('date')) # Create a "date" field for partitioning

)

# Split our IoT Hubs stream into separate streams and write them both into their own Delta locations

write\_turbine\_to\_delta = (

iot\_stream.filter('temperature is null') # Filter out turbine telemetry from other streams

.select('date','timestamp','deviceId','rpm','angle') # Extract the fields of interest

.writeStream.format('delta') # Write our stream to the Delta format

.partitionBy('date') # Partition our data by Date for performance

.option("checkpointLocation", ROOT\_PATH + "/bronze/cp/turbine") # Checkpoint so we can restart streams gracefully

.start(ROOT\_PATH + "/bronze/data/turbine\_raw") # Stream the data into an ADLS Path

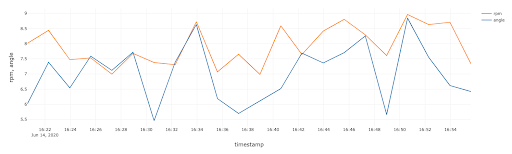
)

Delta allows our IoT data to be queried within seconds of it being captured in IoT Hub.

%sql

-- We can query the data directly from storage immediately as it streams into Delta

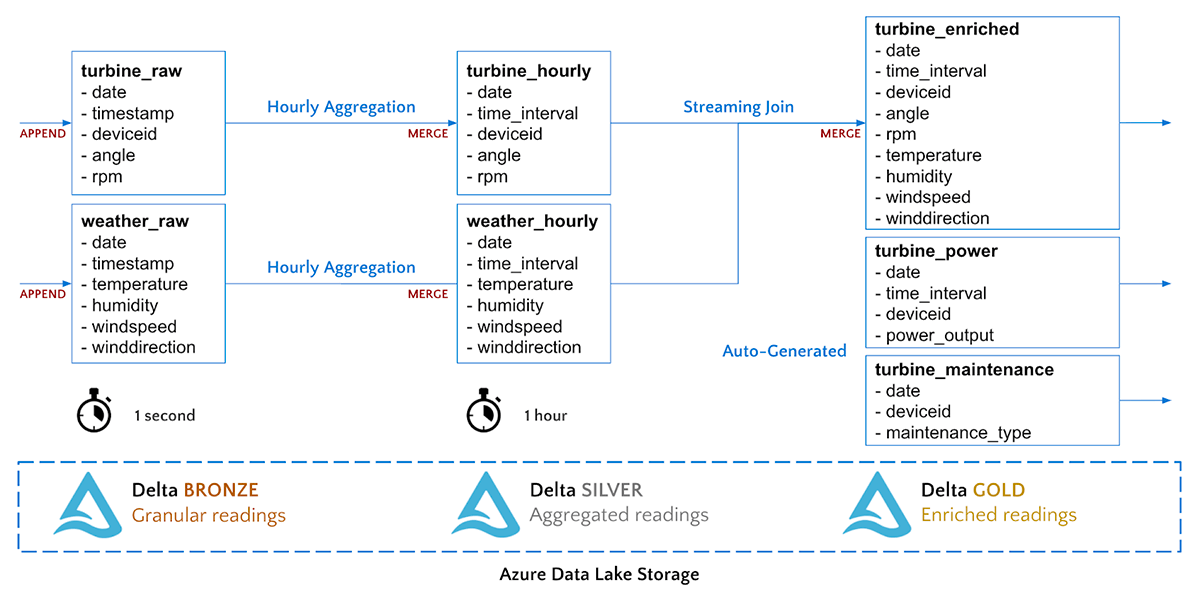
SELECT \* FROM delta.`/tmp/iiot/bronze/data/turbine\_raw` WHERE deviceid = 'WindTurbine-1'

**[](https://databricks.com/wp-content/uploads/2020/08/blog-iiot-part-2-2.png)**

We can now build a downstream pipeline that enriches and aggregates our IIoT applications data for data analytics.

### Data Storage and Processing: Azure Databricks and Delta Lake

Delta supports a multi-hop pipeline approach to data engineering, where data quality and aggregation increases as it streams through the pipeline. Our time-series data will flow through the following Bronze, Silver and Gold data levels.

**[](https://databricks.com/wp-content/uploads/2020/08/blog-iiot-part-2-3.png)**

Our pipeline from Bronze to Silver will simply aggregate our turbine sensor data to 1 hour intervals. We will perform a streaming MERGE command to upsert the aggregated records into our Silver Delta tables.

# Create functions to merge turbine and weather data into their target Delta tables

def merge\_records(incremental, target\_path):

incremental.createOrReplaceTempView("incremental")

# MERGE consists of a target table, a source table (incremental),

# a join key to identify matches (deviceid, time\_interval), and operations to perform

# (UPDATE, INSERT, DELETE) when a match occurs or not

incremental.\_jdf.sparkSession().sql(f"""

MERGE INTO turbine\_hourly t

USING incremental i

ON i.date=t.date AND i.deviceId = t.deviceid AND i.time\_interval = t.time\_interval

WHEN MATCHED THEN UPDATE SET \*

WHEN NOT MATCHED THEN INSERT \*

""")

# Perform the streaming merge into our data stream

turbine\_stream = (

spark.readStream.format('delta').table('turbine\_raw') # Read data as a stream from our source Delta table

.groupBy('deviceId','date',F.window('timestamp','1 hour')) # Aggregate readings to hourly intervals

.agg({"rpm":"avg","angle":"avg"})

.writeStream

.foreachBatch(merge\_records) # Pass each micro-batch to a function

.outputMode("update") # Merge works with update mod

.start()

)

Our pipeline from Silver to Gold will join the two streams together into a single table for hourly weather and turbine measurements.

# Read streams from Delta Silver tables

turbine\_hourly = spark.readStream.format('delta').option("ignoreChanges", True).table("turbine\_hourly")

weather\_hourly = spark.readStream.format('delta').option("ignoreChanges", True).table("weather\_hourly")

# Perform a streaming join to enrich the data

turbine\_enriched = turbine\_hourly.join(weather\_hourly, ['date','time\_interval'])

# Perform a streaming merge into our Gold data stream

merge\_gold\_stream = (

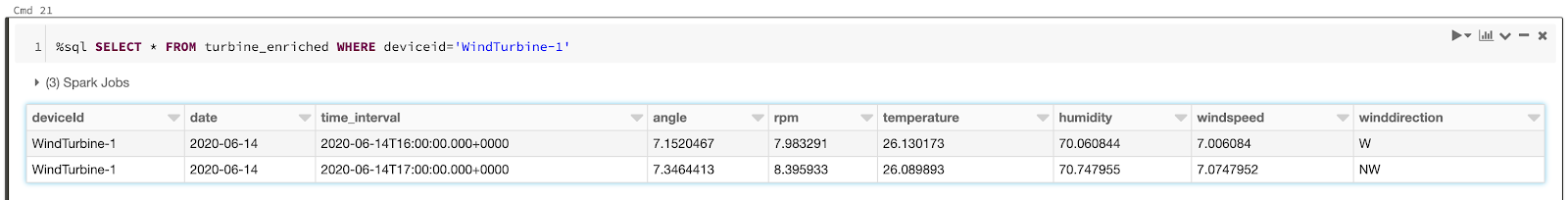
turbine\_enriched.writeStream

.foreachBatch(merge\_records)

.start()

)

We can query our Gold Delta table immediately.

**[](https://databricks.com/wp-content/uploads/2020/08/blog-iiot-part-2-4.png)**

The notebook also contains a cell that will generate historical hourly power readings and daily maintenance logs that will be used for model training. Running that cell will:

1. Backfill historical readings for 1 year in the turbine\_enriched table
2. Generate historical power readings for each turbine in the power\_output table
3. Generate historical maintenance logs for each turbine in the turbine\_maintenance table

We now have enriched, artificial intelligence (AI)-ready data in a performant, reliable format on Azure Data Lake that can be fed into our data science modeling to optimize asset utilization.

%sql

-- Query all 3 tables together

CREATE OR REPLACE VIEW gold\_readings AS

SELECT r.\*,

p.power,

m.maintenance as maintenance

FROM turbine\_enriched r

JOIN turbine\_power p ON (r.date=p.date AND r.time\_interval=p.time\_interval AND r.deviceid=p.deviceid)

LEFT JOIN turbine\_maintenance m ON (r.date=m.date AND r.deviceid=m.deviceid);

SELECT \* FROM gold\_readings

Our data engineering pipeline is complete! Data is now flowing from IoT Hubs to Bronze (raw) to Silver (aggregated) to Gold (enriched). It is time to perform some analytics on our data.

## Summary

To summarize, we have successfully:

* Ingested real-time IIoT data from field devices into Azure
* Performed complex time-series processing on Data Lake directly

They key technology that ties everything together is Delta Lake. Delta on ADLS provides reliable streaming data pipelines and highly performant data science and analytics queries on massive volumes of time-series data. Lastly, it enables organizations to truly adopt a Lakehouse pattern by bringing best of breed Azure tools to a write-once, access-often data store.

The notebook also contains a cell that will generate historical hourly power readings and daily maintenance logs that will be used for model training. Running that cell will:

1. Backfill historical readings for 1 year in the turbine\_enriched table
2. Generate historical power readings for each turbine in the power\_output table
3. Generate historical maintenance logs for each turbine in the turbine\_maintenance table

We now have enriched, artificial intelligence (AI)-ready data in a performant, reliable format on Azure Data Lake that can be fed into our data science modeling to optimize asset utilization.

%sql

-- Query all 3 tables together

CREATE OR REPLACE VIEW gold\_readings AS

SELECT r.\*,

p.power,

m.maintenance as maintenance

FROM turbine\_enriched r

JOIN turbine\_power p ON (r.date=p.date AND r.time\_interval=p.time\_interval AND r.deviceid=p.deviceid)

LEFT JOIN turbine\_maintenance m ON (r.date=m.date AND r.deviceid=m.deviceid);

SELECT \* FROM gold\_readings

Our data engineering pipeline is complete! Data is now flowing from IoT Hubs to Bronze (raw) to Silver (aggregated) to Gold (enriched). It is time to perform some analytics on our data.

## Summary

To summarize, we have successfully:

* Ingested real-time IIoT data from field devices into Azure
* Performed complex time-series processing on Data Lake directly

They key technology that ties everything together is Delta Lake. Delta on ADLS provides reliable streaming data pipelines and highly performant data science and analytics queries on massive volumes of time-series data. Lastly, it enables organizations to truly adopt a Lakehouse pattern by bringing best of breed Azure tools to a write-once, access-often data store.

Architecture 3:

# Azure data platform end-to-end

# Azure data platform

This example scenario demonstrates how to use the extensive family of Azure Data Services to build a modern data platform capable of handling the most common data challenges in an organization.

The solution described in this article combines a range of Azure services that will ingest, process, store, serve, and visualize data from different sources, both structured and unstructured.

This solution architecture demonstrates how a single, unified data platform can be used to meet the most common requirements for:

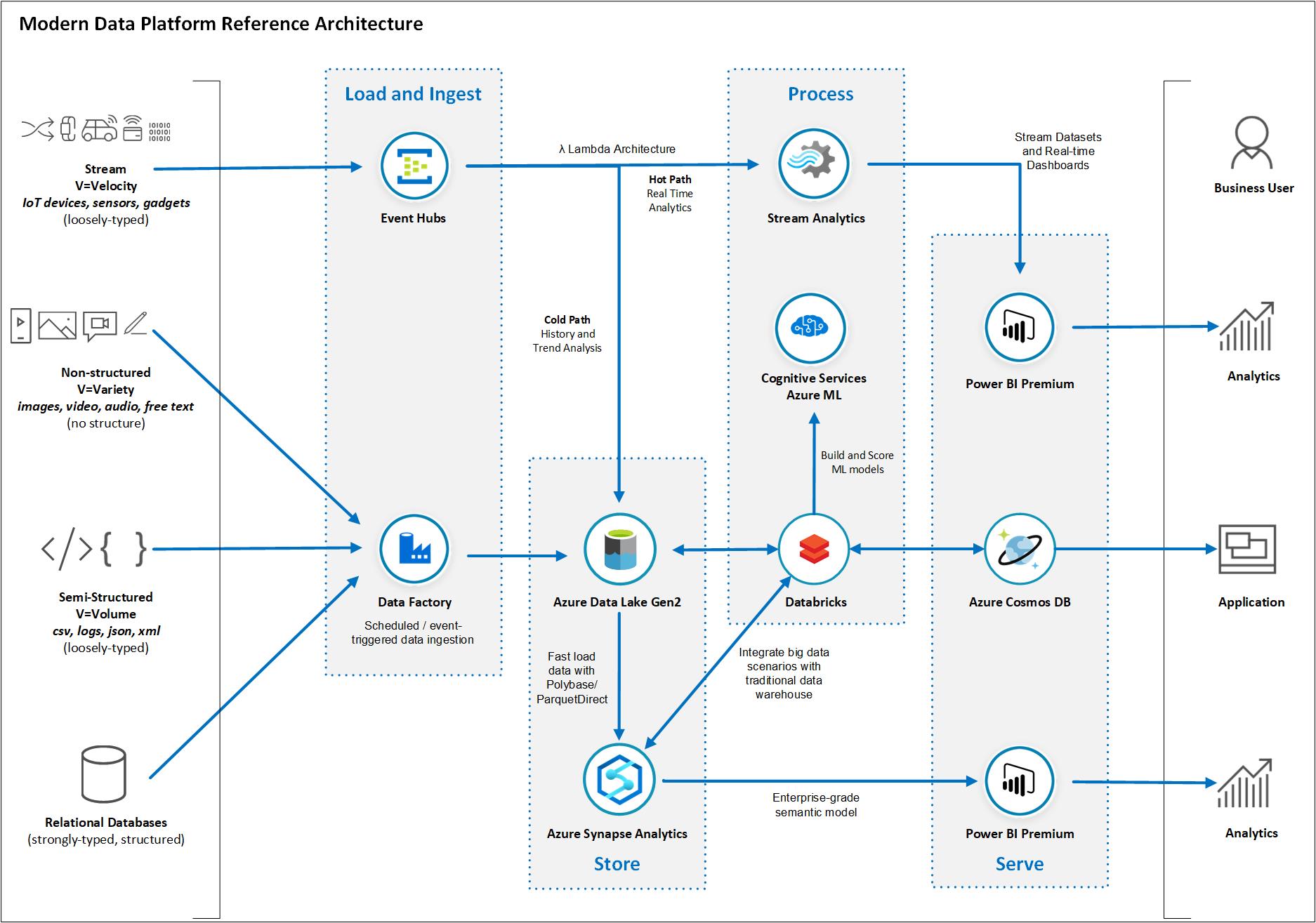
* Traditional relational data pipelines
* Big data transformations
* Unstructured data ingestion and enrichment with AI-based functions
* Stream ingestion and processing following the Lambda architecture
* Serving insights for data-driven applications and rich data visualization

## Relevant use cases

This approach can also be used to:

* Establish an enterprise-wide data hub consisting of a data warehouse for structured data and a data lake for semi-structured and unstructured data. This data hub becomes the single source of truth for your data.
* Integrate relational data sources with other unstructured datasets with the use of big data processing technologies;
* Use semantic modeling and powerful visualization tools for simpler data analysis.

## Architecture

[](https://docs.microsoft.com/en-us/azure/architecture/example-scenario/dataplate2e/media/azure-data-platform-end-to-end.jpg#lightbox)

Note

* The services covered by this architecture are only a subset of a much larger family of Azure services. Similar outcomes can be achieved by using other services or features not covered by this design.
* Specific business requirements for your analytics use case may also ask for the use of different services or features not considered in this design.

The data flows through the solution as follows (from bottom-up):

### Relational databases

1. Use Azure Data Factory pipelines to pull data from a wide variety of databases, both on-premises and in the cloud. Pipelines can be triggered based on a pre-defined schedule, in response to an event or be explicitly called via REST APIs.
2. Still part of the Azure Data Factory pipeline, use Azure Data Lake Store Gen 2 to stage the data copied from the relational databases. You can save the data in delimited text format or compressed as Parquet files.
3. Use Azure Synapse PolyBase capabilities for fast ingestion into your data warehouse tables.
4. Load relevant data from the Azure Synapse data warehouse into Power BI datasets for data visualization. Power BI models implement a semantic model to simplify the analysis of business data and relationships.
5. Business analysts use Power BI reports and dashboards to analyze data and derive business insights.

### Semi-structured data sources

1. Use Azure Data Factory pipelines to pull data from a wide variety of semi-structured data sources, both on-premises and in the cloud. For example, you can ingest data from file-based locations containing CSV or JSON files. You can connect to No-SQL databases such as Cosmos DB or Mongo DB. Or you call REST APIs provided by SaaS applications that will function as your data source for the pipeline.
2. Still part of the Azure Data Factory pipeline, use Azure Data Lake Store Gen 2 to save the original data copied from the semi-structured data source.
3. Azure Data Factory Mapping Data Flows or Azure Databricks notebooks can now be used to process the semi-structured data and apply the necessary transformations before data can be used for reporting. You can save the resulting dataset as Parquet files in the data lake.
4. Use Azure Synapse PolyBase capabilities for fast ingestion into your data warehouse tables.
5. Load relevant data from the Azure Synapse data warehouse into Power BI datasets for data visualization. Power BI models implement a semantic model to simplify the analysis of business data and relationships.
6. Business analysts use Power BI reports and dashboards to analyze data and derive business insights.

### Non-structured data sources

1. Use Azure Data Factory pipelines to pull data from a wide variety of unstructured data sources, both on-premises and in the cloud. For example, you can ingest video, image or free text log data from file-based locations. You can also call REST APIs provided by SaaS applications that will function as your data source for the pipeline.
2. Still part of the Azure Data Factory pipeline, use Azure Data Lake Store Gen 2 to save the original data copied from the unstructured data source.
3. You can invoke Azure Databricks notebooks from your pipeline to process the unstructured data. The notebook can make use of Cognitive Services APIs or invoke custom Azure Machine Learning Service models to generate insights from the unstructured data. You can save the resulting dataset as Parquet files in the data lake.
4. Use Azure Synapse PolyBase capabilities for fast ingestion into your data warehouse tables.
5. Load relevant data from the Azure Synapse data warehouse into Power BI datasets for data visualization. Power BI models implement a semantic model to simplify the analysis of business data and relationships.
6. Business analysts use Power BI reports and dashboards to analyze data and derive business insights.

### Streaming

1. Use Azure Event Hubs to ingest data streams generated by a client application. The Event Hub will then ingest and store streaming data preserving the sequence of events received. Consumers can then connect to Event Hub and retrieve the messages for processing.
2. Configure the Event Hub Capture to save a copy of the events in your data lake. This feature implements the “Cold Path” of the Lambda architecture pattern and allows you to perform historical and trend analysis on the stream data saved in your data lake using tools such as Azure Databricks notebooks.
3. Use a Stream Analytics job to implement the “Hot Path” of the Lambda architecture pattern and derive insights from the stream data in transit. Define at least one input for the data stream coming from your Event Hub, one query to process the input data stream and one Power BI output to where the query results will be sent to.
4. Business analysts then use Power BI real-time datasets and dashboard capabilities for to visualize the fast changing insights generated by your Stream Analytics query.

## Architecture components

The following Azure services have been used in the architecture:

* Azure Data Factory
* Azure Data Lake Gen2
* Azure Synapse Analytics
* Azure Databricks
* Azure Cosmos DB
* Azure Cognitive Services
* Azure Event Hubs
* Azure Stream Analytics
* Microsoft Power BI

If you need further training resources or access to technical documentation, the table below links to Microsoft Learn and to each service’s Technical Documentation.

| **Azure Service** | **Microsoft Learn** | **Technical Documentation** |
| --- | --- | --- |
| Azure Data Factory | [Data ingestion with Azure Data Factory](https://docs.microsoft.com/en-us/learn/modules/data-ingestion-with-azure-data-factory) | [Azure Data Factory Technical Documentation](https://docs.microsoft.com/en-us/azure/data-factory) |
| Azure Synapse Analytics | [Implement a Data Warehouse with Azure Synapse Analytics](https://docs.microsoft.com/en-us/learn/paths/implement-sql-data-warehouse) | [Azure Synapse Analytics Technical Documentation](https://docs.microsoft.com/en-us/azure/sql-data-warehouse) |
| Azure Data Lake Storage Gen2 | [Large Scale Data Processing with Azure Data Lake Storage Gen2](https://docs.microsoft.com/en-us/learn/paths/data-processing-with-azure-adls) | [Azure Data Lake Storage Gen2 Technical Documentation](https://docs.microsoft.com/en-us/azure/storage/blobs/data-lake-storage-introduction) |
| Azure Cognitive Services | [Cognitive Services Learning Paths and Modules](https://docs.microsoft.com/en-us/learn/browse/?term=cognitive) | [Azure Cognitive Services Technical Documentation](https://docs.microsoft.com/en-us/azure/cognitive-services) |
| Azure Cosmos DB | [Work with NoSQL data in Azure Cosmos DB](https://docs.microsoft.com/en-us/learn/paths/work-with-nosql-data-in-azure-cosmos-db) | [Azure Cosmos DB Technical Documentation](https://docs.microsoft.com/en-us/azure/cosmos-db) |
| Azure Databricks | [Perform data engineering with Azure Databricks](https://docs.microsoft.com/en-us/learn/paths/data-engineering-with-databricks) | [Azure Databricks Technical Documentation](https://docs.microsoft.com/en-us/azure/azure-databricks) |
| Azure Event Hubs | [Enable reliable messaging for Big Data applications using Azure Event Hubs](https://docs.microsoft.com/en-us/learn/modules/enable-reliable-messaging-for-big-data-apps-using-event-hubs) | [Azure Event Hubs Technical Documentation](https://docs.microsoft.com/en-us/azure/event-hubs) |
| Azure Stream Analytics | [Implement a Data Streaming Solution with Azure Streaming Analytics](https://docs.microsoft.com/en-us/learn/paths/implement-data-streaming-with-asa) | [Azure Stream Analytics Technical Documentation](https://docs.microsoft.com/en-us/azure/stream-analytics) |
| Power BI | [Create and use analytics reports with Power BI](https://docs.microsoft.com/en-us/learn/paths/create-use-analytics-reports-power-bi) | [Power BI Technical Documentation](https://docs.microsoft.com/en-us/power-bi) |

### Alternatives

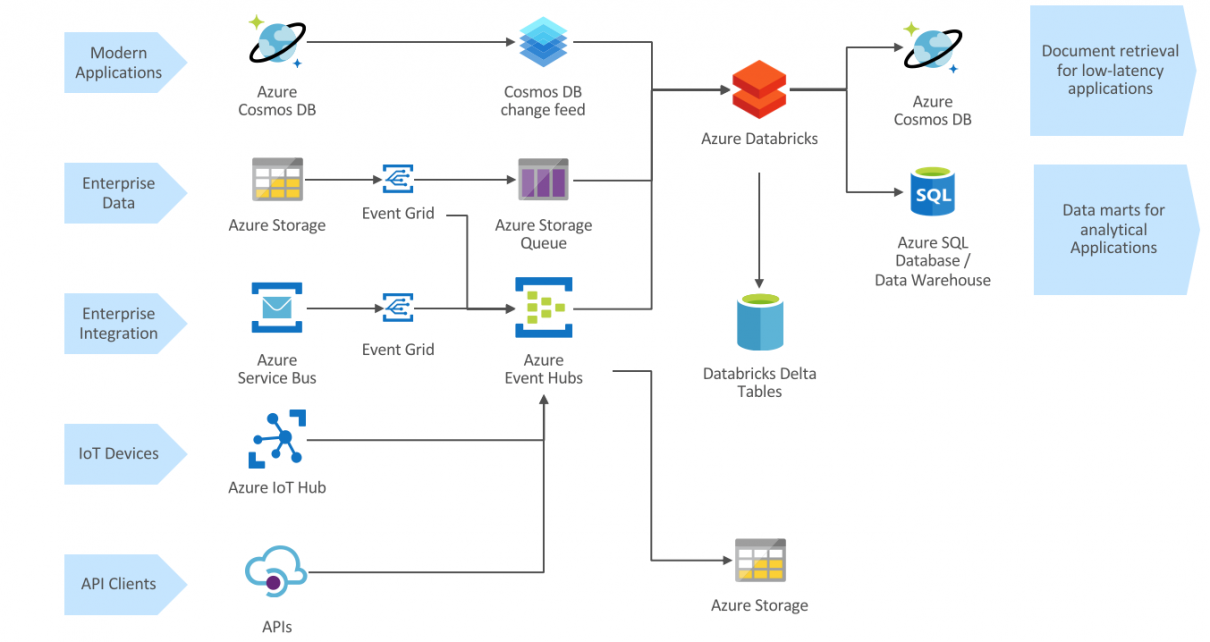
* In the architecture above, Azure Data Factory is the service responsible for data pipeline orchestration. Azure Databricks can also be used to perform the same role through the execution of nested notebooks.
* In the architecture above, Azure Stream Analytics is the service responsible for processing streaming data. Azure Databricks can also be used to perform the same role through the execution of notebooks.
* In the architecture above, Azure Databricks was used to invoke Cognitive Services. You can also make use of Azure Functions to invoke Azure Cognitive Services from an Azure Data Factory Pipeline.
* For comparisons of other alternatives, see:
  + [Choosing a data pipeline orchestration technology in Azure](https://docs.microsoft.com/en-us/azure/architecture/data-guide/technology-choices/pipeline-orchestration-data-movement)
  + [Choosing a batch processing technology in Azure](https://docs.microsoft.com/en-us/azure/architecture/data-guide/technology-choices/batch-processing)
  + [Choosing an analytical data store in Azure](https://docs.microsoft.com/en-us/azure/architecture/data-guide/technology-choices/analytical-data-stores)
  + [Choosing a data analytics technology in Azure](https://docs.microsoft.com/en-us/azure/architecture/data-guide/technology-choices/analysis-visualizations-reporting)

## Considerations

The technologies in this architecture were chosen because each of them provide the necessary functionality to handle the vast majority of data challenges in an organization. These services meet the requirements for scalability and availability, while helping them control costs.

* The [massively parallel processing architecture](https://docs.microsoft.com/en-us/azure/sql-data-warehouse/massively-parallel-processing-mpp-architecture) of Azure Synapse provides scalability and high performance.
* Azure Synapse has [guaranteed SLAs](https://azure.microsoft.com/support/legal/sla/sql-data-warehouse) and [recommended practices for achieving high availability](https://docs.microsoft.com/en-us/azure/sql-data-warehouse/sql-data-warehouse-best-practices).
* When analysis activity is low, the company can [scale Azure Synapse on demand](https://docs.microsoft.com/en-us/azure/sql-data-warehouse/sql-data-warehouse-manage-compute-overview), reducing or even pausing compute to lower costs. analysis-services-bcdr).
* The [Azure Synapse security model](https://docs.microsoft.com/en-us/azure/sql-data-warehouse/sql-data-warehouse-overview-manage-security) provides connection security, [authentication and authorization](https://docs.microsoft.com/en-us/azure/sql-data-warehouse/sql-data-warehouse-authentication) via Azure AD or SQL Server authentication, and encryption.

# Event-Based Analytical Data Processing With Azure Databricks

Table of Contents

* [Cloud-native streaming architecture](https://cloudarchitected.com/2019/03/event-based-analytical-data-processing-with-azure-databricks/#Cloud-native_streaming_architecture)
  + [Overview](https://cloudarchitected.com/2019/03/event-based-analytical-data-processing-with-azure-databricks/#Overview)
  + [Event Sourcing](https://cloudarchitected.com/2019/03/event-based-analytical-data-processing-with-azure-databricks/#Event_Sourcing)
  + [Structured Streaming](https://cloudarchitected.com/2019/03/event-based-analytical-data-processing-with-azure-databricks/#Structured_Streaming)
  + [Analytical engine](https://cloudarchitected.com/2019/03/event-based-analytical-data-processing-with-azure-databricks/#Analytical_engine)
  + [Data Delivery](https://cloudarchitected.com/2019/03/event-based-analytical-data-processing-with-azure-databricks/#Data_Delivery)
* [Event ingestion patterns](https://cloudarchitected.com/2019/03/event-based-analytical-data-processing-with-azure-databricks/#Event_ingestion_patterns)
  + [Data ingestion through Azure Storage](https://cloudarchitected.com/2019/03/event-based-analytical-data-processing-with-azure-databricks/#Data_ingestion_through_Azure_Storage)
  + [Event ingestion through Event Hubs or Cosmos DB](https://cloudarchitected.com/2019/03/event-based-analytical-data-processing-with-azure-databricks/#Event_ingestion_through_Event_Hubs_or_Cosmos_DB)
* [Analytical processing layer](https://cloudarchitected.com/2019/03/event-based-analytical-data-processing-with-azure-databricks/#Analytical_processing_layer)
* [Data delivery](https://cloudarchitected.com/2019/03/event-based-analytical-data-processing-with-azure-databricks/#Data_delivery)
* [Cost optimization](https://cloudarchitected.com/2019/03/event-based-analytical-data-processing-with-azure-databricks/#Cost_optimization)

## Cloud-native streaming architecture

### Overview

Modern data analytics architectures should embrace the high flexibility required for today’s business environment, where the only certainty for every enterprise is that the ability to harness explosive volumes of data in real time is emerging as a a key source of competitive advantage. Fortunately, cloud platforms allow high scalability and cost efficiency, provided cloud-native architectural patterns are implemented. At a minimum, an analytical solution should make no compromises on the following dimensions:

* **Push ingestion**: avoid the time and cost impact of polling data sources repeatedly for data, and the complexity of tracking what data was already processed.
* **Low latency**: ability to process data end-to-end in near-realtime. In cases where low latency is not initially needed, it is still usually desirable to have the ability to reduce latency in the future without having to rearchitect the system.
* **Cost efficiency**: architectures should be able to scale to thousands of data points per second or more, but also efficiently scale down during ramp-up or off periods. When high latency is acceptable, data should be processed using ephemeral compute infrastructure that is only spun a few times per day.
* **Fault tolerance**: this is beyond the ability to recover the state a single component. Implementing end-to-end guarantees requires the processing component to correctly react to upstream failures, and mitigate impact of its recovery on the consistency of downstream delivered data.
* **Single point of logic implementation**: architectures that separate event and batch processing, such as the famous Lambda architecture, require a duplication of logic and are fiendishly difficult to implement consistently.
* **Data flexibility**: the architecture should adapt to very different types of data, from lightweight events to large binary payloads that require custom code to be applied on ingestion.
* **Full-featured engine**: for maximum efficiency on the dimensions of cost, latency and complexity, the entire data processing should be performed in a single go. The data processing engine should therefore support the full set of features required for modern data analytics including: flexible parsing of data formats, analytical processing, complex event processing, machine learning, and native connectivity to a variety of sources and sinks.

Up to the recent past, implementing such requirements would mean setting up complex and disparate components on a platform such as Hadoop and dealing with significant complexity in implementation and maintenance. The use of cloud-native managed platforms largely eliminates that complexity.

One key to achieving those requirements consistently lies is standardizing data processing jobs to run in a streaming manner, and use a stateful stream processing engine such as Spark to process the data. We will detail how this can be achieved using PaaS components on Azure.

### Event Sourcing

Cloud-native components such as Blob Storage or Cosmos DB integrate the native ability to source events from data operations, and external source systems can usually be retrofitted to stream events directly into components such as Event Hub. Otherwise, you can use Blob Storage as an intermediate component. We cover event sourcing patterns in more detail below.

### Structured Streaming

[Spark Structured Streaming](https://docs.azuredatabricks.net/spark/latest/structured-streaming/index.html) is an open-source streaming engine that allows focusing on implementing business logic. The engine manages the essential technical capabilities including:

* Providing end-to-end reliability and correctness guarantees
* Performing complex transformations
* Handling late or out-of-order data
* Integrating with other systems

### Analytical engine

[Databricks Delta](https://docs.azuredatabricks.net/delta/index.html) is a next-generation unified analytics engine built on top of Apache Spark. Databricks Delta provides the components needed for an industrialised analytical engine, including ACID transactions, optimized data layouts and indexes, and features for stream processing into tables.

Thanks to fully featured Spark engine, advanced data transformations can be performed on the data at scale, such as analytical processing or machine learning.

### Data Delivery

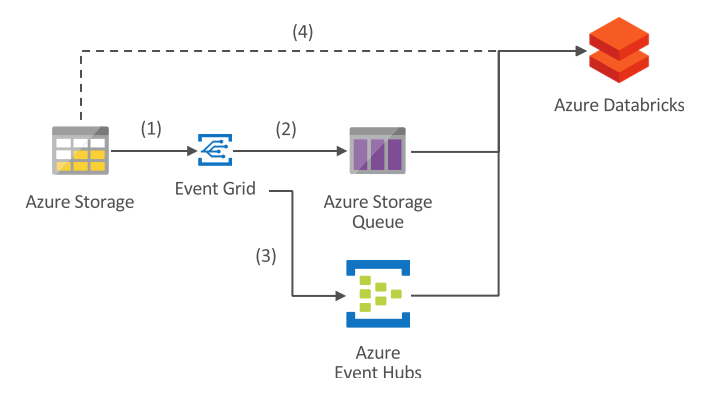
Although Databricks can expose query endpoints, the most effective way to deliver the data to consumers is usually to export it, for example to data marts of different forms optimized for specific access patterns. Databricks provide highly optimized connectors to deliver data in parallel at scale to enterprise database systems.

## Event ingestion patterns

### Data ingestion through Azure Storage

Data ingestion from Azure Storage is a highly flexible way of receiving data from a large variety of sources in structured or unstructured format. Azure Data Factory, Azure Logic Apps or third-party applications can deliver data from on-premises or cloud systems thanks to a large offering of connectors.

The ‘traditional’ approach to analytical data processing is to run batch processing jobs against data in storage at periodic interval. However, this is inefficient, as it requires repeated reading of storage metadata, and custom logic to handle duplication.

Event sourcing patterns from storage

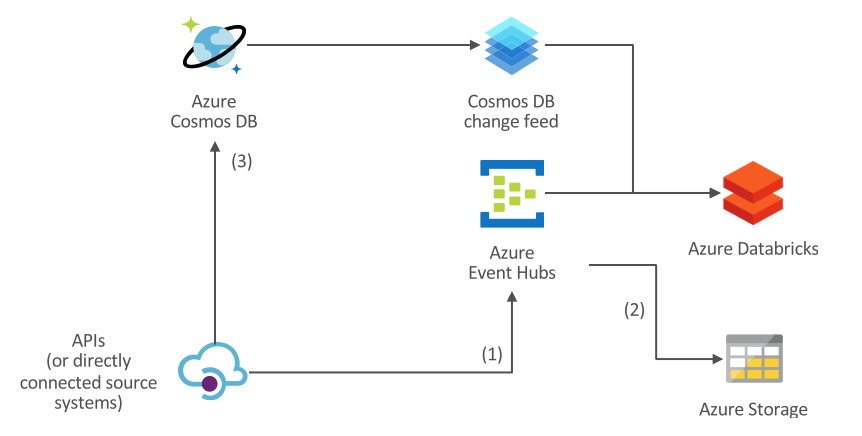
Azure Storage natively supports [event sourcing](https://docs.microsoft.com/en-us/azure/storage/blobs/storage-blob-event-overview), so that files written to storage can immediately trigger an event delivered into Azure Storage Queue or Event Hubs, marked by (1) in the image above.

* With Azure Storage Queue (2), you can use the [optimized ABS-AQS Databricks connector](https://docs.azuredatabricks.net/spark/latest/structured-streaming/aqs.html) to transparently consume the files from the storage source. The connector retrieves the file directly from storage and returns its content as binary.
* With Azure Event Hubs (3), you can use the [Azure Event Hubs Databricks connector](https://docs.azuredatabricks.net/spark/latest/structured-streaming/streaming-event-hubs.html) to retrieve the storage events. You can then access the storage account through a DBFS mount point, to retrieve the file content (4).

While processing storage events with Event Hubs is more complex than with the Queue Storage connector, the ability to write custom code for retrieving file data allows for writing custom logic. For example, if you need to retrieve only one element from a complex XML file, the ABS-AQS connector would retrieve the entire XML file into a Spark DataFrame that you would process later, while with the Event Hubs approach, you could extract the desired element while reading the data.

### Event ingestion through Event Hubs or Cosmos DB

Azure Event Hubs is a highly scalable and effective event ingestion and streaming platform, that can scale to millions of events per seconds. It is based around the same concepts as Apache Kafka, but available as a fully managed platform. It also offers a Kafka-compatible API for easy integration with third-party components.

Native event ingestion patterns

When source data can be represented and ingested in the form of streamed events, it is very compelling to streaming data directly to Event Hubs, either by connecting the source system directly or through an API layer (1). This allows low latency and high scalability. As Event Hubs only stores data for a limited time duration, the Event Hubs Capture mechanism can be used to store a copy of the events into permanent storage (2).

An compelling alternative design is to stream the events into Azure Cosmos DB (3). Through the Change Feed mechanism, Cosmos DB can effectively serve as an event sourcing feed. This is a very [effective solution](https://azure.microsoft.com/en-us/blog/the-emerging-big-data-architectural-pattern/) if the events are to be made available for online querying or as the glue layer of a serverless application.

## Analytical processing layer

Spark Structured Streaming is a highly scalable. Although streaming is most commonly used for low-latency, always-on processing, it also lends itself to [sporadic processing on ephemeral compute resources](https://databricks.com/blog/2017/05/22/running-streaming-jobs-day-10x-cost-savings.html) to reduce cost when higher latencies are sufficient.

Databricks Delta is a very efficient analytical engine for Spark, especially when used in streaming workloads. Some key features the engine provides are the ability to ingest streaming data directly into tables that are automatically managed and optimized, and the ability to use ‘[upserts](https://databricks.com/blog/2019/03/19/efficient-upserts-into-data-lakes-databricks-delta.html)‘ (SQL MERGE commands) which are an essential capability of stream processing workloads and ETL processes in general.

The core logic can be expressed as simply as creating a Delta table and a MERGE statement as below. This can be extended with only the business logic needed to process the data. For instance, a [Streaming Stock Data Analysis solution](https://databricks.com/blog/2018/07/19/simplify-streaming-stock-data-analysis-using-databricks-delta.html) can be implemented in a few dozen lines of code. Partitioning, data management, error recovery are all performed automatically.

CREATE TABLE events (  
   date DATE,  
   eventId STRING,  
   eventType STRING,  
   data STRING)  
 USING delta  
 PARTITIONED BY (date)

MERGE INTO events  
 USING updates  
 ON events.eventId = updates.eventId  
 WHEN MATCHED THEN  
   UPDATE SET  
     events.data = updates.data  
 WHEN NOT MATCHED  
   THEN INSERT (date, eventId, data) VALUES (date, eventId, data)

## Data delivery

If the data will be accessed by point queries or short range queries with low latency, Cosmos DB is an excellent platform to achieve low-millisecond latencies with global geodistribution. One scenario would be precomputing shopping recommendations for the users of a retail website, and making them available in milliseconds when the user logs in. The [Cosmos DB Spark Connector](https://docs.azuredatabricks.net/spark/latest/data-sources/azure/cosmosdb-connector.html) allows efficiently streaming large data volumes into (and out of) Cosmos DB.

When data will be retrieved through analytical processing engines, such as Business Intelligence applications, Azure SQL Database is an excellent platform. Here again, a [Spark connector](https://docs.azuredatabricks.net/spark/latest/data-sources/sql-databases-azure.html) allows efficient parallel writes to SQL Database. When scaling to many terabytes and an MPP SQL engine is needed to process complex analytical transformations or perform additional data processing, [Azure SQL Data Warehouse](https://docs.azuredatabricks.net/spark/latest/data-sources/azure/sql-data-warehouse.html) provides an alternative SQL engine.

# **LAMBDA ARCHITECTURE IN AZURE**

Lambda architecture is the state-of-the-industry, Big Data workload pattern for handling batch and streaming workloads in a single system. If you’re researching how to modernize your data program, the lambda architecture is the place to start.

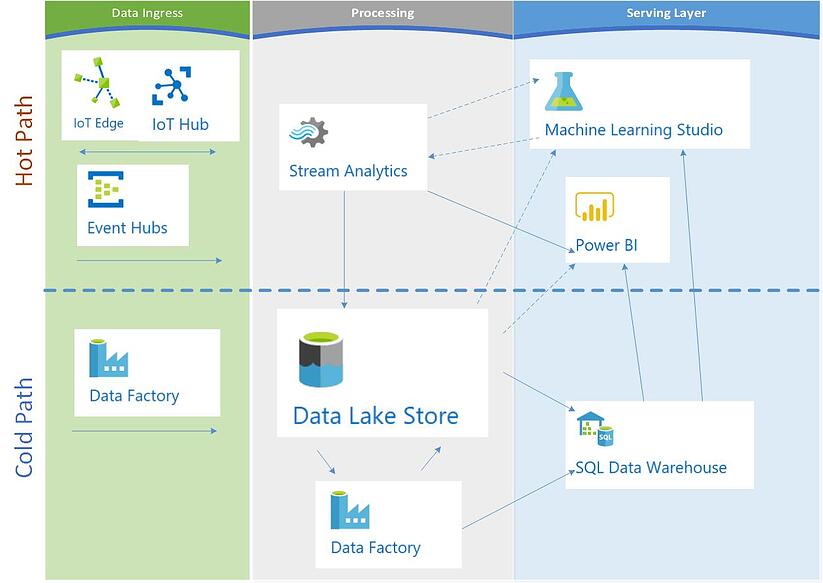
Let’s review the key concepts, parse through the tooling options in [Microsoft Azure](https://azure.microsoft.com/en-us/), examine some sample reference architectures, and discuss common criticisms of lambda. In a follow-up post, we’ll introduce the emerging kappa architecture and compare the benefits and limitations against lambda.

## Lambda Architecture Overview

The key components of the lambda architecture are the **hot** and **cold** data processing paths, and a common serving layer that combines outputs for both paths. The **hot path** refers to streaming data workloads and the **cold path** applies to batch-processed data. The goal of the architecture is to present a holistic view of an organization’s data, both from history and in near real-time, within a combined serving layer, as the following [Microsoft](https://blogs.msdn.microsoft.com/uk_faculty_connection/2017/02/24/big-data-on-azure-with-no-limits-data-analytics-and-managed-clusters/) visual illustrates.

|  |
| --- |
| [Lambda Architecture Hot Cold Path](https://blogs.msdn.microsoft.com/uk_faculty_connection/2017/02/24/big-data-on-azure-with-no-limits-data-analytics-and-managed-clusters/) |

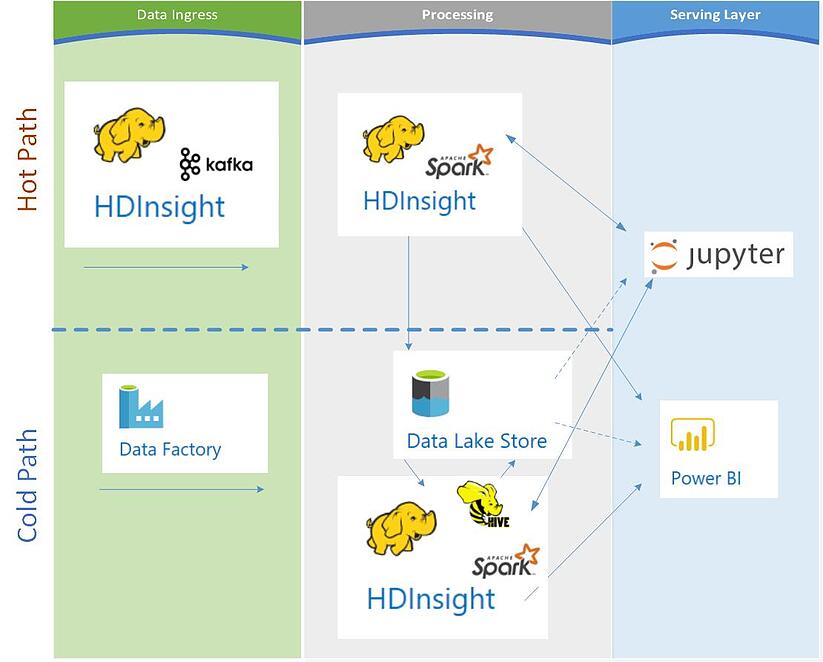
Unless you’re outside of the current capabilities of a service (or will be soon), then the managed service architecture is the best place to start.



There are lots of options for augmenting, substituting, and extending this architecture. The goal here is to give a baseline of what you would probably need in your ecosystem to support lambda if preferring native Azure services.

### Reference Architecture for HDInsight

Below is an implementation with preference for [HDInsight services](https://azure.microsoft.com/en-us/services/hdinsight/):

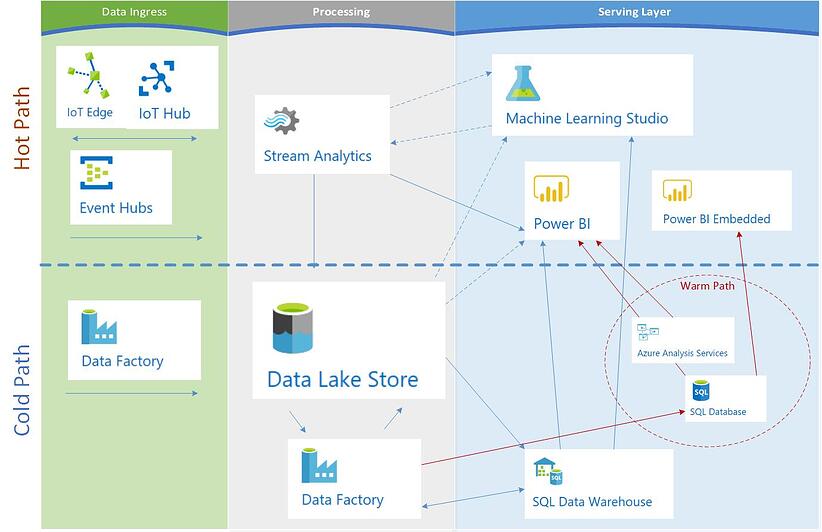


You’ll notice that we’ve listed [Azure Data Factory](https://azure.microsoft.com/en-us/services/data-factory/) (ADF) as the ingest engine for batch. The data movement, lineage, monitoring, and orchestration capabilities of ADF are extremely difficult to substitute for in the Azure cloud. Even running HDInsight jobs as Data Factory linked services automatically handles the spin up and tear down of clusters for you. So, while you have lots of options as far as analytics, machine learning, querying, and compute services, if you’re considering any type of Big Data workload in Azure, then planning on ADF as part of your architecture will simplify and accelerate your development cycle.

## Limitations of Lambda

### Extending for concurrency and frequency

An assumption that you often see with lambda, as modeled above, is low concurrency and/or frequency, specifically in the cold path. This does not fit many large organizations’ internal needs, or even those of small organizations offering reporting and analytics to their end customers. In practice, the serving layer is usually extended to include a hub-and-spoke architecture that incorporates a structured data mart to support the most commonly queried data (either by partition or entity). I personally don’t know of a common name for this, but I like to think of it as the**warm path**. We are purposely prioritizing some batch-processed data into services that support higher concurrency at a lower cost. Over time, the partitions are aged out of the warm path, but persist within the cold query path. This augmentation to lambda also helps simplify multi-tenancy, self-service BI, and embedded analytics use cases.



## Common Criticisms

Lambda is an organic result of the limitations of existing tools. Distributed systems architects and developers commonly criticize its complexity – and rightly so. Those of us that have worked extensively in Extract-Transform-Load and symmetric multiprocessing systems see red flags when code is replicated in multiple services. Ensuring data quality and code conformity across multiple systems, whether massively parallel processing (MPP) or symmetrically parallel system (SMP), has the same best practice: **the least amount of times you reproduce code is always the correct number of times.**

We reproduce code in lambda because different services in MPP systems are better at different tasks. The maturity of tools historically hasn’t allowed us to process streams and batch in a single tool. This is starting to change, with [Apache Spark](https://spark.apache.org/) emerging as a single preferred compute service for stream and batch querying, hence the timing of [Azure Databricks](https://databricks.com/product/azure). However, on the storage side, what was meant to be an immutable store that is the data lake in practice, can become the dreaded swamp when governance or testing fails; which is not uncommon. A fundamentally different assumption to how we process data is required to combat this degradation. Enter: the kappa architecture, which we’ll examine in the next post of this series.

