* **What is delta lake and its advantages?**

**data lakes have the following challenges:**

1. **Reading and writing into data lakes is not reliable.**
2. **The data quality in data lakes is low.**
3. **Poor performance with increasing amounts of data.**
4. **Updating records in data lakes is hard.**

**Delta Lake is an additional storage layer that brings reliability to your data lakes built on HDFS and cloud storage by providing**

1. **ACID transactions through optimistic concurrency control between writes and**
2. **snapshot isolation for consistent reads during writes.**
3. **Delta Lake also provides built-in data versioning for easy rollbacks and reproducing reports**
4. **Schema Enforcement: Data Lake helps to avoid bad data getting your data lakes by providing the ability to specify the schema and help enforce it.It prevents data corruption by preventing the bad data get into the system even before the data is ingested into the data lake by giving sensible error messages.**
5. **Delta Lake acts on top of your existing data lake, not replacing data lakes.**
6. **Delta Lake uses versioned Parquet files to store your data in your cloud storage.**
7. **Apart from the versions, Delta Lake also stores a transaction log to keep track of all the commits made to the table or blob store directory to provide ACID transactions**

**Let us try to understand ACID failure in spark with the above scenario.**

**A in ACID stands for Atomicity,**

* **What is Atomicity: Either all changes take place or none, the system is never in halfway state.**
* **How spark fails: While writing data, (at Line 4 above), if a failure occurs at a stage where old data is removed and new data is not yet written, data loss occurs. We have lost old data and we were not able to write new data due to job failure, atomicity fails. [It can vary according to file output committer used, please do read about File output committer to see how data writing takes place, the scenario I explained is for v2]**

**C in ACID stands for Consistency,**

* **What is Consistency: Data must be consistent and valid in the system at all times.**
* **How Spark fails: As seen above, in the case of failure and data loss, we are left with invalid data in the system, consistency fails.**

**I in ACID stands for Isolation,**

* **What is Isolation: Multiple transactions occur in isolation**
* **How spark fails: Consider two jobs running in parallel, one as described above and another which is also using the same dataset, if one job overwrites the dataset while other is still using it, failure might happen, isolation fails.**

**D in ACID stands for Durability,**

* **What is Durability: Changes once made are never lost, even in the case of system failure.**
* **How spark might fail: Spark really doesn’t affect the durability, it is mainly governed by the storage layer, but since we are losing data in case of job failures, in my opinion, it is a durability failure.**

**Sample delta statements:**

**events.write.partitionBy("date").format("delta").save("/mnt/delta/events")**

**spark.sql("CREATE TABLE events USING DELTA LOCATION '/mnt/delta/events/'")**

**Streama:**

**from pyspark.sql.types import \***

**inputPath = "/databricks-datasets/structured-streaming/events/"**

**jsonSchema = StructType([ StructField("time", TimestampType(), True), StructField("action", StringType(), True) ])**

**eventsDF = (**

**spark**

**.readStream**

**.schema(jsonSchema) # Set the schema of the JSON data**

**.option("maxFilesPerTrigger", 1) # Treat a sequence of files as a stream by picking one file at a time**

**.json(inputPath)**

**)**

**(eventsDF.writeStream**

**.outputMode("append")**

**.option("checkpointLocation", "/mnt/delta/events/\_checkpoints/etl-from-json")**

**.table("events")**

## **Batch upserts**

**To merge a set of updates and insertions into an existing table, you use the MERGE INTO statement. For example, the following statement takes a stream of updates and merges it into the events table. When there is already an event present with the same eventId, Delta Lake updates the data column using the given expression. When there is no matching event, Delta Lake adds a new row.**

**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)**

### **Query an earlier version of the table (time travel)**

**Delta Lake time travel allows you to query an older snapshot of a Delta table.For timestamp\_string, only date or timestamp strings are accepted. For example, "2019-01-01" and "2019-01-01'T'00:00:00.000Z".**

**To query an older version of a table, specify a version or timestamp in a SELECT statement. For example, to query version 0 from the history above, use:**

**SELECT \* FROM events VERSION AS OF 0**

**SELECT \* FROM events TIMESTAMP AS OF '2019-01-29 00:37:58'**

**df1 = spark.read.format("delta").option("timestampAsOf", timestamp\_string).load("/mnt/delta/events")**

**df2 = spark.read.format("delta").option("versionAsOf", version).load("/mnt/delta/events")**

## **Optimize a table**

**Once you have performed multiple changes to a table, you might have a lot of small files. To improve the speed of read queries, you can use OPTIMIZE to collapse small files into larger ones:**

**OPTIMIZE delta.`/mnt/delta/events`**

**OPTIMIZE events**

### **Z-order by columns**

**To improve read performance further, you can co-locate related information in the same set of files by Z-Ordering. This co-locality is automatically used by Delta Lake data-skipping algorithms to dramatically reduce the amount of data that needs to be read. To Z-Order data, you specify the columns to order on in the ZORDER BY clause. For example, to co-locate by eventType, run:**

**OPTIMIZE events**

**ZORDER BY (eventType)**

## **Clean up snapshots**

**Delta Lake provides snapshot isolation for reads, which means that it is safe to run OPTIMIZE even while other users or jobs are querying the table. Eventually however, you should clean up old snapshots. You can do this by running the VACUUM command:**

**VACUUM events**

**You control the age of the latest retained snapshot by using the RETAIN <N> HOURS option:**

**VACUUM events RETAIN 24 HOURS**

# **What is azure databricks?**

**Azure Databricks is an Apache Spark-based analytics service optimized for the MS Azure cloud services platform is designed with the originators of Apache Spark. Moreover, it is associated with MS Azure for a one-click setup and an interactive working space that eases collaboration within data scientists, data engineers, and business analysts.**

# **Databricks runtimes**

**Databricks runtimes are the set of core components that run on Databricks**[**clusters**](https://docs.databricks.com/clusters/index.html)**. Databricks offers several types of runtimes.**

# **Workspace**

**A Databricks workspace is an environment for accessing all of your Databricks assets. The workspace organizes objects (**[**notebooks**](https://docs.databricks.com/workspace/workspace-assets.html#ws-notebooks)**,**[**libraries**](https://docs.databricks.com/workspace/workspace-assets.html#ws-libraries)**, and**[**experiments**](https://docs.databricks.com/applications/mlflow/tracking.html#mlflow-experiments)**) into**[**folders**](https://docs.databricks.com/workspace/workspace-objects.html#folders)**, and provides access to**[**data**](https://docs.databricks.com/data/index.html)**and computational resources such as**[**clusters**](https://docs.databricks.com/workspace/workspace-assets.html#ws-clusters)**and**[**jobs**](https://docs.databricks.com/workspace/workspace-assets.html#ws-jobs)**.**

# **Clusters**

**A Databricks cluster is a set of computation resources and configurations on which you run data engineering, data science, and data analytics workloads, such as production ETL pipelines, streaming analytics, ad-hoc analytics, and machine learning.**

**Databricks Utilities**

**Databricks Utilities (DBUtils) make it easy to perform powerful combinations of tasks. You can use the utilities to work with object storage efficiently, to chain and parameterize notebooks, and to work with secrets. DBUtils are not supported outside of notebooks.**

**All dbutils utilities are available in Python, R, and Scala notebooks.**[**File system utilities**](https://docs.databricks.com/dev-tools/databricks-utils.html#dbutils-fs)**are not available in R notebooks; however, you can use a**[**language magic command**](https://docs.databricks.com/notebooks/notebooks-use.html#language-magic)**to invoke those dbutils methods in R and SQL notebooks. For example, to list the [Databricks datasets](https://docs.databricks.com/data/databricks-datasets.html" \l "databricks-datasets) DBFS folder in an R or SQL notebook, run the command:**

# **Databricks File System (DBFS)**

**Databricks File System (DBFS) is a distributed file system mounted into a Databricks workspace and available on Databricks clusters. DBFS is an abstraction on top of scalable object storage and offers the following benefits:**

* **Allows you to**[**mount**](https://docs.databricks.com/data/databricks-file-system.html#mount-storage)**storage objects so that you can seamlessly access data without requiring credentials.**
* **Allows you to interact with object storage using directory and file semantics instead of storage URLs.**
* **Persists files to object storage, so you won’t lose data after you terminate a cluster.**

## **DBFS root**

**The default storage location in DBFS is known as the DBFS root. Several types of data are stored in the following DBFS root locations:**

* **/FileStore: Imported data files, generated plots, and uploaded libraries. See**[**Special DBFS root locations**](https://docs.databricks.com/data/databricks-file-system.html#special-dbfs-root-locations)**.**
* **/databricks-datasets: Sample public datasets. See**[**Special DBFS root locations**](https://docs.databricks.com/data/databricks-file-system.html#special-dbfs-root-locations)**.**
* **/databricks-results: Files generated by downloading the**[**full results**](https://docs.databricks.com/notebooks/notebooks-use.html#download-full-results)**of a query.**
* **/databricks/init: Global and cluster-named (deprecated) [init scripts](https://docs.databricks.com/clusters/init-scripts.html).**
* **/user/hive/warehouse: Data and metadata for non-external Hive tables.**

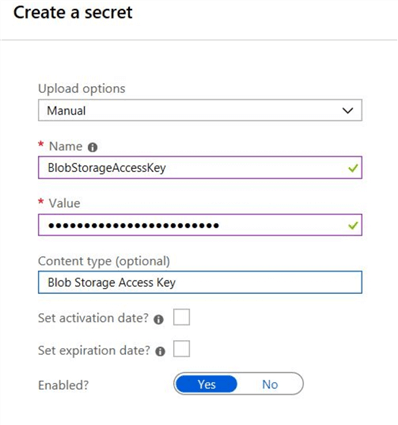
# **How to access azure keyvault secrets in databricks?**

**Azure Databricks offers Secret Management. Secret Management allows users to share credentials in a secure mechanism. Currently Azure Databricks offers two types of Secret Scopes:**

* **Azure Key Vault-backed: To reference secrets stored in an Azure Key Vault, you can create a secret scope backed by Azure Key Vault. Azure Key Vault-backed secrets are only supported for Azure Databricks Premium Plan.**
* **Databricks-backed: A Databricks-backed scope is stored in (backed by) an Azure Databricks database. You create a Databricks-backed secret scope using the Databricks CLI (version 0.7.1 and above).**

**Azure Key Vault-backed procedure**

**1.Creating Secret in Azure Key Vault**

****

**2. Creating Azure Key Vault Secret Scope in Databricks**

## **Azure Databricks - Creating the Azure Key Vault backed secret scope.**

## **3.Using Azure Key Vault Secret Scope and Secret in Azure Databricks Notebook**

**dbutils.secrets.get(scope = "azurekeyvault\_secret\_scope", key = "BlobStorageAccessKey")**

**#azurekeyvault\_secret\_scope --> Azure Key Vault based scope which we created in Databricks**

**#BlobStorageAccessKey --> Secret name which we created in Azure Key Vault**

**How to access Access ADLS\ Azure Bob Storage from Data Bricks Notebooks**

**Data Bricks Notebooks can access ADLS\Blob storage files either by mounting storage using Data bricks DBFS file system or directly by using spark API.**

**If storage is mounted in Databricks DBFS, all users in the Databricks workspace can access to the mounted Azure Data Lake Storage Gen1 account. Once Mounted can access in all notebooks by all users.**

**If there is requirement to access only once then we can access the ADLS\Blob storage using direct Spark API providing connection details to spark configuration.**

**Procedure 1: Mounting ADLS/Blob storage in Databricks DBFS**

**Mounting ADLS in Databricks DBFS:**

[**DBFS**](https://docs.azuredatabricks.net/user-guide/dbfs-databricks-file-system.html)**mount points let you mount Azure Data Lake Store for all users in the workspace. Once it is mounted, the data can be accessed directly via a DBFS path from all clusters, without the need for providing credentials every time**

**Scala**

**val configs = Map(**

**"dfs.adls.oauth2.access.token.provider.type" -> "ClientCredential",**

**"dfs.adls.oauth2.client.id" -> "<your-service-client-id>",**

**"dfs.adls.oauth2.credential" -> dbutils.secrets.get(scope = "<scope-name>", key = "<key-name>"),**

**"dfs.adls.oauth2.refresh.url" -> "https://login.microsoftonline.com/<your-directory-id>/oauth2/token")**

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

**dbutils.fs.mount(**

**source = "adl://<your-data-lake-store-account-name>.azuredatalakestore.net/<your-directory-name>",**

**mountPoint = "/mnt/<mount-name>",**

**extraConfigs = configs)**

**Python**

**configs = {"dfs.adls.oauth2.access.token.provider.type": "ClientCredential",**

**"dfs.adls.oauth2.client.id": "<your-service-client-id>",**

**"dfs.adls.oauth2.credential": dbutils.secrets.get(scope = "<scope-name>", key = "<key-name>"),**

**"dfs.adls.oauth2.refresh.url": "https://login.microsoftonline.com/<your-directory-id>/oauth2/token"}**

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

**dbutils.fs.mount(**

**source = "adl://<your-data-lake-store-account-name>.azuredatalakestore.net/<your-directory-name>",**

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

**extra\_configs = configs)**

* **<mount-name> is a DBFS path that represents where the Azure Data Lake Storage Gen1 account or a folder inside it (specified in source) will be mounted in DBFS.**
* **dbutils.secrets.get(scope = "<scope-name>", key = "<key-name>") retrieves your service credential that has been stored as a**[**secret**](https://docs.databricks.com/user-guide/secrets/secrets.html#secrets)**in a**[**secret scope**](https://docs.databricks.com/user-guide/secrets/secret-scopes.html#secret-scopes)**.**

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

**Scala**

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

**Python**

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

**Mounting BLOB Storage in Databricks DBFS:**

**Scala**

**dbutils.fs.mount(**

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

**mountPoint = "/mnt/<mount-name>",**

**extraConfigs = Map("<conf-key>" -> dbutils.secrets.get(scope = "<scope-name>", key = "<key-name>")))**

**Python**

**dbutils.fs.mount(**

**source = "wasbs://<your-container-name>@<your-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>")})**

* + **<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.<your-storage-account-name>.blob.core.windows.net or fs.azure.sas.<your-container-name>.<your-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/user-guide/secrets/secrets.html#secrets)**in a**[**secret scope**](https://docs.databricks.com/user-guide/secrets/secret-scopes.html#secret-scopes)**.**

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

**Scala**

**// scala**

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

**Python**

**# python**

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

**Procedure 2 : Accessing ADLS\Azure storage directly by Spark API**

**Spark api need to use oAuth2 protocol to access for a azure resources**

**With Spark configs, the Azure Data Lake Store settings can be specified per notebook. To keep things simple, we can provide credentials in plaintext. However, it is strongly discourage from storing secrets in plaintext. Instead, recommend storing the credentials as Data bricks.**

**Accessing ADLS directly by spark 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**

**spark.conf.set("dfs.adls.oauth2.access.token.provider.type", "ClientCredential")**

**spark.conf.set("dfs.adls.oauth2.client.id", "<your-service-client-id>")**

**spark.conf.set("dfs.adls.oauth2.credential", dbutils.secrets.get(scope = "<scope-name>", key = "<key-name>"))**

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

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

**After providing credentials, you can read from Azure Data Lake Storage Gen1 using Spark and Databricks APIs:**

**val df = spark.read.parquet("adl://<your-data-lake-store-account-name>.azuredatalakestore.net/<your-directory-name>")**

**dbutils.fs.ls("adl://<your-data-lake-store-account-name>.azuredatalakestore.net/<your-directory-name>")**

**Azure Data Lake Storage Gen1 provides directory level access control, so the service principal must have access to the directories that you want to read from as well as the Data Lake Store resource.**

[**Access Azure Blob Storage directly**](https://docs.databricks.com/spark/latest/data-sources/azure/azure-storage.html#id4)

**You can read data from Azure Blob Storage using the Spark API and Databricks APIs:**

**Set up an account access key:**

**spark.conf.set(**

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

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

* **Set up a SAS for a container:**

**spark.conf.set(**

**"fs.azure.sas.<your-container-name>.<your-storage-account-name>.blob.core.windows.net",**

**"<complete-query-string-of-your-sas-for-the-container>")**

**Once an account access key or a SAS is set up in your notebook, you can use standard Spark and Databricks APIs to read from the storage account:**

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

**dbutils.fs.ls("wasbs://<your-container-name>@<your-storage-account-name>.blob.core.windows.net/<your-directory-name>")**

**Azure Data Factory**

**Azure Data Factory is the platform that solves such data scenarios. It is a cloud-based data integration service that allows you to create data-driven workflows in the cloud for orchestrating and automating data movement and data transformation. Using Azure Data Factory, you can create and schedule 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 Azure HDInsight Hadoop, Spark, Azure Data Lake Analytics, and Azure Machine Learning.**

**Additionally, you can publish output data to data stores such as Azure SQL Data Warehouse for business intelligence (BI) applications to consume. Ultimately, through Azure Data Factory, raw data can be organized into meaningful data stores and data lakes for better business decisions.**

## **How does it work?**

**The pipelines (data-driven workflows) in Azure Data Factory typically perform the following four steps:**

### **Connect and collect**

**Enterprises have data of various types that are located in disparate sources on-premises, in the cloud, structured, unstructured, and semi-structured, all arriving at different intervals and speeds.**

**The first step in building an information production system is to connect to all the required sources of data and processing, such as software-as-a-service (SaaS) services, databases, file shares, and FTP web services. The next step is to move the data as needed to a centralized location for subsequent processing.**

**Without Data Factory, enterprises must build custom data movement components or write custom services to integrate these data sources and processing. It's expensive and hard to integrate and maintain such systems. In addition, they often lack the enterprise-grade monitoring, alerting, and the controls that a fully managed service can offer.**

**With Data Factory, you can use the**[**Copy Activity**](https://docs.microsoft.com/en-us/azure/data-factory/copy-activity-overview)**in a data pipeline to move data from both on-premises and cloud source data stores to a centralization data store in the cloud for further analysis. For example, you can collect data in Azure Data Lake Store and transform the data later by using an Azure Data Lake Analytics compute service. You can also collect data in Azure Blob storage and transform it later by using an Azure HDInsight Hadoop cluster.**

### **Transform and enrich**

**After data is present in a centralized data store in the cloud, process or transform the collected data by using compute services such as HDInsight Hadoop, Spark, Data Lake Analytics, and Machine Learning. You want to reliably produce transformed data on a maintainable and controlled schedule to feed production environments with trusted data.**

### **Publish**

**After the raw data has been refined into a business-ready consumable form, load the data into Azure Data Warehouse, Azure SQL Database, Azure CosmosDB, or whichever analytics engine your business users can point to from their business intelligence tools.**

### **Monitor**

**After you have successfully built and deployed your data integration pipeline, providing business value from refined data, monitor the scheduled activities and pipelines for success and failure rates. Azure Data Factory has built-in support for pipeline monitoring via Azure Monitor, API, PowerShell, Log Analytics, and health panels on the Azure portal.**

## **Top-level concepts**

**An Azure subscription might have one or more Azure Data Factory instances (or data factories). Azure Data Factory is composed of four key components. These components work together to provide the platform on which you can compose data-driven workflows with steps to move and transform data.**

### **Pipeline**

**A data factory might have one or more pipelines. A pipeline is a logical grouping of activities that performs a unit of work. Together, the activities in a pipeline perform a task. For example, a pipeline can contain a group of activities that ingests data from an Azure blob, and then runs a Hive query on an HDInsight cluster to partition the data.**

**The benefit of this is that the pipeline allows you to manage the activities as a set instead of managing each one individually. The activities in a pipeline can be chained together to operate sequentially, or they can operate independently in parallel.**

### **Activity**

**Activities represent a processing step in a pipeline. For example, you might use a copy activity to copy data from one data store to another data store. Similarly, you might use a Hive activity, which runs a Hive query on an Azure HDInsight cluster, to transform or analyze your data. Data Factory supports three types of activities: data movement activities, data transformation activities, and control activities.**

### **Datasets**

**Datasets represent data structures within the data stores, which simply point to or reference the data you want to use in your activities as inputs or outputs.**

### **Linked services**

**Linked services are much like connection strings, which define the connection information that's needed for Data Factory to connect to external resources. Think of it this way: a linked service defines the connection to the data source, and a dataset represents the structure of the data. For example, an Azure Storage-linked service specifies a connection string to connect to the Azure Storage account. Additionally, an Azure blob dataset specifies the blob container and the folder that contains the data.**

**Linked services are used for two purposes in Data Factory:**

* **To represent a data store that includes, but isn't limited to, an on-premises SQL Server database, Oracle database, file share, or Azure blob storage account. For a list of supported data stores, see the**[**copy activity**](https://docs.microsoft.com/en-us/azure/data-factory/copy-activity-overview)**article.**
* **To represent a compute resource that can host the execution of an activity. For example, the HDInsightHive activity runs on an HDInsight Hadoop cluster. For a list of transformation activities and supported compute environments, see the**[**transform data**](https://docs.microsoft.com/en-us/azure/data-factory/transform-data)**article.**

### **Triggers**

**Triggers represent the unit of processing that determines when a pipeline execution needs to be kicked off. There are different types of triggers for different types of events.**

### **Pipeline runs**

**A pipeline run is an instance of the pipeline execution. Pipeline runs are typically instantiated by passing the arguments to the parameters that are defined in pipelines. The arguments can be passed manually or within the trigger definition.**

### **Parameters**

**Parameters are key-value pairs of read-only configuration.  Parameters are defined in the pipeline. The arguments for the defined parameters are passed during execution from the run context that was created by a trigger or a pipeline that was executed manually. Activities within the pipeline consume the parameter values.**

**A dataset is a strongly typed parameter and a reusable/referenceable entity. An activity can reference datasets and can consume the properties that are defined in the dataset definition.**

**A linked service is also a strongly typed parameter that contains the connection information to either a data store or a compute environment. It is also a reusable/referenceable entity.**

### **Control flow**

**Control flow is an orchestration of pipeline activities that includes chaining activities in a sequence, branching, defining parameters at the pipeline level, and passing arguments while invoking the pipeline on-demand or from a trigger. It also includes custom-state passing and looping containers, that is, For-each iterators.**

**What is Azure Storage?**

**Azure Storage enables you to store terabytes of data to support small to big data use cases. It is highly scalable, highly available, and can handle millions of requests per second on average. Azure Blob Storage is one of the types of services provided by Azure Storage.**

**Azure provides two key types of storage for unstructured data: Azure Blob Storage and Azure Data Lake Store.**

**Azure Blob Storage**

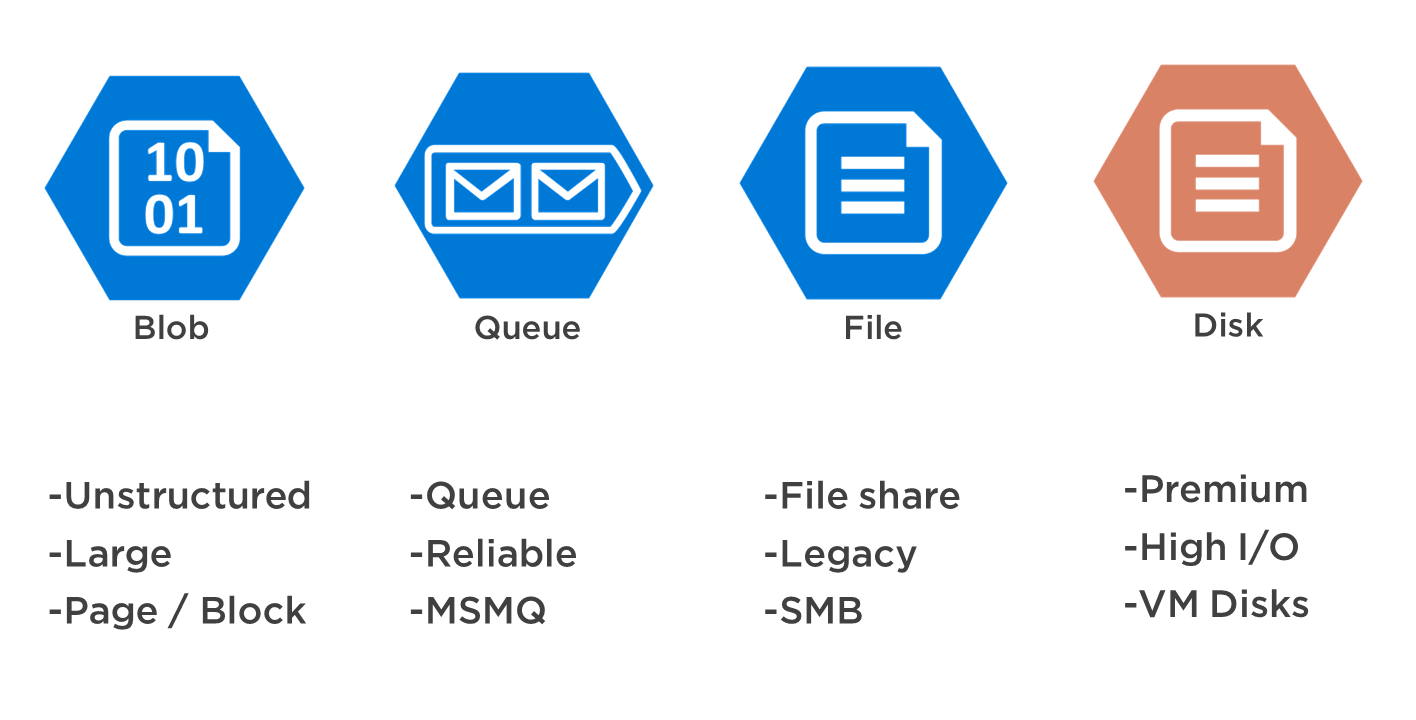
**Azure Blob Storage stores unstructured object data. A blob can be any type of text or binary data, such as a document or media file.**

**Structure : Object store with flat namespace**

**Blob storage is also referred to as object storage**

**Azure Storage**

**If you need to store files and small rows of data at large scale, without advanced query capabilities,**[**Azure Storage**](https://azure.microsoft.com/services/storage/)**is your best bet.**

****

**Azure Storage consists out of multiple services that are each optimized for a certain usage scenario. They are described in this post, and here is a summary of them:**

* [**Azure Blob Storage**](https://azure.microsoft.com/services/storage/blobs/)
  + **Useful for storing files, small and large, like audio, video or VHD files**
* [**Azure Queue Storage**](https://azure.microsoft.com/services/storage/queues/)
  + **Meant for storing small messages that are picked up by other applications. Queue Storage can help to decouple your applications**
* [**Azure File Storage**](https://azure.microsoft.com/services/storage/files/)
  + **Based on the SMB protocol, File Storage is meant to be mounted as a disk in a VM. It is very useful to use for lifting and shifting applications into the cloud**
* [**Azure Disk Storage**](https://azure.microsoft.com/services/storage/premium-storage/)
  + **Disk Storage is optimized for high I/O operations and can be used as a hard disk for a VM, like a server**

**All of these services share common features, like data encryption at rest and authentication and authorization. Additionally, by default, everything you store in Azure Storage is**[**replicated**](https://docs.microsoft.com/azure/storage/common/storage-redundancy)**3 times within the chosen data center. You can choose to also have the storage replicate to another data center using the Geo-redundant storage tier, which then also creates 3 copies of your data in a secondary data center, making the total number of copies 6. This ensures that your data is safe in case of failure or disaster.**

**Azure Data Lake Store (ADLS V1)**

**Another store that is optimized for storing large amounts of data for reporting and analytical purposes is the**[**Azure Data Lake Store**](https://azure.microsoft.com/services/data-lake-store/)**. The Data Lake store is geared towards storing data in its native format, making it a great store for non-relational data.**

**The Data Lake Store works differently from SQL Data Warehouse in that you don’t define a data schema up front where you do need to do that with a SQL Data Warehouse. You store data in whatever form it comes in and define a schema when you retrieve it.**

**The Data Lake Store is a place where you put large amounts of data that you want to analyze for information. For when you don’t yet know which questions you want the data to answer. Once you know the questions to ask and the format of the answer, you can transform the data in the Data Lake into more structured data with tools like**[**Azure Data Lake Analytics**](https://azure.microsoft.com/services/data-lake-analytics/)**,**[**Azure HDInsight**](https://azure.microsoft.com/services/hdinsight/)**or**[**Azure Analysis Services**](https://azure.microsoft.com/services/analysis-services/)**and move the structured data into Azure SQL Data warehouse to be visualized by tools like [PowerBI](https://powerbi.microsoft.com/" \t "_blank).**

**Azure Data Lake Store is the data store that you use for large amounts of semi- and unstructured, non-relational data that you need to process at scale.**

**Azure Data Lake Store provides what enterprises look for in storage today and it:**

**Provides additional enterprise-grade security features like encryption and uses Azure Active Directory for authentication and authorization.**

**Is compatible with Hadoop Distributed File System (HDFS) and works with the Hadoop ecosystem including Azure HDInsight.**

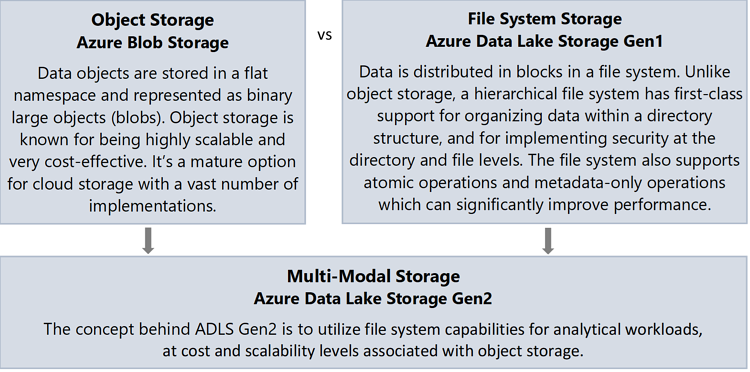
**Structure : xHierarchical file system**

**Includes Azure HDInsight clusters, which can be provisioned and configured to directly access data stored in Data Lake Store.**

**Allows data stored in Data Lake Store to be easily analyzed using Hadoop analytic frameworks such as MapReduce, Spark, or Hive.**

**Azure Data Lake Store (ADLS V2)**

**Fundamentally, ADLS Gen2 is seeking to take advantage of file system benefits without giving up the type of scalability and cost-effectiveness available with an object store:**

****

## **What is Azure Data Factory?**

**Data Factory is a fully managed, cloud-based, data-integration service that automates the movement and transformation of data. Like a factory that runs equipment to transform raw materials into finished goods, Azure Data Factory orchestrates existing services that collect raw data and transform it into ready-to-use information.**

**By using Azure Data Factory, you can create data-driven workflows to move data between on-premises and cloud data stores. And you can process and transform data by using compute services such as Azure HDInsight, Azure Data Lake Analytics, and the SQL Server Integration Services (SSIS) integration runtime**

### **Azure Data Factory Following Activities**

**1. First of all, it is a cloud based solution where it can integrate with different types of data stores to gather information or data.**

**2. It helps you to create data driven workflows to execute the same**

**3. All the data driven workflows are called “pipelines”.**

**4. Once the data is gathered, processing tools like Azure HDInsight Hadoop, Spark, Azure Data Lake Analytics can be used where the data can be transformed and can be pass to the BI professionals where they can analyze the data.**

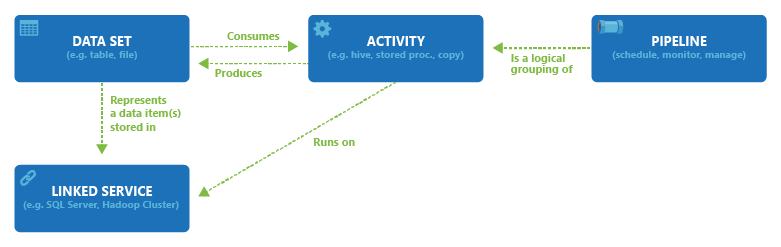
**Workflow In Depth:**

**As we have discussed, a pipeline is nothing but a data-driven workflow where in Azure Data Factory it is executed in three simple steps, they are:**

**1. Connect and Collect**

**2. Transform and Enrich**

**3. Publish**

****

**Connect and Collect:**

**When it comes to data storage, especially in enterprises, a variety of data stores are utilized to store the data. The first and foremost step in the building an Information production system is to connect all the required sources of the data, such as like Saas services, file shares, FTP, web services so that the data can be pushed to a centralized location for data processing.**

**Without a proper data factor, the organizations have to build or develop a custom data movement components so that the data sources can be integrated. This is an expensive affair without the use of Data Factory.**

**Transform and Enrich:**

**As completing the connect and collect phase, the next phase is to transform the data and massage it to a level where the reporting layer can be utilized and harvest the data and generate respective analyzed reports.**

**Tools like Data Lake Analytics and**[**Machine learning**](https://mindmajix.com/azure-machine-learning)**can be achieved at this stage.**

**Within this process, it is considered to be reliable because the produced transformed data is well maintained and controlled.**

**Publish:**

**Once the above two stages are completed, the data will be transformed to a stage where the BI team can actually consume the data and start with their analysis. The transformed data from the cloud will be pushed to on- premises sources like SQL Server.**

## **Top-level concepts**

## **An Azure subscription might have one or more Azure Data Factory instances (or data factories). Azure Data Factory is composed of four key components. These components work together to provide the platform on which you can compose data-driven workflows with steps to move and transform data.**

### **Pipeline**

**A data factory might have one or more pipelines. A pipeline is a logical grouping of activities that performs a unit of work. Together, the activities in a pipeline perform a task. For example, a pipeline can contain a group of activities that ingests data from an Azure blob, and then runs a Hive query on an HDInsight cluster to partition the data.**

**The benefit of this is that the pipeline allows you to manage the activities as a set instead of managing each one individually. The activities in a pipeline can be chained together to operate sequentially, or they can operate independently in parallel.**

### **Activity**

**Activities represent a processing step in a pipeline. For example, you might use a copy activity to copy data from one data store to another data store. Similarly, you might use a Hive activity, which runs a Hive query on an Azure HDInsight cluster, to transform or analyze your data. Data Factory supports three types of activities: data movement activities, data transformation activities, and control activities.**

### **Datasets**

**Datasets represent data structures within the data stores, which simply point to or reference the data you want to use in your activities as inputs or outputs.**

### **Linked services**

**Linked services are much like connection strings, which define the connection information that's needed for Data Factory to connect to external resources. Think of it this way: a linked service defines the connection to the data source, and a dataset represents the structure of the data. For example, an Azure Storage-linked service specifies a connection string to connect to the Azure Storage account. Additionally, an Azure blob dataset specifies the blob container and the folder that contains the data.**

**Linked services are used for two purposes in Data Factory:**

* **To represent a data store that includes, but isn't limited to, an on-premises SQL Server database, Oracle database, file share, or Azure blob storage account. For a list of supported data stores, see the**[**copy activity**](https://docs.microsoft.com/en-us/azure/data-factory/copy-activity-overview)**article.**
* **To represent a compute resource that can host the execution of an activity. For example, the HDInsightHive activity runs on an HDInsight Hadoop cluster. For a list of transformation activities and supported compute environments, see the**[**transform data**](https://docs.microsoft.com/en-us/azure/data-factory/transform-data)**article.**

### **Triggers**

**Triggers represent the unit of processing that determines when a pipeline execution needs to be kicked off. There are different types of triggers for different types of events.**

### **Pipeline runs**

**A pipeline run is an instance of the pipeline execution. Pipeline runs are typically instantiated by passing the arguments to the parameters that are defined in pipelines. The arguments can be passed manually or within the trigger definition.**

### **Parameters**

**Parameters are key-value pairs of read-only configuration.  Parameters are defined in the pipeline. The arguments for the defined parameters are passed during execution from the run context that was created by a trigger or a pipeline that was executed manually. Activities within the pipeline consume the parameter values.**

**A dataset is a strongly typed parameter and a reusable/referenceable entity. An activity can reference datasets and can consume the properties that are defined in the dataset definition.**

**A linked service is also a strongly typed parameter that contains the connection information to either a data store or a compute environment. It is also a reusable/referenceable entity.**

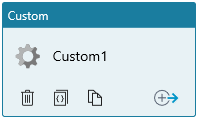
### **Control flow**

**Control flow is an orchestration of pipeline activities that includes chaining activities in a sequence, branching, defining parameters at the pipeline level, and passing arguments while invoking the pipeline on-demand or from a trigger. It also includes custom-state passing and looping containers, that is, For-each iterators.**

**There are two types of activities that you can use in an Azure Data Factory pipeline.**

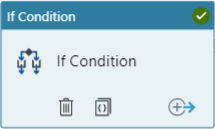
* [**Data movement activities**](https://docs.microsoft.com/en-us/azure/data-factory/copy-activity-overview)**to move data between**[**supported source and sink data stores**](https://docs.microsoft.com/en-us/azure/data-factory/copy-activity-overview#supported-data-stores-and-formats)**.**
* [**Data transformation activities**](https://docs.microsoft.com/en-us/azure/data-factory/transform-data)**to transform data using compute services such as Azure HDInsight, Azure Batch, and Azure Machine Learning.**

**Custom Activiry**

****

**To move data to/from a data store that Data Factory does not support, or to transform/process data in a way that isn't supported by Data Factory, you can create a Custom activity with your own data movement or transformation logic and use the activity in a pipeline.**

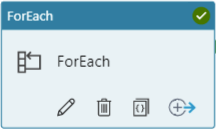
### **If Condition**

****

**If Condition evaluates the boolean expression. Depending on the expression result (true or false), the pipeline will invoke an appropriate set of activities. This activity has the same behavior as a typical “if” statement in many programming languages.**

**For instance, you can provide the following expression as a condition (using pipeline parameters): @bool(pipeline().parameters.paramValue). When you run a pipeline you can define the parameter value. The program will perform the appropriate branch with activities depending on that value.**

### **For Each**

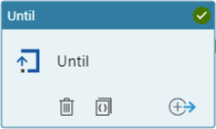
****

**As with the If Condition activity, the For Each activity concept is similar to that in programming languages. It allows you to iterate over a specific set of items and repeat a set of activities.**

**Imagine a situation where you have to copy files to multiple locations within Blob storage. To achieve this, just provide a For Each loop, where the parameter is a collection of desired destination folder paths. For Each loop will iterate through each item and will use it as a destination path for copy activity inside the loop.**

**When you have a lot of items to iterate, you can speed up the execution by setting isSequentialproperty to false. This will change the mode to parallel execution. The current limit is 20 concurrent iterations.**

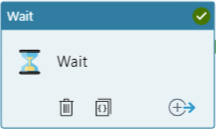
### **Until activity**

****

**Here, the situation is very similar to For Each and If Condition. We have a loop, but in this case, we iterate till the specified condition is true.**

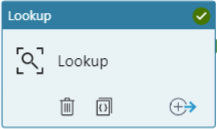
**Example usage scenario might be constant web service calls (Web activity) every 10 seconds. In order to wait 10 seconds between each execution you should use Wait activity.**

### **Wait**

****

**The usage case is very simple. The activity defines in seconds how much time it should wait before continuing pipeline execution of subsequent activities. What you have to do is set the value for waitTimeInSeconds property.**

### **Lookup**

****

**This is another interesting activity. You can use it to read values from an external source as an input for your pipeline. Then you might use the output from Lookup activity in subsequent activities.**

**An example scenario might be where you use Lookup to take values from Azure SQL Database table as an input collection through which the For Each loop should iterate. Basically, you can use Lookup activity to read configuration for your pipeline.**

**Currently, the feature supports several sources:**

* **Files stored on Azure Blob or File System (file must be formatted as JSON)**
* **Azure SQL Database, Azure SQL Data Warehouse, SQL Server**
* **Azure Table storage.**

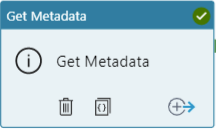
**Another limitation is the number of rows returned by lookup activity which is limited to 5000 records and max. size is 10 MB. Lookup output is formatted as a JSON file, i.e. a set or an array of objects.**

### **Web**

****

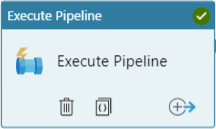
**With web activity, you can call any REST API. For more details about Web activity please refer to the official**[**documentation**](https://docs.microsoft.com/en-us/azure/data-factory/control-flow-web-activity)**.**

### **Get Metadata**

****

**This activity allows for collecting metadata about Azure Data Factory. At the time of writing this article, the Get Metadata activity supports only retrieving metadata from Blob datasets. You can retrieve information on dataset size, structure and last modified time. Please refer to official**[**documentation**](https://docs.microsoft.com/en-us/azure/data-factory/control-flow-get-metadata-activity)**for more details.**

### **Execute Pipeline**

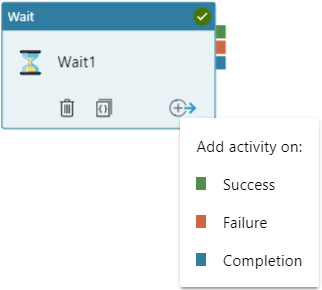
****

**This lets you run another pipeline (child) from an existing one (parent). This is especially useful when your pipeline expanded and you have repeatable workflow steps. This is a good scenario to put such a piece of workflow in a separate pipeline and reuse it whenever possible.**

**Additionally, it is possible to make this “brick” more generic if you define the appropriate parameters. You can pass their values from parent pipeline to child pipeline.**

**One more thing worth mentioning is the waitOnCompletion property. It defines whether the pipeline should wait for the related pipeline (parent) to finish execution before the run (child).**

### **On Success, Failure and Completion**

****

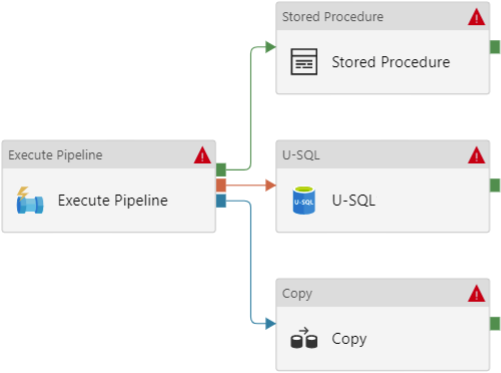
**The above activities are not the only possible ways to control pipeline flow. With Azure Data Factory V2 you can define which activity should be executed next, depending on the execution result of the current one. You can define workflow path by selecting an activity and clicking on the “+” button. Here you have three options:**

* **Success – go to next activity when the current execution ends with success.**
* **Failure – go to next activity when the current execution ends with error (failure).**
* **Completion – go to next activity when the current execution ends, no matter the result. It can be failure or success.**

**In the below example you can see the possible execution paths, i.e. three lines: green, red and blue. But in the end, there are only two possible execution paths.**

* **In case of successful Execute Pipeline activity, the Stored Procedure activity and Copy activity will be executed**
* **In case of failed Execute Pipeline activity, the U-SQL activity and Copy activity will be executed.**

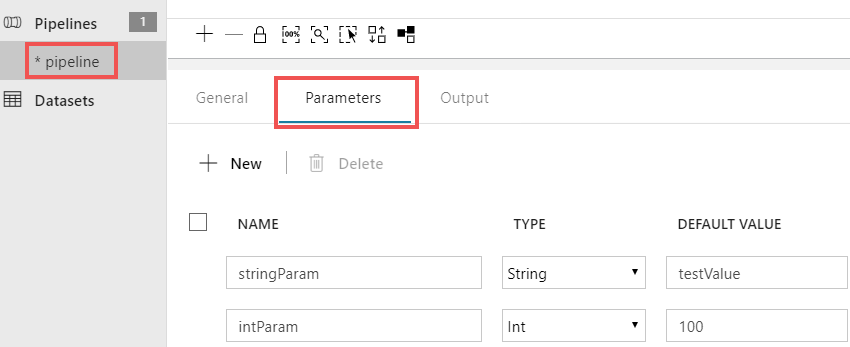
**As you can see, the program will always execute the Copy activity, no matter what the result of Execute Pipeline activity is.**

****

## **@Parameters**

**Great! We’ve gained some knowledge about the new activities and pipeline workflow. Now, let’s discuss parameters.**

**Parameters allow for making the pipeline more flexible. We define parameters on a pipeline level. To do so, just select the pipeline and switch to Parameters tab. Then click the + New button and provide parameter name, specify the type and provide a default value.**

****

**In order to refer to defined pipeline parameter values, use the following expression:**

* **@pipeline().parameters.stringParam**
* **@pipeline().parameters.intParam**

**The syntax is quite simple, i.e. @pipeline().parameters.ParameterName. Nevertheless, a more detailed description you will find**[**here**](https://docs.microsoft.com/en-us/azure/data-factory/control-flow-expression-language-functions)**.**

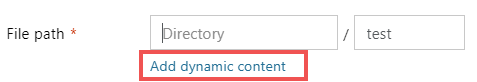
**We know how to define parameters and how to refer to them. We can use them in the following situations:**

* **to set the property value of activities, i.e. as a condition in If Condition activity, or setting Data Movement Units in Copy activity**
* **to set the property value of datasets, i.e. file path, file format, stored procedure or table name.**

**You can assign parameter value to a specific property (of activity or dataset) in two ways.**

### **Assigning parameter value directly from a specific property**

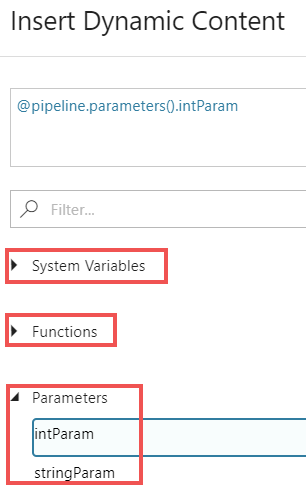
**For example, you have defined Azure Blob dataset. Go to Connection tab and set the cursor on File Path; Add dynamic content should appear.**

****

**Once you click on it, the Insert Dynamic Content panel should appear. At the bottom, you should see the Parameters section with all parameters defined within the pipeline. Double click on a specific parameter in order to set an expression rather than write it by hand.**

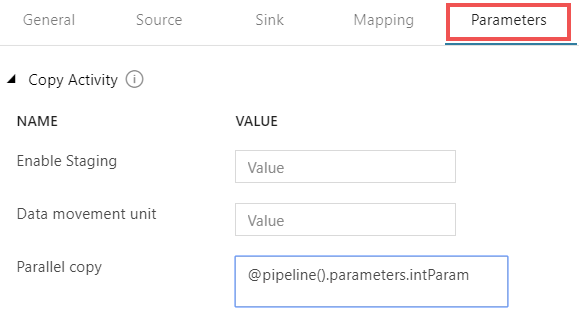
**Here you can also build a more advanced expression using expression language with a set of built-in functions (date, time, strings functions etc.) and system variables.**

**If you wish to dive deeper into this topic, read the official documentation where**[**expression language, functions**](https://docs.microsoft.com/en-us/azure/data-factory/control-flow-expression-language-functions)**and**[**system variables**](https://docs.microsoft.com/en-us/azure/data-factory/control-flow-system-variables)**are explained.**

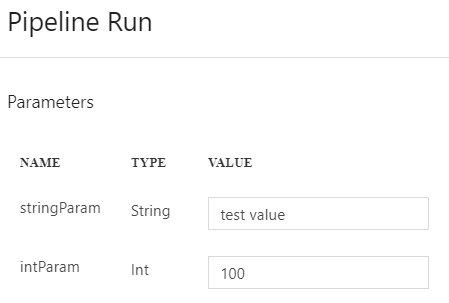
****

### **Use the Parameters tab**

**The second option is to use the Parameters tab on a specific activity or dataset. Here you have a list of properties for which you can set parameters. As you probably noticed, these properties are available in other tabs (within a specific activity or dataset), but here you can find them all in one place. It is more convenient to set properties from here rather than search for them one by one on separate tabs. Notice that you cannot set every dataset or activity property value dynamicallythrough parameters, e.g. linked service for the dataset.**

****

**When all is done, i.e. the parameters are defined and their references are set to appropriate properties, run your pipeline. On the Pipeline Run panel, you will see a list of all defined parameters with default values. Here is the place where you can change them, and test pipeline with the new parameters context.**

****

## **Summary**

**In this post, I went through several new activities introduced to Azure Data Factory V2. These activities significantly improve the possibilities for building a more advanced pipeline workflow logic.  Additionally, it is possible to define a pipeline workflow path based on activity completion result.**

### **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.**

### **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.**

### **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** |

### **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 data.**

### **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.**

**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.**

## **Event Hub**

* **Azure Event Hubs is a Big Data streaming platform and event ingestion service, capable of receiving and processing millions of events per second. Event Hubs can process and store events, data, or telemetry produced by distributed software and devices. Data sent to an event hub can be transformed and stored using any real-time analytics provider or batching/storage adapters**

**Structured Streaming + Event Hubs Integration Guide**

**Structured Streaming integration for Azure Event Hubs to read data from Event Hubs.**

**Table of Contents**

* [**Linking**](https://github.com/Azure/azure-event-hubs-spark/blob/master/docs/structured-streaming-eventhubs-integration.md#linking)
* [**User Configurations**](https://github.com/Azure/azure-event-hubs-spark/blob/master/docs/structured-streaming-eventhubs-integration.md#user-configuration)
  + [**Connection String**](https://github.com/Azure/azure-event-hubs-spark/blob/master/docs/structured-streaming-eventhubs-integration.md#connection-string)
  + [**EventHubsConf**](https://github.com/Azure/azure-event-hubs-spark/blob/master/docs/structured-streaming-eventhubs-integration.md#eventhubsconf)
* [**Reading Data from Event Hubs**](https://github.com/Azure/azure-event-hubs-spark/blob/master/docs/structured-streaming-eventhubs-integration.md#reading-data-from-event-hubs)
  + [**Creating an Event Hubs Source for Streaming Queries**](https://github.com/Azure/azure-event-hubs-spark/blob/master/docs/structured-streaming-eventhubs-integration.md#creating-an-event-hubs-source-for-streaming-queries)
  + [**Creating an Event Hubs Source for Batch Queries**](https://github.com/Azure/azure-event-hubs-spark/blob/master/docs/structured-streaming-eventhubs-integration.md#creating-an-event-hubs-source-for-batch-queries)
* [**Writing Data to EventHubs**](https://github.com/Azure/azure-event-hubs-spark/blob/master/docs/structured-streaming-eventhubs-integration.md#writing-data-to-eventhubs)
  + [**Creating an EventHubs Sink for Streaming Queries**](https://github.com/Azure/azure-event-hubs-spark/blob/master/docs/structured-streaming-eventhubs-integration.md#creating-an-eventhubs-sink-for-streaming-queries)
  + [**Writing the output of Batch Queries to EventHubs**](https://github.com/Azure/azure-event-hubs-spark/blob/master/docs/structured-streaming-eventhubs-integration.md#writing-the-output-of-batch-queries-to-eventhubs)
* [**Managing Throughput**](https://github.com/Azure/azure-event-hubs-spark/blob/master/docs/structured-streaming-eventhubs-integration.md#managing-throughput)
* [**Serialization of Event Data Properties**](https://github.com/Azure/azure-event-hubs-spark/blob/master/docs/structured-streaming-eventhubs-integration.md#serialization-of-event-data-properties)
* [**Deploying**](https://github.com/Azure/azure-event-hubs-spark/blob/master/docs/structured-streaming-eventhubs-integration.md#deploying)

**Linking**

**For Scala/Java applications using SBT/Maven project definitions, link your application with the following artifact:**

**groupId = com.microsoft.azure**

**artifactId = azure-eventhubs-spark\_2.11**

**version = 2.3.17**

**or**

**groupId = com.microsoft.azure**

**artifactId = azure-eventhubs-spark\_2.12**

**version = 2.3.17**

**For Python applications, you need to add this above library and its dependencies when deploying your application. See the**[**Deploying**](https://github.com/Azure/azure-event-hubs-spark/blob/master/docs/structured-streaming-eventhubs-integration.md#deploying)**subsection below.**

**User Configuration**

**Connection String**

**An Event Hubs connection string is required to connect to the Event Hubs service. You can get the connection string for your Event Hubs instance from the**[**Azure Portal**](https://portal.azure.com/)**or by using the ConnectionStringBuilder in our library.**

**Azure Portal**

**When you get the connection string from the Azure Portal, it may or may not have the EntityPath key. Consider:**

**// Without an entity path**

**val without = "Endpoint=ENDPOINT;SharedAccessKeyName=KEY\_NAME;SharedAccessKey=KEY"**

**// With an entity path**

**val with = "Endpoint=sb://SAMPLE;SharedAccessKeyName=KEY\_NAME;SharedAccessKey=KEY;EntityPath=EVENTHUB\_NAME"**

**To connect to your EventHubs, an EntityPath must be present. If your connection string doesn't have one, don't worry! This will take care of it:**

**import org.apache.spark.eventhubs.ConnectionStringBuilder**

**val connectionString = ConnectionStringBuilder(without) // defined in the previous code block**

**.setEventHubName("EVENTHUB\_NAME")**

**.build**

**ConnectionStringBuilder**

**Alternatively, you can use the ConnectionStringBuilder to make your connection string.**

**import org.apache.spark.eventhubs.ConnectionStringBuilder**

**val connectionString = ConnectionStringBuilder()**

**.setNamespaceName("NAMESPACE\_NAME")**

**.setEventHubName("EVENTHUB\_NAME")**

**.setSasKeyName("KEY\_NAME")**

**.setSasKey("KEY")**

**.build**

**EventHubsConf**

**All configuration relating to Event Hubs happens in your EventHubsConf. To create an EventHubsConf, you must pass a connection string:**

**val connectionString = "YOUR.CONNECTION.STRING"**

**val ehConf = EventHubsConf(connectionString)**

**Please read the**[**Connection String**](https://github.com/Azure/azure-event-hubs-spark/blob/master/docs/structured-streaming-eventhubs-integration.md#connection-string)**subsection for more information on obtaining a valid connection string.**

**Additionally, the following configurations are optional:**

| **Option** | **value** | **default** | **query type** | **meaning** |
| --- | --- | --- | --- | --- |
| **consumerGroup** | **String** | **"$Default"** | **streaming and batch** | **A consumer group is a view of an entire event hub. Consumer groups enable multiple consuming applications to each have a separate view of the event stream, and to read the stream independently at their own pace and with their own offsets. More info is available**[**here**](https://docs.microsoft.com/en-us/azure/event-hubs/event-hubs-features#event-consumers) |
| **startingPositions** | **Map[NameAndPartition, EventPosition]** | **end of stream** | **streaming and batch** | **Sets starting positions for specific partitions. If any positions are set in this option, they take priority over any other option. If nothing is configured within this option, then the setting in startingPosition is used. If no position has been set in either option, we will start consuming from the end of the partition.** |
| **startingPosition** | **EventPosition** | **end of stream** | **streaming and batch** | **The starting position for your Structured Streaming job. Please read startingPositions for detail on which order the options are read.** |
| **endingPositions** | **Map[NameAndPartition, EventPosition]** | **end of stream** | **batch query** | **The ending position of a batch query on a per partition basis. This works the same as startingPositions.** |
| **endingPosition** | **EventPosition** | **end of stream** | **batch query** | **The ending position of a batch query. This works the same as startingPosition.** |
| **maxEventsPerTrigger** | **long** | **partitionCount \* 1000** | **streaming query** | **Rate limit on maximum number of events processed per trigger interval. The specified total number of events will be proportionally split across partitions of different volume.** |
| **receiverTimeout** | **java.time.Duration** | **60 seconds** | **streaming and batch** | **The amount of time Event Hub receive calls will be retried before throwing an exception.** |
| **operationTimeout** | **java.time.Duration** | **300 seconds** | **streaming and batch** | **The amount of time Event Hub API calls will be retried before throwing an exception.** |
| **prefetchCount** | **int** | **500** | **streaming and batch** | **Sets the prefetch count for the underlying receiver and controls how many events are received in advance.** |
| **threadPoolSize** | **int** | **16** | **streaming and batch** | **Sets the size of thread pool.** |

**For each option, there exists a corresponding setter in the EventHubsConf. For example:**

**import org.apache.spark.eventhubs.\_**

**val cs = "YOUR.CONNECTION.STRING"**

**val ehConf = EventHubsConf(cs)**

**.setConsumerGroup("sample-cg")**

**.setMaxEventsPerTrigger(10000)**

**.setReceiverTimeout(Duration.ofSeconds(30))**

**EventPosition**

**The EventHubsConf allows users to specify starting (and ending) positions with the EventPosition class. EventPosition defines a position of an event in an Event Hub partition. The position can be an enqueued time, offset, sequence number, the start of the stream, or the end of the stream. It's (hopefully!) pretty straightforward:**

**import org.apache.spark.eventhubs.\_**

**EventPosition.fromOffset("246812") // Specifies offset 246812**

**EventPosition.fromSequenceNumber(100L) // Specifies sequence number 100**

**EventPosition.fromEnqueuedTime(Instant.now) // Specifies any event after the current time**

**EventPosition.fromStartOfStream // Specifies from start of stream**

**EventPosition.fromEndOfStream // Specifies from end of stream**

**If you'd like to start (or end) at a specific position, simply create the correct EventPosition and set it in your EventHubsConf:**

**val cs = "YOUR.CONNECTION.STRING"**

**val ehConf = EventHubsConf(cs)**

**.setStartingPosition(EventPosition.fromEndOfStream)**

**Per Partition Configuration**

**For advanced users, we have provided the option to configure starting and ending positions on a per partition basis. Simply pass a Map[NameAndPartition, EventPosition] to your EventHubsConf. Consider:**

**// name is the EventHub name!**

**val positions = Map(**

**new NameAndPartition(name, 0) -> EventPosition.fromStartOfStream,**

**new NameAndPartition(name, 1) -> EventPosition.fromSequenceNumber(100L)**

**)**

**val cs = "YOUR.CONNECTION.STRING"**

**val ehConf = EventHubsConf(cs)**

**.setStartingPositions(positions)**

**.setStartingPosition(EventPosition.fromEndOfStream)**

**In this case, partition 0 starts from the beginning of the partition, partition 1 starts from sequence number 100L, and all other partitions will start from the end of the partitions. You can start from any position on any partition you'd like!**

**IoT Hub**

**If using IoT Hub, getting your connection string is the only part of the process that is different - all other documentation still applies. Follow these instructions to get your EventHubs-compatible connection string:**

1. **Go to the**[**Azure Portal**](https://ms.portal.azure.com/)**and find your IoT Hub instance**
2. **Click on Endpoints under Messaging. Then click on Events.**
3. **Find your EventHub-compatible name and EventHub-compatible endpoint.**

**import org.apache.spark.eventhubs.ConnectionStringBuilder**

**// Build connection string with the above information**

**val connectionString = ConnectionStringBuilder("YOUR.EVENTHUB.COMPATIBLE.ENDPOINT")**

**.setEventHubName("YOUR.EVENTHUB.COMPATIBLE.NAME")**

**.build**

**Reading Data from Event Hubs**

**Creating an Event Hubs Source for Streaming Queries**

**// Source with default settings**

**val connectionString = "Valid EventHubs connection string."**

**val ehConf = EventHubsConf(connectionString)**

**val df = spark**

**.readStream**

**.format("eventhubs")**

**.options(ehConf.toMap)**

**.load()**

**val eventhubs = df.select($"body" cast "string")**

**// Source with per partition starting positions and rate limiting. In this case, we'll start from**

**// a sequence number for partition 0, enqueued time for partition 3, the end of stream**

**// for partition 5, and the start of stream for all other partitions.**

**val connectionString = "Valid EventHubs connection string."**

**val name = connectionString.getEventHubName**

**val positions = Map(**

**new NameAndPartition(name, 0) -> EventPosition.fromSequenceNumber(1000L, isInclusive = true),**

**new NameAndPartition(name, 3) -> EventPosition.fromEnqueuedTime(Instant.now),**

**new NameAndPartition(name, 5) -> EventPosition.fromEndOfStream**

**)**

**val ehConf = EventHubsConf(connectionString)**

**.setStartingPositions(positions)**

**.setMaxEventsPerTrigger(10000)**

**val df = spark**

**.readStream**

**.format("eventhubs")**

**.options(ehConf.toMap)**

**.load()**

**Creating an Event Hubs Source for Batch Queries**

**// Simple batch query**

**val df = spark**

**.read**

**.format("eventhubs")**

**.options(ehConf.toMap)**

**.load()**

**df.select($"body" cast "string")**

**// start from same place across all partitions. end at the same place accross all partitions.**

**val ehConf = EventHubsConf("VALID.CONNECTION.STRING")**

**.setStartingPosition(EventPosition.fromSequenceNumber(1000L)**

**.setEndingPosition(EventPosition.fromEnqueuedTime(Instant.now)**

**// per partition config**

**val start = Map(**

**new NameAndPartition(name, 0) -> EventPosition.fromSequenceNumber(1000L),**

**new NameAndPartition(name, 1) -> EventPosition.fromOffset("100")**

**)**

**val end = Map(**

**new NameAndPartition(name, 0) -> EventPosition.fromEnqueuedTime(Instant.now),**

**new NameAndPartition(name, 1) -> EventPosition.fromSequenceNumber(1000L)**

**)**

**val ehConf = EventHubsConf("VALID.CONNECTION.STRING")**

**.setStartingPositions(start)**

**.setEndingPositions(end)**

**Each row in the source has the following schema:**

| **Column** | **Type** |
| --- | --- |
| **body** | **binary** |
| **partition** | **string** |
| **offset** | **string** |
| **sequenceNumber** | **long** |
| **enqueuedTime** | **timestamp** |
| **publisher** | **string** |
| **partitionKey** | **string** |
| **properties** | **map[string, json]** |
| **systemProperties** | **map[string, json]** |

**Writing Data to EventHubs**

**Here, we describe the support for writting Streaming Queries and Batch Queries to Azure EventHubs. Take note that, today, Azure EventHubs only supports at least once semantics. Consequently, when writing - either Streaming Queries or Batch Queries - to EventHubs, some records may be duplicated; this can happen, for example, if EventHubs needs to retry an event that was not acknowledged by the EventHubs service, event if the service received and stored the event. Structured Streaming cannot prevent such duplicates from ocurring due to these EventHubs write semantics. However, if writing the query is successful, then you can assume that the query output was written at least once. A possible solution to remove duplicates when reading the written data could be to introduce a primary (unique) key that can be used to perform de-duplication when reading.**

**The Dataframe being written to EventHubs should have the following columns in the schema:**

| **Column** | **Type** |
| --- | --- |
| **body (required)** | **string or binary** |
| **partitionId (\*optional)** | **string** |
| **partitionKey (\*optional)** | **string** |
| **properties (optional)** | **map[string, string]** |

* **Only one (partitionId or partitionKey) can be set at a time. If both are set, your Structured Streaming job will be stopped.**

**The body column is the only required option. If a partitionId and partitionKey are not provided, then events will distributed to partitions using a round-robin model. Alternatively, if a partitionId is provided, the query output will be sent to that specific partition exclusively. Sending to a single partition is not a recommended pattern. Finally, if a partionKey is provided, each event will be sent with the provided partitionKey. For more information on how a partitionKey works, click**[**here**](https://docs.microsoft.com/en-us/azure/event-hubs/event-hubs-programming-guide#partition-key)**.**

**Users can also provided properties via a map[string, string] if they would like to send any additional properties with their events.**

**Creating an EventHubs Sink for Streaming Queries**

**// Write body data from a DataFrame to EventHubs. Events are distributed across partitions using round-robin model.**

**val ds = df**

**.select("body")**

**.writeStream**

**.format("eventhubs")**

**.options(ehWriteConf.toMap) // EventHubsConf containing the destination EventHub connection string.**

**.start()**

**// Write body data from a DataFrame to EventHubs with a partitionKey**

**val ds = df**

**.selectExpr("partitionKey", "body")**

**.writeStream**

**.format("eventhubs")**

**.options(ehWriteConf.toMap) // EventHubsConf containing the destination EventHub connection string.**

**.start()**

**Writing the output of Batch Queries to EventHubs**

**// Write body data from a DataFrame to EventHubs. Events are distributed across partitions using round-robin model.**

**df.select("body")**

**.write**

**.format("eventhubs")**

**.options(ehWriteConf.toMap) // EventHubsConf containing the destination EventHub connection string.**

**.save()**

**// Write body data**

**df.selectExpr("partitionKey", "body")**

**.write**

**.format("eventhubs")**

**.options(ehWriteConf.toMap) // EventHubsConf containing the destination EventHub connection string.**

**.save()**

**ForeachWriter**

**An implementation of ForeachWriter is offered by the EventHubsForeachWriter. For simple round-robin sends, this is the fastest way to write your data from Spark to Event Hubs. For any other send pattern, you must use the EventHubsSink. A sample is shown below:**

**val ehConf = EventHubsConf("YOUR\_CONNECTION\_STRING")**

**val writer = EventHubsForeachWriter(ehConf)**

**val query =**

**streamingSelectDF**

**.writeStream**

**.foreach(writer)**

**.outputMode("update")**

**.trigger(ProcessingTime("25 seconds"))**

**.start()**

**Managing Throughput**

**When you create an Event Hubs namespace, you are prompted to choose how many throughput units you want for your namespace. A single throughput unit (or TU) entitles you to:**

* **Up to 1 MB per second of ingress events (events sent into an event hub), but no more than 1000 ingress events or API calls per second.**
* **Up to 2 MB per second of egress events (events consumed from an event hub).**

**With that said, your TUs set an upper bound for the throughput in your streaming application, and this upper bound needs to be set in Spark as well. In Structured Streaming, this is done with the maxEventsPerTrigger option.**

**Let's say you have 1 TU for a single 4-partition Event Hub instance. This means that Spark is able to consume 2 MB per second from your Event Hub without being throttled. If maxEventsPerTrigger is set such that Spark consumes less than 2 MB, then consumption will happen within a second. You're free to leave it as such or you can increase your maxEventsPerTrigger up to 2 MB per second. If maxEventsPerTrigger is set such that Spark consumes greater than 2 MB, your micro-batch will always take more than one second to be created because consuming from Event Hubs will always take at least one second. You're free to leave it as is or you can increase your TUs to increase throughput.**

**Deploying**

**As with any Spark applications, spark-submit is used to launch your application. azure-eventhubs-spark\_2.11 and its dependencies can be directly added to spark-submit using --packages, such as,**

**./bin/spark-submit --packages com.microsoft.azure:azure-eventhubs-spark\_2.11:2.3.17 ...**

**For experimenting on spark-shell, you can also use --packages to add azure-eventhubs-spark\_2.11 and its dependencies directly,**

**DTA Streaming code :**

**// Databricks notebook source**

**import java.io.ByteArrayInputStream**

**import org.apache.spark.eventhubs.{ConnectionStringBuilder, EventHubsConf, EventPosition}**

**import org.apache.spark.{SparkConf, SparkContext, sql}**

**import org.apache.spark.serializer.KryoSerializer**

**import org.apache.spark.sql.functions.\_**

**import org.apache.spark.sql.streaming.{OutputMode, ProcessingTime, StreamingQuery}**

**import java.util.zip.GZIPInputStream**

**import org.apache.spark.sql.streaming.Trigger**

**import org.apache.spark.sql.types.\_**

**import org.apache.spark.sql.{DataFrame, Row}**

**// COMMAND ----------**

**val connectionString = ConnectionStringBuilder("Endpoint=sb://pqmmjeeventhub001-ns.servicebus.windows.net/;SharedAccessKeyName=oeminterface;SharedAccessKey=jvvxk0J50xgZQTixcmpvwDkxgxMqxf5NFubHmqO+NXk=;EntityPath=fuso.export").setEventHubName("fuso.export").build**

**val ehConf = EventHubsConf(connectionString).setConsumerGroup("oem7").setStartingPosition(EventPosition.fromEndOfStream)**

**val eventHubDF = spark.readStream.format("eventhubs").options(ehConf.toMap).load()**

**val canSignals = eventHubDF.select($"enqueuedTime", get\_json\_object(($"body").cast("string"),**

**"$.VIN").alias("VIN"),**

**get\_json\_object(($"body").cast("string"), "$.TT").alias("TT").alias("TT").cast(LongType),**

**get\_json\_object(($"body").cast("string"), "$.MSG\_TYPE").alias("MSG\_TYPE"),**

**get\_json\_object(($"body").cast("string"), "$.ABS").alias("ABS"),**

**get\_json\_object(($"body").cast("string"), "$.ADA").alias("ADA"),**

**get\_json\_object(($"body").cast("string"), "$.ADW").alias("ADW"),**

**get\_json\_object(($"body").cast("string"), "$.YEC").alias("YEC"))**

**val canColumns = "ABS,ADA,ADW,ALT,APP,ASP,ATL,IMP,INS,KOM,LDA,LDD,MLO,MLZ,NME,NRX,NTX,PTM,PTO,RAS,RBC,REC,RHS,RLS,RLV,SDP,SDS,STV,TPI,TPS,TQH,TTS,VVM,YBC,YEC"**

**def canUDF = udf((flagCol: String, r: Row) => {**

**var signal: String = ""**

**val flagColList: List[String] = flagCol.reverse.split(s""",""").map(x => x.trim).mkString(",").reverse.split(s",").toList**

**flagColList.foreach { x =>**

**if (r.getAs(x) != null) {**

**signal += r.getAs(x).toString.replaceAll(""""E":""", """"SN":""" + "\"" + x.toString + "\"" + "," + """"E":""")**

**}**

**}**

**signal.replaceAll("\\]\\[", ",")**

**})**

**val canSignalsDF = canSignals.withColumn("Event", canUDF(lit(canColumns), struct(canSignals.columns map col: \_\*))).withColumn("type", lit("CAN")).select("vin", "tt", "type", "msg\_type", "enqueuedTime", "Event")**

**val canFinalDF = canSignalsDF.withColumn("year", year(to\_date($"enqueuedTime", "MM/dd/yyyy"))).withColumn("month", month(to\_date($"enqueuedTime", "MM/dd/yyyy"))).withColumn("day", dayofmonth(to\_date($"enqueuedTime", "MM/dd/yyyy")))**

**def writeStreamer(input: DataFrame, checkPointFolder: String, output: String): StreamingQuery = {**

**input**

**.writeStream**

**.format("com.databricks.spark.avro")**

**.partitionBy("year", "month", "day")**

**.option("checkpointLocation", checkPointFolder)**

**.option("path", output)**

**.outputMode(OutputMode.Append)**

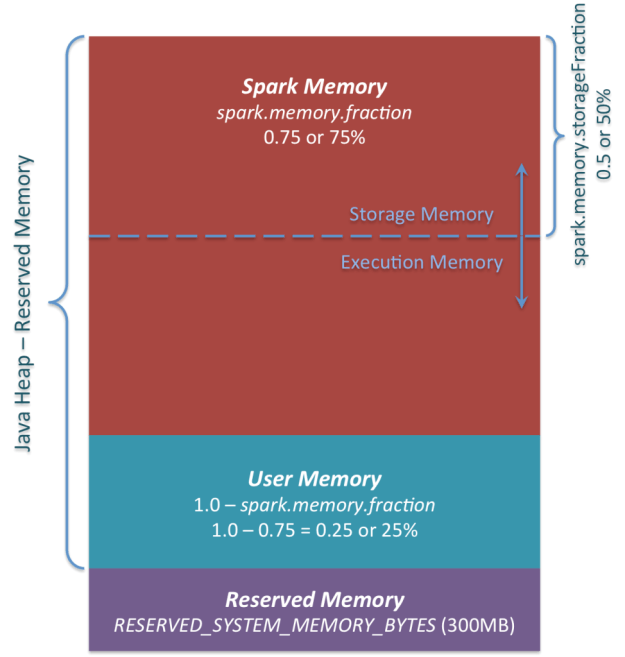
**.start()**

**}**

**writeStreamer(canFinalDF, "/mnt/qm01pasparkhdisa-defcon/qmctdl/CAN\_CheckPoint\_ALL\_SIGNALS", "/mnt/qm01pasparkhdisa-defcon/qmctdl/CAN\_DATA\_ALL\_SIGNALS")**

**// Done streaming code implementation**

**How you mange spark memory and memory /performance issues .**

**[](https://i1.wp.com/0x0fff.com/wp-content/uploads/2016/01/Spark-Memory-Management-1.6.0.png)**

**Apache Spark Unified Memory Manager introduced in v1.6.0+**

**You can see 3 main memory regions on the diagram:**

1. **Reserved Memory. This is the memory reserved by the system, and its size is hardcoded. As of Spark 1.6.0, its value is 300MB, which means that this 300MB of RAM does not participate in Spark memory region size calculations, and its size cannot be changed in any way without Spark recompilation or setting spark.testing.reservedMemory, which is not recommended as it is a testing parameter not intended to be used in production. Be aware, this memory is only called “reserved”, in fact it is not used by Spark in any way, but it sets the limit on what you can allocate for Spark usage. Even if you want to give all the Java Heap for Spark to cache your data, you won’t be able to do so as this “reserved” part would remain spare (not really spare, it would store lots of Spark internal objects). For your information, if you don’t give Spark executor at least 1.5 \* Reserved Memory = 450MB heap, it will fail with “please use larger heap size” error message.**
2. **User Memory. This is the memory pool that remains after the allocation of Spark Memory, and it is completely up to you to use it in a way you like. You can store your own data structures there that would be used in RDD transformations. For example, you can rewrite Spark aggregation by using mapPartitions transformation maintaining hash table for this aggregation to run, which would consume so called User Memory. In Spark 1.6.0 the size of this memory pool can be calculated as (“Java Heap” – “Reserved Memory”) \* (1.0 – spark.memory.fraction), which is by default equal to (“Java Heap” – 300MB) \* 0.25. For example, with 4GB heap you would have 949MB of User Memory. And again, this is the User Memory and its completely up to you what would be stored in this RAM and how, Spark makes completely no accounting on what you do there and whether you respect this boundary or not. Not respecting this boundary in your code might cause OOM error.**
3. **Spark Memory. Finally, this is the memory pool managed by Apache Spark. Its size can be calculated as (“Java Heap” – “Reserved Memory”) \* spark.memory.fraction, and with Spark 1.6.0 defaults it gives us (“Java Heap” – 300MB) \* 0.75. For example, with 4GB heap this pool would be 2847MB in size. This whole pool is split into 2 regions – Storage Memory and Execution Memory, and the boundary between them is set by spark.memory.storageFractionparameter, which defaults to 0.5. The advantage of this new memory management scheme is that this boundary is not static, and in case of memory pressure the boundary would be moved, i.e. one region would grow by borrowing space from another one. I would discuss the “moving” this boundary a bit later, now let’s focus on how this memory is being used:**
   1. **Storage Memory. This pool is used for both storing Apache Spark cached data and for temporary space serialized data “unroll”. Also all the “broadcast” variables are stored there as cached blocks. In case you’re curious, here’s the code of**[**unroll**](https://github.com/apache/spark/blob/branch-1.6/core/src/main/scala/org/apache/spark/storage/MemoryStore.scala#L249)**. As you may see, it does not require that enough memory for unrolled block to be available – in case there is not enough memory to fit the whole unrolled partition it would directly put it to the drive if desired persistence level allows this. As of “broadcast”, all the broadcast variables are stored in cache with MEMORY\_AND\_DISKpersistence level.**
   2. **Execution Memory. This pool is used for storing the objects required during the execution of Spark tasks. For example, it is used to store**[**shuffle intermediate buffer on the Map side**](https://0x0fff.com/spark-architecture-shuffle/)**in memory, also it is used to store hash table for hash aggregation step. This pool also supports spilling on disk if not enough memory is available, but the blocks from this pool cannot be forcefully evicted by other threads (tasks).**

**Ok, so now let’s focus on the moving boundary between Storage Memory and Execution Memory. Due to nature of Execution Memory, you cannot forcefully evict blocks from this pool, because this is the data used in intermediate computations and the process requiring this memory would simply fail if the block it refers to won’t be found. But it is not so for the Storage Memory – it is just a cache of blocks stored in RAM, and if we evict the block from there we can just update the block metadata reflecting the fact this block was evicted to HDD (or simply removed), and trying to access this block Spark would read it from HDD (or recalculate in case your persistence level does not allow to spill on HDD).**

**So, we can forcefully evict the block from Storage Memory, but cannot do so from Execution Memory. When Execution Memory pool can borrow some space from Storage Memory? It happens when either:**

* **There is free space available in Storage Memory pool, i.e. cached blocks don’t use all the memory available there. Then it just reduces the Storage Memory pool size, increasing the Execution Memory pool.**
* **Storage Memory pool size exceeds the initial Storage Memory region size and it has all this space utilized. This situation causes forceful eviction of the blocks from Storage Memory pool, unless it reaches its initial size.**

**In turn, Storage Memory pool can borrow some space from Execution Memory pool only if there is some free space in Execution Memory pool available.**

**a few suggestions:**

* **If your nodes are configured to have 6g maximum for Spark (and are leaving a little for other processes), then use 6g rather than 4g, spark.executor.memory=6g. Make sure you're using as much memory as possible by checking the UI (it will say how much mem you're using)**
* **Try using more partitions, you should have 2 - 4 per CPU. IME increasing the number of partitions is often the easiest way to make a program more stable (and often faster). For huge amounts of data you may need way more than 4 per CPU, I've had to use 8000 partitions in some cases!**
* **Decrease the fraction of memory reserved for caching, using spark.storage.memoryFraction. If you don't use cache() or persist in your code, this might as well be 0. It's default is 0.6, which means you only get 0.4 \* 4g memory for your heap. IME reducing the mem frac often makes OOMs go away. UPDATE: From spark 1.6 apparently we will no longer need to play with these values, spark will determine them automatically.**
* **Similar to above but shuffle memory fraction. If your job doesn't need much shuffle memory then set it to a lower value (this might cause your shuffles to spill to disk which can have catastrophic impact on speed). Sometimes when it's a shuffle operation that's OOMing you need to do the opposite i.e. set it to something large, like 0.8, or make sure you allow your shuffles to spill to disk (it's the default since 1.0.0).**
* **Watch out for memory leaks, these are often caused by accidentally closing over objects you don't need in your lambdas. The way to diagnose is to look out for the "task serialized as XXX bytes" in the logs, if XXX is larger than a few k or more than an MB, you may have a memory leak. See**[**https://stackoverflow.com/a/25270600/1586965**](https://stackoverflow.com/a/25270600/1586965)
* **Related to above; use broadcast variables if you really do need large objects.**
* **If you are caching large RDDs and can sacrifice some access time consider serialising the RDD**[**http://spark.apache.org/docs/latest/tuning.html#serialized-rdd-storage**](http://spark.apache.org/docs/latest/tuning.html#serialized-rdd-storage)**. Or even caching them on disk (which sometimes isn't that bad if using SSDs)**

**Difference between Spark Caching and Broadcasting ?**

**Broadcast variables are useful when large datasets needs to be cached in executors.**

**Broadcast variables in Apache Spark is a mechanism for sharing variables across executors that are meant to be read-only. Without broadcast variables these variables would be shipped to each executor for every transformation and action, and this can cause network overhead. However, with broadcast variables, they are shipped once to all executors and are cached for future reference.**

## **Broadcast Variables Use case**

**Imagine that while doing a transformation we need to lookup a large table of zip codes/pin codes. Here, it is neither feasible to send the large lookup table every time to the executors, nor can we query the database every time. The solution should be to convert this lookup table to a broadcast variables and Spark will cache it in every executor for future reference.**

**You cache or broadcast an object when you want to use it multiple times.**

**You can only cache an RDD or RDD-derivative, whereas you can broadcast any kind of object, including RDDs.**

**We use cache() when we're dealing with an RDD/DataFrame/DataSet and we want to use the dataset multiple times without recomputing it afresh each time.**

**We broadcast an object when**

1. **we're dealing with an RDD/DataFrame/DataSet which is relatively small, and broadcasting it offers performance benefits over caching (e.g. if we're using the dataset in a join)**
2. **we're dealing with a plain old Scala/Java object and it will be used across multiple stages of a job.**
3. **You can cache an RDD in spark when that RDD is smaller among all other RDDs and that RDD is going to use throughout the process.**
4. **When you cache the RDD it doesn’t mean that it is distributed , if you want to distribute then you have to use Broadcast variables and Accumulators .**

**Differnece Between Spark streaming and Structured Streaming ?**

**Spark provides us with two ways of working with streaming data:**

1. **Spark Streaming**
2. **Structured Streaming (introduced with Spark 2.x)**

**Let's discuss what these are exactly, what the differences are, and which one is better.**

## **Spark Streaming**

**Spark Streaming is a separate library in Spark to process continuously flowing streaming data. It provides us with the DStream API, which is powered by Spark RDDs. DStreams provide us data divided into chunks as RDDs received from the source of streaming to be processed and, after processing, sends it to the destination. Cool, right?!**

## **Structured Streaming**

**From the Spark 2.x release onwards, Structured Streaming came into the picture. Built on the Spark SQL library, Structured Streaming is another way to handle streaming with Spark. This model of streaming is based on Dataframe and Dataset APIs. Hence, with this library, we can easily apply any SQL query (using the DataFrame API) or Scala operations (using DataSet API) on streaming data.**

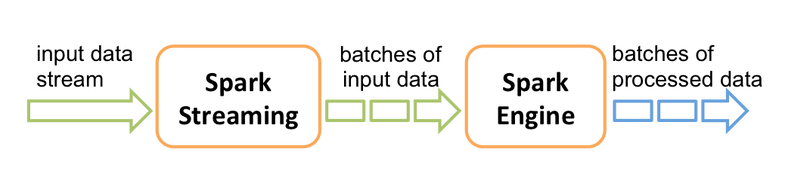
**Okay, so that was the summarized theory for both ways of streaming in Spark. Now we need to compare the two.**

## **Distinctions**

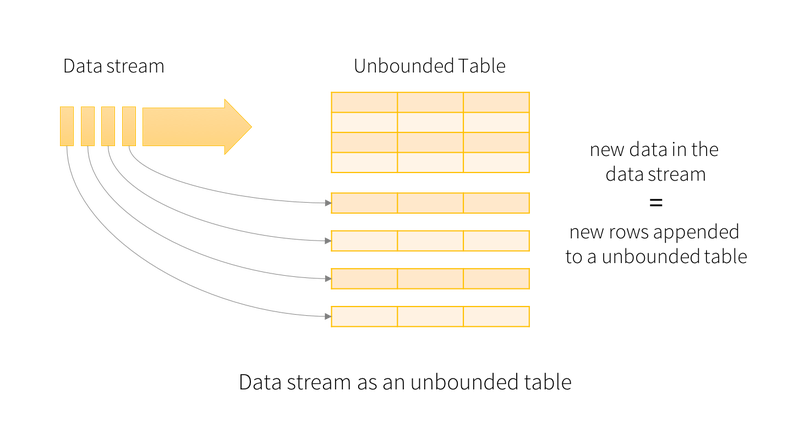
### **1. Real Streaming**

**What does real streaming imply? Data which is unbounded and is being processed upon being received from the source. This definition is satisfiable (more or less).**

**If we talk about Spark Streaming, this is not the case. Spark Streaming works on something we call a micro batch. The stream pipeline is registered with some operations and Spark polls the source after every batch duration (defined in the application) and then a batch is created of the received data, i.e. each incoming record belongs to a batch of DStream. Each batch represents an RDD.**

****

**Structured Streaming works on the same architecture of polling the data after some duration, based on your trigger interval, but it has some distinction from the Spark Streaming which makes it more inclined towards real streaming. In Structured Streaming, there is no batch concept. The received data in a trigger is appended to the continuously flowing data stream. Each row of the data stream is processed and the result is updated into the unbounded result table. How you want your result (updated, new result only, or all the results) depends on the mode of your operations (Complete, Update, Append).**

**Winner of this round: Structured Streaming.**

### **2. RDD vs. DataFrames/DataSet**

**Another distinction can be the use case of different APIs in both streaming models. In summary, we read that Spark Streaming works on the DStream API, which is internally using RDDs and Structured Streaming uses DataFrame and Dataset APIs to perform streaming operations. So, it is a straight comparison between using RDDs or DataFrames. There are several blogs available which compare DataFrames and RDDs in terms of `performance` and `ease of use.` This is a good read for**[**RDD v/s Dataframes**](https://blog.knoldus.com/spark-rdd-vs-dataframes/)**. All those comparisons lead to one result: that DataFrames are more optimized in terms of processing and provide more options for aggregations and other operations with a variety of functions available (many more functions are now supported natively in Spark 2.4).**

**So Structured Streaming wins here with flying colors.**

### **3. Processing With the Vent Time, Handling Late Data**

**One big issue in the streaming world is how to process data according to the event-time. Event-time is the time when the event actually happened. It is not necessary for the source of the streaming engine to prove data in real-time. There may be latencies in data generation and handing over the data to the processing engine. There is no such option in Spark Streaming to work on the data using the event-time. It only works with the timestamp when the data is received by the Spark. Based on the ingestion timestamp, Spark Streaming puts the data in a batch even if the event is generated early and belonged to the earlier batch, which may result in less accurate information as it is equal to the data loss. On the other hand, Structured Streaming provides the functionality to process data on the basis of event-time when the timestamp of the event is included in the data received. This is a major feature introduced in Structured Streaming which provides a different way of processing the data according to the time of data generation in the real world. With this, we can handle data coming in late and get more accurate results.**

**With event-time handling of late data, Structured Streaming outweighs Spark Streaming.**

### **4. End-to-End Guarantees**

**Every application requires fault tolerance and end-to-end guarantees of data delivery. Whenever the application fails, it must be able to restart from the same point where it failed in order to avoid data loss and duplication. To provide fault tolerance, Spark Streaming and Structured Streaming both use the checkpointing to save the progress of a job. But this approach still has many holes which may cause data loss.**

**Other than checkpointing, Structured Streaming has applied two conditions to recover from any error:**

1. **The source must be replayable.**
2. **The sinks must support idempotent operations to support reprocessing in case of failures.**

**Here's a link to the docs to learn**[**more**](https://spark.apache.org/docs/latest/structured-streaming-programming-guide.html#fault-tolerance-semantics)**.**

**With restricted sinks, Spark Structured Streaming always provides end-to-end, exactly once semantics. Way to go Structured Streaming!**

### **5. Restricted or Flexible**

**Sink: The destination of a streaming operation. It can be external storage, a simple output to console, or any action**

**With Spark Streaming, there is no restriction to use any type of sink. Here we have the method [foreachRDD](https://spark.apache.org/docs/2.2.0/streaming-programming-guide.html" \l "design-patterns-for-using-foreachrdd" \t "_blank) to perform some action on the stream. This method returns us the RDDs created by each batch one-by-one and we can perform any actions over them, like saving to storage or performing some computations. We can cache an RDD and perform multiple actions on it as well (even sending the data to multiple databases).**

**But in Structures Streaming, until v2.3, we had a limited number of**[**output sinks**](https://spark.apache.org/docs/2.3.0/structured-streaming-programming-guide.html#output-sinks)**and, with one sink, only one operation could be performed and we could not save the output to multiple external storages. To use a custom sink, the user needed to implement [ForeachWriter](https://spark.apache.org/docs/2.2.0/api/scala/index.html" \l "org.apache.spark.sql.ForeachWriter" \t "_blank). But here comes Spark 2.4, and with it we get a new sink called [foreachBatch](https://spark.apache.org/docs/latest/structured-streaming-programming-guide.html" \l "foreachbatch" \t "_blank). This sink gives us the resultant output table as a DataFrame and hence we can use this DataFrame to perform our custom operations.**

**With this new sink, the `restricted` Structured Streaming is now more `flexible` and gives it an edge over the Spark Streaming and other over flexible sinks.**

## **Conclusion**

**We saw a fair comparison between Spark Streaming and Spark Structured Streaming. We can clearly say that Structured Streaming is more inclined to real-time streaming but Spark Streaming focuses more on batch processing. The APIs are better and optimized in Structured Streaming where Spark Streaming is still based on the old RDDs.**

**So to conclude this post, we can simply say that Structured Streaming is a better streaming platform in comparison to Spark Streaming.**

**What is explode()**

**explode() takes in an array (or a map) as an input and outputs the elements of the array (map) as separate rows.**

**val df = sc.parallelize(Seq((1, Seq(2,3,4), Seq(5,6,7)), (2, Seq(3,4,5), Seq(6,7,8)), (3, Seq(4,5,6), Seq(7,8,9)))).toDF(“a”, “b”, “c”)  
val df1 = df.select(df(“a”),explode(df(“b”)).alias(“b\_columns”),df(“c”))  
val df2 = df1.select(df1(“a”),df1(“b\_columns”),explode(df1(“c”).alias(“c\_columns”))).show()  
}**

**}**

**df.show() —> Initial Collections**

**|  a|        b|        c|  
|  1|[2, 3, 4]|[5, 6, 7]|  
|  2|[3, 4, 5]|[6, 7, 8]|  
|  3|[4, 5, 6]|[7, 8, 9]|  
+—+———+———+**

**df1.show() —> First Exploded Collections**

**+—+———+———+  
|  a|b\_columns|        c|  
+—+———+———+  
|  1|        2|[5, 6, 7]|  
|  1|        3|[5, 6, 7]|  
|  1|        4|[5, 6, 7]|  
|  2|        3|[6, 7, 8]|  
|  2|        4|[6, 7, 8]|  
|  2|        5|[6, 7, 8]|  
|  3|        4|[7, 8, 9]|  
|  3|        5|[7, 8, 9]|  
|  3|        6|[7, 8, 9]|**

**Difference between collect and toLocalIterator**

**The key issue here is that when you use a location of file:// every machine assume you are talking about it’s local filesystem. This can lead to madness, what you actually want is all of the data to write to a single file on the Driver’s Local filesystem.**

**A first instinct my be to use collect on the RDD before attempting to write it to a file but this has a distinct limitation. When you use collect every partition is moved from the remote cluster to the driver machine at the same time. This means if you use collect you can never write a file larger than driver heap.**

**toLocalIterator lets us get around this by only pulling down a single Spark partition’s worth of data to the DRiver at a time. This means that you can write as large a file as HDD space you have as long as no one Spark Partition is bigger than the driver heap.**

**The most popular Spark's method used to bring data to the driver is collect(). It executes given job in all partitions (executors side) and collects all results (driver side) with Array.concat(results: \_\*) method. The toLocalIterator does the contrary. Instead of launching the job simultaneously on all partitions it executes the job on 1 partition at once. So, the driver must have enough memory to store the biggest partition.**

**The implementation details look like:**

[**?**](https://www.waitingforcode.com/apache-spark/collecting-part-data-driver-rdd-tolocaiIterator/read)

|  |  |
| --- | --- |
| **1**  **2**  **3**  **4**  **5**  **6** | **def toLocalIterator: Iterator[T] = withScope {**  **def collectPartition(p: Int): Array[T] = {**  **sc.runJob(this, (iter: Iterator[T]) => iter.toArray, Seq(p)).head**  **}**  **(0 until partitions.length).iterator.flatMap(i => collectPartition(i))**  **}** |

**It's important to note, however, that the toLocalIterator doesn't prevent against OOM problems on the driver side. As already mentioned, the driver must be ready to handle the biggest partition. In the case of a lot of created objects, small number of partitions and bad partitioning (e.g. one partition storing 90% of data), the OOM problems are still real.**

# **Windowing Functions In Hive**

**Windowing allows you to create a window on a set of data further allowing aggregation surrounding that data. Windowing in Hive is introduced from Hive 0.11. In this blog, we will be giving a demo on the windowing functions available in Hive.**

**Windowing in Hive includes the following functions**

* **Lead**
  + **The number of rows to lead can optionally be specified. If the number of rows to lead is not specified, the lead is one row.**
  + **Returns null when the lead for the current row extends beyond the end of the window.**
* **Lag**

**The number of rows to lag can optionally be specified. If the number of rows to lag is not specified, the lag is one row.**

**Returns null when the lag for the current row extends before the beginning of the window.**

* **FIRST\_VALUE**
* **LAST\_VALUE**

**The OVER clause**

* **OVER with standard aggregates:**
  + **COUNT**
  + **SUM**
  + **MIN**
  + **MAX**
  + **AVG**

**OVER with a PARTITION BY statement with one or more partitioning columns.**

* **OVER with PARTITION BY and ORDER BY with one or more partitioning and/or ordering columns.**

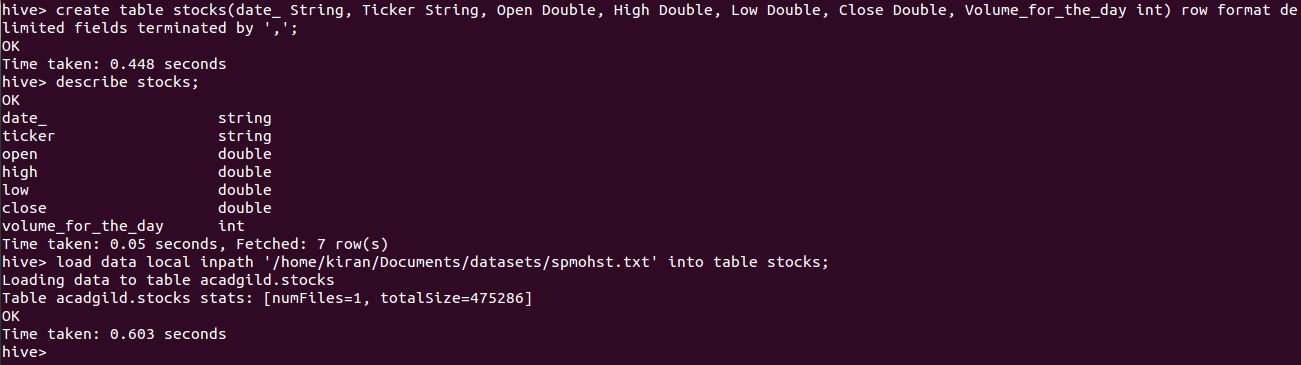
**Analytics functions**

* **RANK**
* **ROW\_NUMBER**
* **DENSE\_RANK**
* **CUME\_DIST**
* **PERCENT\_RANK**
* **NTILE**

**To give you a brief idea of these windowing functions in Hive, we will be using stock market data. You can download the sample stocks data**[**from here**](https://drive.google.com/open?id=0ByJLBTmJojjzbVhvSnQwNlhXUWs)**and load into your stocks table.**

**Now we will create a table to load this stock market data as shown below.**

**create table stocks (date\_ String, Ticker String, Open Double, High Double, Low Double, Close Double, Volume\_for\_the\_day int) row format delimited fields terminated by ',';**

****

**Let us dive deeper into the window functions in Hive.**

## **Lag**

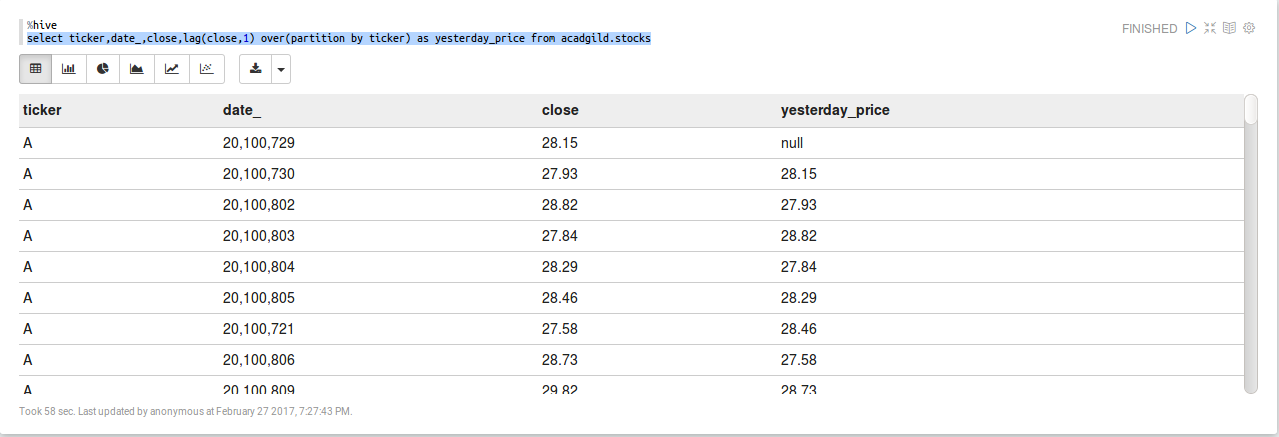
**This function returns the values of the previous row. You can specify an integer offset which designates the row position else it will take the default integer offset as 1.**

**Here is the sample function for lag**

**select ticker,date\_,close,lag(close,1) over(partition by ticker) as yesterday\_price from acadgild.stocks**

**Here using lag we can display the yesterday’s closing price of the ticker. Lag is to be used with over function, inside the over function you can use partition or order by classes.**

**In the below screenshot, you can see the closing price of the stock for the day and the yesterday’s price.**

****

## **Lead**

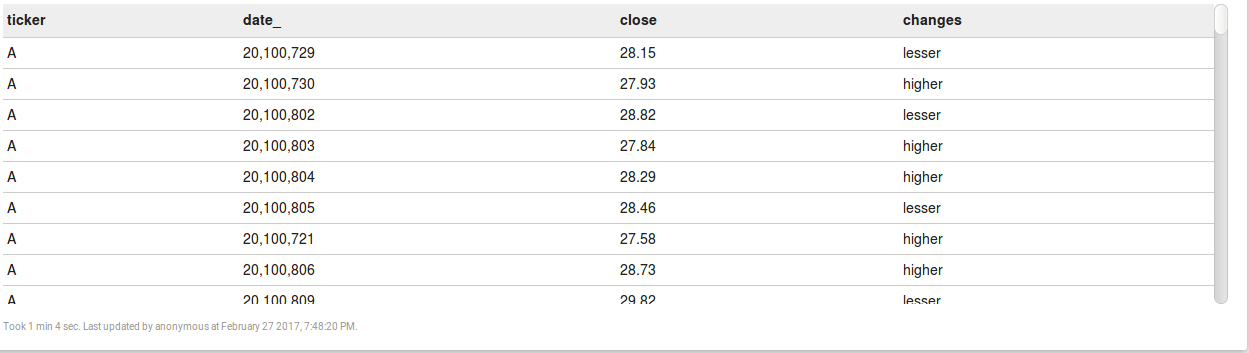
**This function returns the values from the following rows. You can specify an integer offset which designates the row position else it will take the default integer offset as 1.**

**Here is the sample function for lead**

**Now using the lead function, we will find that whether the following day’s closing price is higher or lesser than today’s and that can be done as follows.**

**select ticker,date\_,close,case(lead(close,1) over(partition by ticker)-close)>0 when true then "higher" when false then "lesser" end as Changes from acadgild.stocks**

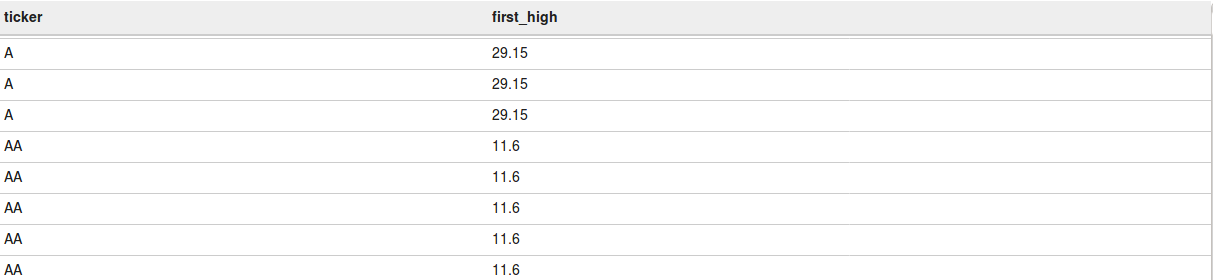
**In the below screenshot, you can see the result.**

****

#### **FIRST\_VALUE**

**It returns the value of the first row from that window. With the below query, you can see the first row high price of the ticker for all the days.**

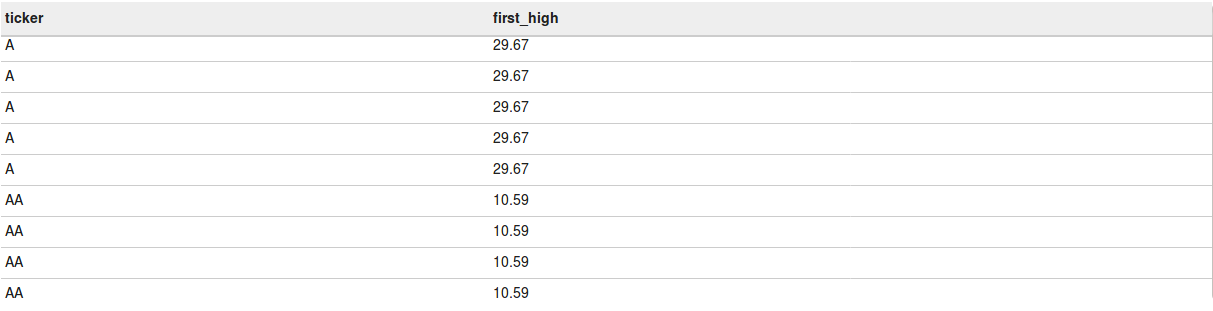
**select ticker,first\_value(high) over(partition by ticker) as first\_high from acadgild.stocks**

****

#### **LAST\_VALUE**

**It is the reverse of FIRST\_VALUE. It returns the value of the last row from that window. With the below query, you can see the last row high price value of the ticker for all the days.**

**select ticker,last\_value(high) over(partition by ticker) as first\_high from acadgild.stocks**

****

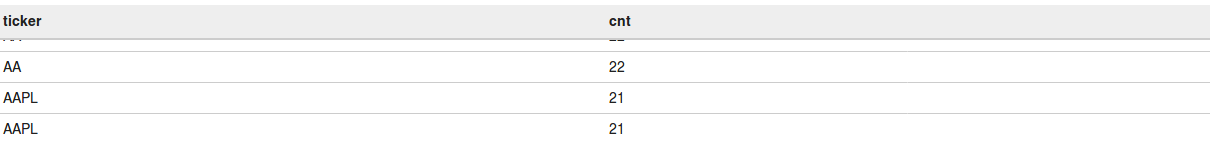
**Let us now see the usage of the aggregate function using Over.**

#### **Count**

**It returns the count of all the values for the expression written in the over clause. From the below query, we can find the number of rows present for each ticker.**

**select ticker,count(ticker) over(partition by ticker) as cnt from acadgild.stocks**

**For each partition, the count of ticker will be calculated, you can see the same in the below screen shot.**

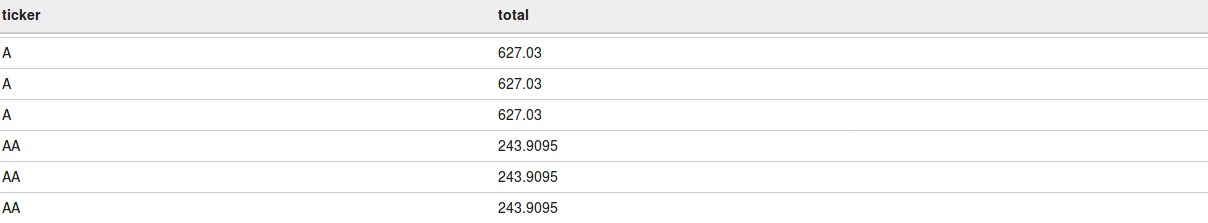
****

#### **Sum**

**It returns the sum of all the values for the expression written in the over clause. From the below query, we can find the sum of all the closing stock prices for that particular ticker.**

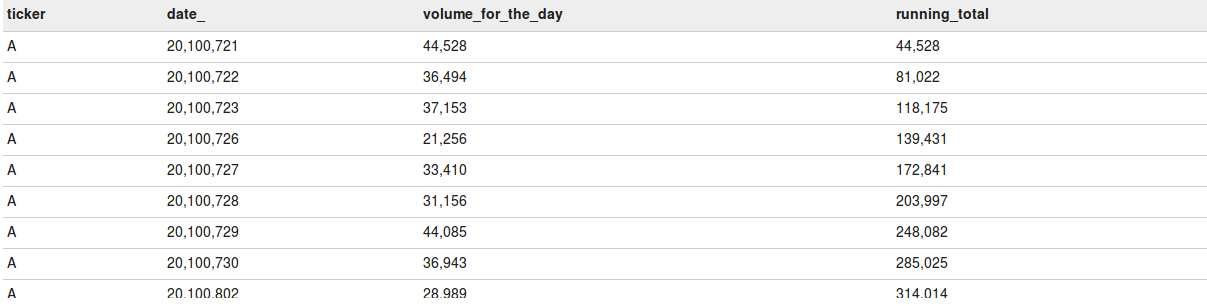
**select ticker,sum(close) over(partition by ticker) as total from acadgild.stocks**

**For each ticker, the sum of all the closing prices will be calculated, you can see the same in the below screen shot.**

****

**For suppose let us take if you want to get running total of the volume\_for\_the\_day for all the days for every ticker then you can do this with the below query.**

**select ticker,date\_,volume\_for\_the\_day,sum(volume\_for\_the\_day) over(partition by ticker order by date\_) as running\_total from acadgild.stocks**

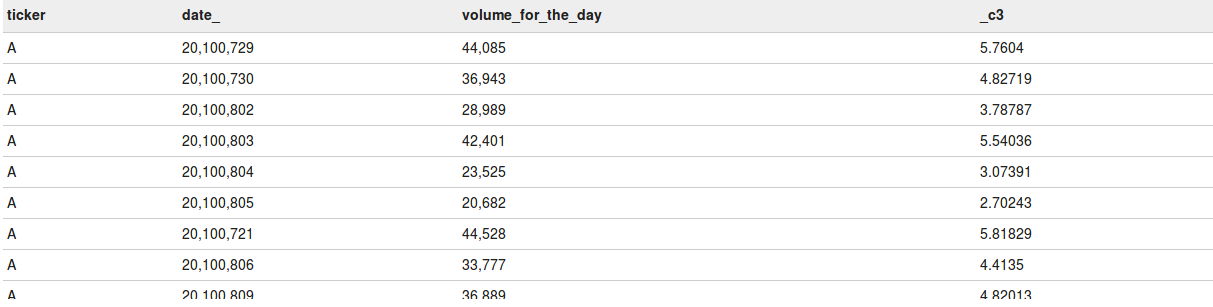
****

**In the above screenshot, you can see the volume\_for\_the\_day for each day and the running total is the sum of volume\_for\_the\_day’s that are elapsed.**

#### **Finding the percentage of each row value**

**Now let’s take a scenario where you need to find the percentage of the volume\_for\_the\_day on the total volumes for that particular ticker and that can be done as follows.**

**select ticker,date\_,volume\_for\_the\_day,(volume\_for\_the\_day\*100/(sum(volume\_for\_the\_day) over(partition by ticker))) from acadgild.stocks**

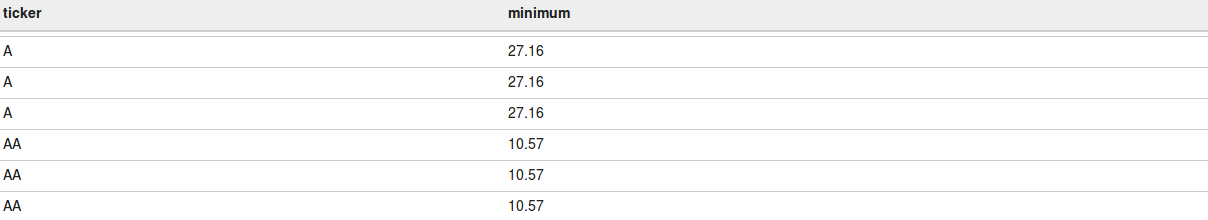
****

**In the above screenshot, you can see that the percentage contribution of the volumes for the day is found based on the total volume for that ticker.**

#### **Min**

**It returns the minimum value of the column for the rows in that over clause. From the below query, we can find the minimum closing stock price for each particular ticker.**

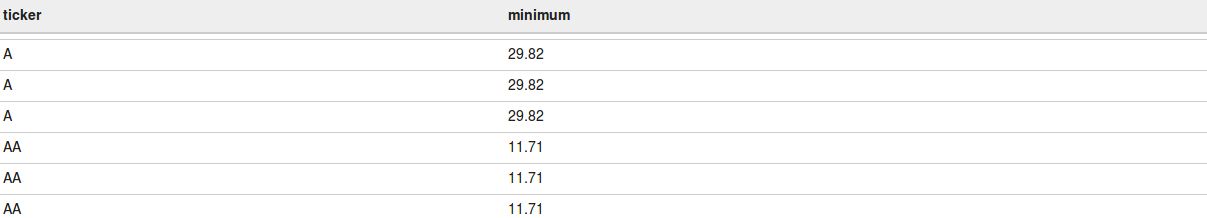
**select ticker, min(close) over(partition by ticker) as minimum from acadgild.stocks**

****

#### **Max**

**It returns the maximum value of the column for the rows in that over clause. From the below query, we can find the maximum closing stock price for each particular ticker.**

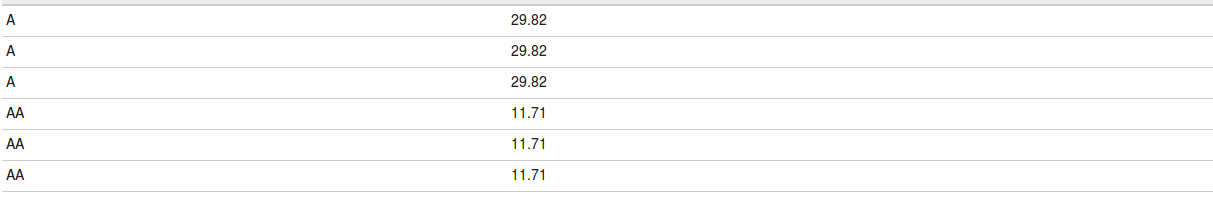
**select ticker, max(close) over(partition by ticker) as maximum from acadgild.stocks**

****

#### **AVG**

**It returns the average value of the column for the rows that over clause returns. From the below query, we can find the average closing stock price for each particular ticker.**

**select ticker, avg(close) over(partition by ticker) as maximum from acadgild.stocks**

****

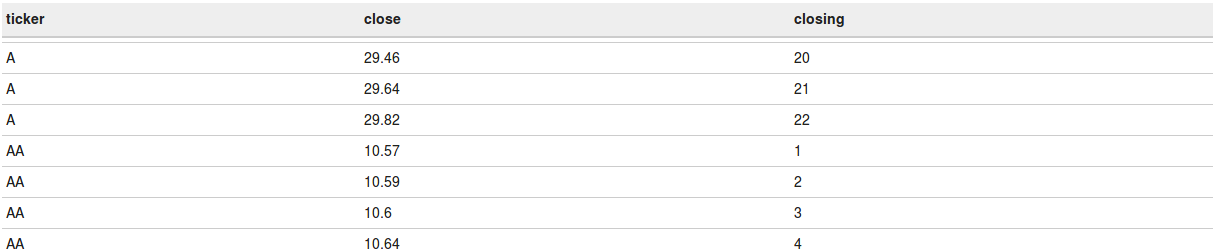
**Now let us work on some Analytic functions.**

#### **Rank**

**The rank function will return the rank of the values as per the result set of the over clause. If two values are same then it will give the same rank to those 2 values and then for the next value, the sub-sequent rank will be skipped.**

**The below query will rank the closing prices of the stock for each ticker. The same you can see in the below screenshot.**

**select ticker,close,rank() over(partition by ticker order by close) as closing from acadgild.stocks**

****

#### **Row\_number**

**Row number will return the continuous sequence of numbers for all the rows of the result set of the over clause.**

**From the below query, you will get the ticker, closing price and its row number for each ticker.**

**select ticker,close,row\_number() over(partition by ticker order by close) as num from acadgild.stocks**

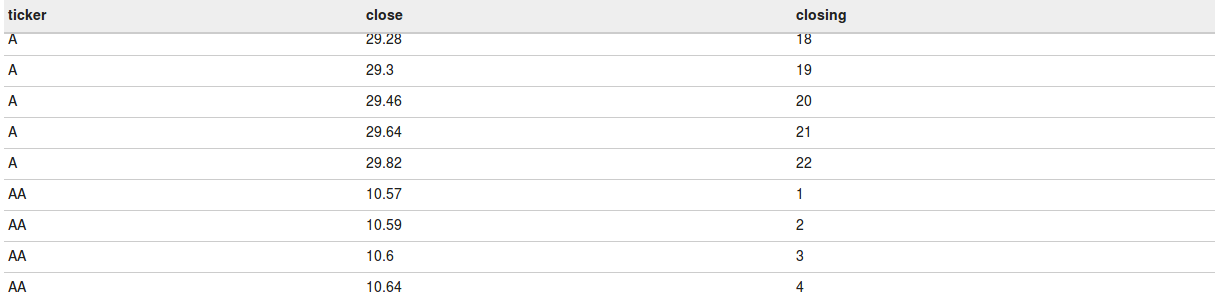
****

#### **Dense\_rank**

**It is same as the rank() function but the difference is if any duplicate value is present then the rank will not be skipped for the subsequent rows. Each unique value will get the ranks in a sequence.**

**The below query will rank the closing prices of the stock for each ticker. The same you can see in the below screenshot.**

**select ticker,close,dense\_rank() over(partition by ticker order by close) as closing from acadgild.stocks**

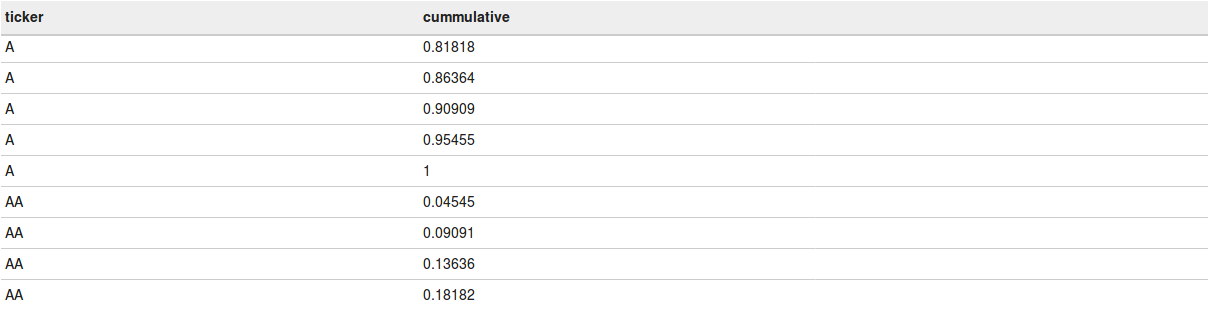
****

#### **Cume\_dist**

**It returns the cumulative distribution of a value. It results from 0 to 1. For suppose if the total number of records are 10 then for the 1st row the cume\_dist will be 1/10 and for the second 2/10 and so on till 10/10.**

**This cume\_dist will be calculated in accordance with the result set returned by the over clause. The below query will result in the cumulative of each record for every ticker.**

**select ticker,cume\_dist() over(partition by ticker order by close) as cummulative from acadgild.stocks**

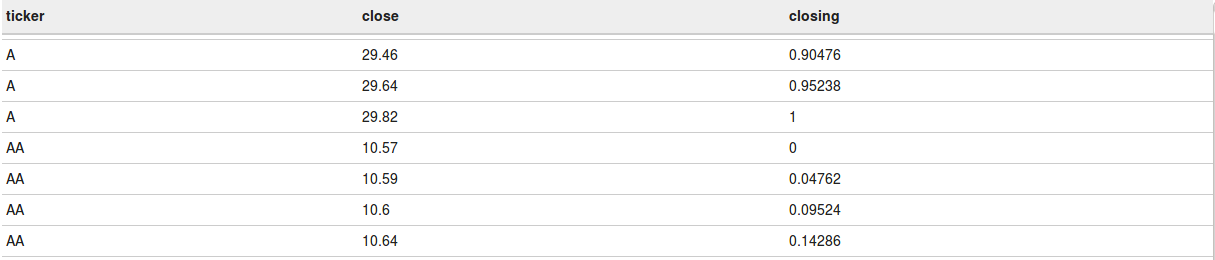
****

#### **Percent\_rank**

**It returns the percentage rank of each row within the result set of over clause. Percent\_rank is calculated in accordance with the rank of the row and the calculation is as follows (rank-1)/(total\_rows\_in\_group – 1). If the result set has only one row then the percent\_rank will be 0.**

**The below query will calculate the percent\_rank for every row in each partition and you can see the same in the below screen shot.**

**select ticker,close,percent\_rank() over(partition by ticker order by close) as closing from acadgild.stocks**

****

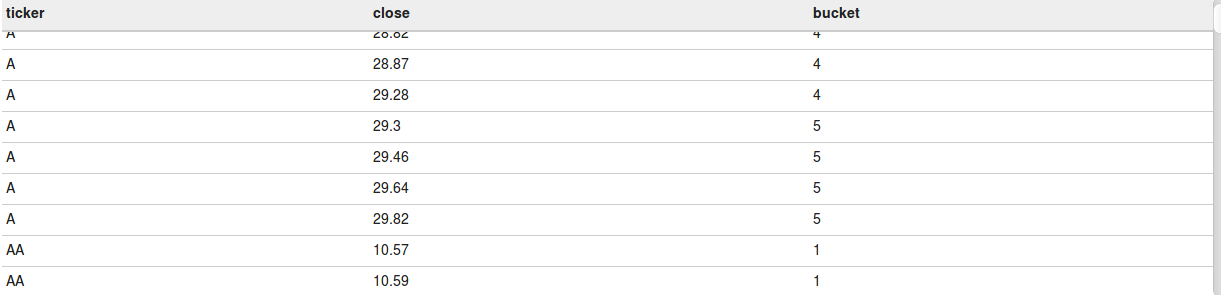
#### **Ntile**

**It returns the bucket number of the particular value. For suppose if you say Ntile(5) then it will create 5 buckets based on the result set of the over clause after that it will place the first 20% of the records in the 1st bucket and so on till 5th bucket.**

**The below query will create 5 buckets for every ticker and the first 20% records for every ticker will be in the 1st bucket and so on.**

**select ticker,ntile(5) over(partition by ticker order by close ) as bucket from acadgild.stocks**

**In the below screenshot, you can see that 5 buckets will be created for every ticker and the least 20% closing prices will be in the first bucket and the next 20% will be in the second bucket and so on till 5th bucket for all the tickers.**

****

# **Performance Tuning of an Apache Kafka/Spark Streaming System**

**The main issues for these applications were caused by trying to run a development system's code, tested on AWS instances on a physical, on-premise cluster running on real data. The original developer was never given access to the production cluster or the real data.**

**Apache Ignite was a huge source of problems, principally because it is such a new project that nobody had any real experience with it and also because it is not a very mature project yet.**

### **First target: Improve Spark Streaming performance**

**The Spark Streaming application was running in about 4.5 minutes, and the project goal was to run in about 30 seconds. We needed to find 9x speedup worth of improvements, and due to time constraints, we couldn’t afford to change any code!**

**The system had to be ready for production testing within a week, so the code from the architecture and algorithm point of view was assumed to be correct and good enough that we could reach the performance requirement only with tuning.**

**Fix RPC timeout exceptions**

**We found the correct solution from somebody having the same problem, as seen in**[**SPARK-14140 in JIRA**](https://issues.apache.org/jira/browse/SPARK-14140)**. They recommend increasing the spark.executor.heartbeatInterval from 10s to 20s.**

**I think this problem may be caused by nodes getting busy from disk or CPU spikes because of Kafka, Ignite, or garbage collector pauses. Since Spark runs on all nodes, the issue was random. (See the cluster services layout table in the first section.)**

**The configuration change fixed this issue completely. We haven’t seen it happen since.**

**Increase driver and executor memory**

**Out of memory issues and random crashes of the application were solved by increasing the memory from 20g per executor to 40g per executor as well as 40g for the driver. Happily, the machines in the production cluster were heavily provisioned with memory. This is a good practice with a new application, since you don’t know how much you will need at first.**

**The issue was difficult to debug with precision, lacking accurate information, since the Spark UI reports very little memory consumption. In practice, as this setting is easy to change, we empirically settled on 40g being the smallest memory size for the application to run stably.**

**Increase parallelism: increase number of partitions in Kafka**

**The input data was unbalanced, and most of the application processing time was spent processing Topic 1 (with 85% of the throughput). Kafka partitions are matched 1:1 with the number of partitions in the input RDD, leading to only 36 partitions, meaning we can only keep 36 cores busy on this task. To increase the parallelism, we need to increase the number of partitions. So we split topic 1 into 12 topics each, with 6 partitions, for a total of 72 partitions. We did a simple modification to the producer to divide the data evenly from the first log into 12 topics, instead of just one. Zero code needed to be modified on the consumer side.**

**Right-size the executors**

**The original application was running only 3 executors with 72 total cores. We configured the application to run with 80 cores at a maximum of 10 cores per executor, for a total of 8 executors. Note that with 16 real cores per node on a 10-node cluster, we’re leaving plenty of resources for Kafka brokers, Ignite, and HDFS/NN to run on.**

**Increase the batch window from 30s to 1m**

**The data is pushed into Kafka by the producer as batches every 30s, as it is gathered by FTP batches from the remote systems. Such an arrangement is common in telecom applications due to a need to deal with equipment and systems from a bewildering range of manufacturers, technology, and ages.**

**This meant that the input stream was very lumpy, as shown in the screenshot of Spark UI's Streaming tab:**

**Increasing the window to 1m allowed us to smooth out the input and gave the system a chance to process the data in 1 minute or less and still be stable.**

# [**Limit Kafka batches size when using Spark Streaming**](https://stackoverflow.com/questions/39981650/limit-kafka-batches-size-when-using-spark-streaming)

**Batch size is product of 3 parameters**

1. **batchDuration: The time interval at which streaming data will be divided into batches (in Seconds).**
2. **spark.streaming.kafka.maxRatePerPartition: set the maximum number of messages per partition per second. This when combined with batchDuration will control the batch size. You want the maxRatePerPartition to be set, and large (otherwise you are effectively throttling your job) and batchDuration to be very small.**
3. **No of partitions in kafka topic**

**imiting the Max batch size will greatly help to control the processing time, however, it increase the processing latency of message.**

**By settings below properties, we could control the batch size spark.streaming.receiver.maxRate= spark.streaming.kafka.maxRatePerPartition=**

**You could even dynamically set the batch size based on processing time, by enabling the back pressure spark.streaming.backpressure.enabled:true spark.streaming.backpressure.initialRate:**

**Is-there-a-way-to-take-the-first-1000-rows-of-a-spark-dataframe**

**If I use df.take(1000) then I end up with an array of rows- not a dataframe, so that won't work for me.**

**The method you are looking for is**[**.limit**](https://spark.apache.org/docs/latest/api/scala/index.html#org.apache.spark.sql.Dataset@limit(n:Int):org.apache.spark.sql.Dataset[T])**.**

**Limit transformation Returns a new Dataset by taking the first n rows. The difference between this function and head is that head returns an array while limit returns a new Dataset.**

**df1.limit(2000).write.format("json").save("dbfs:/mnt/Samples/vehicle\_sample\_data/vinchannel")**

**What are the optimization techniques that can be used in Apache Spark?**

**Optimization Techniques in Spark**

**(i)Data Serialization - Java Serialization, Kyro serialization**

**(ii)Memory Tuning - Data Structure tuning, Garbage collection tuning**

**(iii)Memory Management - Cache() and Persist()**

1. **To decrease the size of object used Spark Kyro serialization which is 10 times better than default java serialization.**
2. **Instead of re partition use coalesce ,this will reduce no of shuffles.**
3. **Cache or persist data/rdd/data frame if the data is to used further for computation.**

**2. Spark SQL WINDOW operation:**

**Aggregate/Analytic and ranking functions uses spark sql window operation.**

[**https://knockdata.github.io/spark-window-function/**](https://knockdata.github.io/spark-window-function/)**.**

**How to get top 2 salaries for every department using spark code**

**The following example keeps the top 2 employees salary wise, others have to go. On the sample dataset, Wilma and Maja have the same salary. Maja has to go according to order, unfortunately.**

**val overCategory = Window.partitionBy('depName).orderBy('salary desc)**

**val df = empsalary.withColumn(**

**"row\_number", row\_number() over overCategory).filter(**

**'row\_number <= 2).select(**

**"depName", "empNo", "name", "salary")**

**df.show(false)**

**Difference between repartition and coallase ? What and when is advantage ?**

**repartition - its recommended to use repartition while increasing no of partitions, because it involve shuffling of all the data.**

**coalesce- it’s is recommended to use coalesce while reducing no of partitions. For example if you have 3 partitions and you want to reduce it to 2 partitions, Coalesce will move 3rd partition Data to partition 1 and 2. Partition 1 and 2 will remains in same Container.but repartition will shuffle data in all partitions so network usage between executor will be high and it impacts the performance.**

**Performance wise coalesce performance better than repartition while reducing no of partitions.**

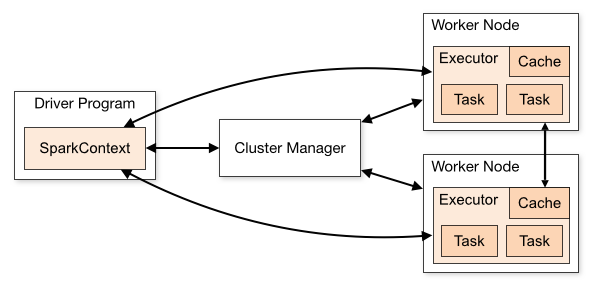
**Difference between coalesce and repartition**

**coalesce uses existing partitions to minimize the amount of data that's shuffled. repartition creates new partitions and does a full shuffle. coalesce results in partitions with different amounts of data (sometimes partitions that have much different sizes) and repartition results in roughly equal sized partitions.**

**Keep in mind that repartitioning your data is a fairly expensive operation. Spark also has an optimized version of repartition() called coalesce() that allows avoiding data movement, but only if you are decreasing the number of RDD partitions.**

**What is container and executor and how they are linked ?**

**The first fact to understand is: each Spark executor runs as a YARN container [2]. This and the fact that Spark executors for an application are fixed, and so are the resources allotted to each executor, a Spark application takes up resources for its entire duration.**

**[](https://i.stack.imgur.com/cwrMN.png)**

**Spark uses a master/slave architecture. As you can see in the figure, it has one central coordinator (Driver) that communicates with many distributed workers (executors). The driver and each of the executors run in their own Java processes.**

**DRIVER**

**The driver is the process where the main method runs. First it converts the user program into tasks and after that it schedules the tasks on the executors.**

**EXECUTORS**

**Spark Executor runs within a Yarn Container, not across Containers.**

**Executors run on a given Worker.**

**Executors are worker nodes' processes in charge of running individual tasks in a given Spark job. They are launched at the beginning of a Spark application and typically run for the entire lifetime of an application. Once they have run the task they send the results to the driver. They also provide in-memory storage for RDDs that are cached by user programs through Block Manager.**

**How to decide num of executors and executor memory in spark application ?**

### **Case 1 Hardware – 6 Nodes and each node have 16 cores, 64 GB RAM**

**First on each node, 1 core and 1 GB is needed for Operating System and Hadoop Daemons, so we have 15 cores, 63 GB RAM for each node**

**We start with how to choose number of cores:**

**Number of cores = Concurrent tasks an executor can run**

**So we might think, more concurrent tasks for each executor will give better performance. But research shows that any application with more than 5 concurrent tasks, would lead to a bad show. So the optimal value is 5.**

**This number comes from the ability of an executor to run parallel tasks and not from how many cores a system has. So the number 5 stays same even if we have double (32) cores in the CPU**

**Number of executors:**

**Coming to the next step, with 5 as cores per executor, and 15 as total available cores in one node (CPU) – we come to 3 executors per node which is 15/5. We need to calculate the number of executors on each node and then get the total number for the job.**

**So with 6 nodes, and 3 executors per node – we get a total of 18 executors. Out of 18 we need 1 executor (java process) for Application Master in YARN. So final number is 17 executors**

**This 17 is the number we give to spark using –num-executors while running from spark-submit shell command**

**Memory for each executor:**

**From above step, we have 3 executors per node. And available RAM on each node is 63 GB**

**So memory for each executor in each node is 63/3 = 21GB.**

**However small overhead memory is also needed to determine the full memory request to YARN for each executor.**

**The formula for that overhead is max(384, .07 \* spark.executor.memory)**

**Calculating that overhead:  .07 \* 21 (Here 21 is calculated as above 63/3) = 1.47**

**Since 1.47 GB > 384 MB, the overhead is 1.47**

**Take the above from each 21 above => 21 – 1.47 ~ 19 GB**

**So executor memory – 19 GB**

**Final numbers – Executors – 17, Cores 5, Executor Memory – 19 GB**

**What is standalone \clinet and cluster mode in spark ?**

**Spark currently supports two deploy modes. In client mode, the driver is launched in the same process as the client that submits the application. In cluster mode, however, the driver is launched from one of the Worker processes inside the cluster, and the client process exits as soon as it fulfills its responsibility of submitting the application without waiting for the application to finish.**

**/bin/spark-submit \**

**--class <main-class>**

**--master <master-url> \**

**--deploy-mode <deploy-mode> \**

**--conf <key>=<value> \**

**... # other options**

**<application-jar> \**

**[application-arguments]**

**When working in Cluster mode, all JARs related to the execution of your application need to be publicly available to all the workers. This means you can either manually place them in a shared place or in a folder for each of the workers.**

**Driver opens up a dedicated Netty HTTP server and distributes the JAR files specified to all Worker nodes (big advantage).**

**i. Spark Client Mode**

**As we know, the behavior of spark job depends on the “driver” component. So here,”driver” component of spark job will run on the machine from which job is submitted, this is what we call “client mode”.**

* **When job submitting machine is within or near to “spark infrastructure”. Since there is no high network latency of data movement for final result generation between “spark infrastructure” and “driver”, then, this mode works very fine.**
* **When job submitting machine is very remote to “spark infrastructure”, also have high network latency. Hence, in that case, this spark mode does not work in a good manner.**

**ii. Spark Cluster Mode**

**Else, “driver” component of spark job will not run on the local machine from which job is submitted, this is what we call “cluster mode”. In addition, here spark job will launch “driver” component inside the cluster.**

* **When job submitting machine is remote from “spark infrastructure”. Since, within “spark infrastructure”, “driver” component will be running. Thus, it reduces data movement between job submitting machine and “spark infrastructure”. In such case, This mode works totally fine.**
* **While we work with this spark mode, the chance of network disconnection between “driver” and “spark infrastructure” reduces. Since they reside in the same infrastructure. Also, reduces the chance of job failure.**

**But in case of --deploy-mode cluster:**

1. **You won't be able to see the detailed logs in the terminal.**
2. **Since driver is not created in the Master itself, you won't be able to terminate the job from the terminal.**

**But in case of --deploy-mode client:**

1. **You will be able to see the detailed logs in the terminal.**
2. **You will be able to terminate the job from the terminal itself.**

**These are the basic things that I have noticed till now.**

**TCS:**

**1) which distributed system u Use for hadoop?**

**2)When will use Bucketing and partioning, what is difference b/t them.**

**3)what is map side join ? and reduce side join ?**

**4) what is broadcast in Spark?**

**5) sqoop command to import all the tables in one database except any 4 tables, join that 4 tables and import all the data.**

**sqoop import-all-tables --connect jdbc:mysql://localhost/sqoop --username root --password hadoop --target-dir '/Sqoop21/AllTables'**

**sqoop import-all-tables --connect jdbc:mysql://localhost/sqoop --username root --password hadoop --target-dir '/Sqoop21/AllTables' --exclude-tables <table1>,<tables2>**

**sqoop import-all-tables --connect jdbc:mysql://localhost/sqoop --username root --password hadoop --warehouse-dir '/Sqoop'**

**6)Why to use hive ? difference between impala and hive ? is there any difference performence wise ?**

**7) why Spark sql ?**

**Capgemini :**

**1) Roles and responsibilities and project.**

**2) scope of hbase and hadoop in your project.**

**3) diff b/w hlog and hfile.**

**4) Hbase architecture.**

**5) role of zookeeper in hbase.**

**6) types of compaction.**

**7) suppose if i have billions of small files, how can i store in hdfs.**

**8) what is bulk loading ?**

**9)how to do bulk loading by Java API**

**10) hive to hbase integration.**

**11) performance techniques in hive.**

**12) diff b/w partiotioning and bucketing.**

**13) how to do update in Hbase.**

**14) how to delete operation habase.**

**15) how to update operation in hive**

**16)word count program in hive and pig**

**17)how to display first 10 rows in hbase**

**18) what is need of hive and pig in your project.**

**19) last updated value and last modified date in sqoop**

**20)example of flume configuration**

**21) Oozie- asked about job.properties and workflow.xml**

**22) what are the things in above files.**

**23) why you are using ORC table and AVRO tables.**

**24) different file formats.**

**Capgemini by me:**

**1) Mar reduce - Job flow**

**1a) what is record reader , how inputs come to this ? which java api**

**2)What is delta load in Hive.**

**3) what is incremental load in hive, how you do it.**

**4) Possible to delete and update one row in hive ?**

**He said yes its possible**

**5)you dont have access to hive prompt, how you check table is partitioned or not?**

**6)2 tables are there need to copy from one table to other table , how to do that ?**

**6) static loading and dynamic loading in hive ?**

**ITC Infotech:**

**1)explain about your project architecture.**

**2) how to submit a spark job?**

**3) what are trasformations and actions.**

**4) how to create RDDs? and ways ?**

**5) what is use of partitions?**

**6) write a query to bucketing**

**7) write a query for 2nd max of table in hive.**

**8) how to get duplicate records.**

**L&T Infotech:**

**1) write a query to get the duplicate list from a table.**

**2) what is ranking**

**3)explain YARN architechture**

**4)**

**Infosys by Me:**

**1) what is the ETL tool you are using in your project.**

**ETL = Extract, transform, Load**

* **Informatica Power center.**
* **Data Stage.**
* **Ab initio.**
* **SSIS.**
* **Talend.**
* **Pentaho.**
* **ODI (Oracle Data Integrator)**
* **SAS ETL.**

**OLTP (On-line Transaction Processing) is characterized by a large number of short on-line transactions (INSERT, UPDATE, DELETE). The main emphasis for OLTP systems is put on very fast query processing, maintaining data integrity in multi-access environments and an effectiveness measured by number of transactions per second. In OLTP database there is detailed and current data, and schema used to store transactional databases is the entity model (usually 3NF).   
  
- OLAP (On-line Analytical Processing) is characterized by relatively low volume of transactions. Queries are often very complex and involve aggregations. For OLAP systems a response time is an effectiveness measure. OLAP applications are widely used by Data Mining techniques. In OLAP database there is aggregated, historical data, stored in multi-dimensional schemas (usually star schema).**

**We can divide IT systems into transactional (OLTP) and analytical (OLAP). In general we can assume that OLTP systems provide source data to data warehouses, whereas OLAP systems help to analyze it.**

**2) what are the mode that runs Hive ?**

**A) Mapreduce, local**

**3) Where to change these running mode ? and how to check hive running on which mode?**

**4) diff between partition and bucketing ?**

**when cardinality is there then partition , else bucketing.**

**5) Syntax of views and external table creations.**

**6) have you worked on creating UDFS?**

**7) YARN runs on where ?**

**8) YARN Components?**

**9) how to see MYSQL service is running or not?**

**10) how see linux os version ?**

**Ans) lsb\_release -a**

**Uname -r**

**11) how to change replication factor in hdfs for a file ?**

**hadoop fsck /user/training/file1.txt -files -blocks -location -racks**

**12) how to see replication faction factor ?**

**A) you see in the web ui thru name node 50070 access file system, you can see each file replication factor.**

**B) hadoop fs -stat %r /user/training/file1.txt**

**13) how to change block size ? where to change in file configuration.**

**14) default port for name node? (50070)**

**15) oozie work flow where you write? which file?**

**16) how to submit oozie job?**

**$ oozie job -oozie http://localhost:11000/oozie -config examples/apps/map-reduce/job.properties -run**

**.**

**job: 14-20090525161321-oozie-tucu**

**Check the workflow job status:**

**$ oozie job -oozie http://localhost:11000/oozie -info 14-20090525161321-oozie-tucu**

**.**

**.----------------------------------------------------------------------------------------------------------------------------------------------------------------**

**Workflow Name : map-reduce-wf**

**App Path : hdfs://localhost:8020/user/tucu/examples/apps/map-reduce**

**Status : SUCCEEDED**

**Run : 0**

**User : tucu**

**Group : users**

**Created : 2009-05-26 05:01 +0000**

**Started : 2009-05-26 05:01 +0000**

**Ended : 2009-05-26 05:01 +0000**

**Actions**

**.----------------------------------------------------------------------------------------------------------------------------------------------------------------**

**Action Name Type Status Transition External Id External Status Error Code Start Time End Time**

**.----------------------------------------------------------------------------------------------------------------------------------------------------------------**

**mr-node map-reduce OK end job\_200904281535\_0254 SUCCEEDED - 2009-05-26 05:01 +0000 2009-05-26 05:01 +0000**

**.----------------------------------------------------------------------------------------------------------------------------------------------------------------**

**Mphasis By Me:**

**1) when you run scoop import how data in the form of ?**

**2) when scoop job submit what would happen in terms of map reduce ?**

**3) how many map tasks will launch when scoop job submits?**

**By default, four tasks are used.**

**4) how to schedule scoop job?**

**5) which type of tables in hive you prefer when you have 500gb of data? managed table or external table why?**

**5A) what would happen in managed tables?**

**5B) have you used ORC file formate? will it support ACID transactions ?**

**6) what would happen in sort by and reduce by key?**

**7)How date format will be stored in hive ? Write date format**

**8) what is accumulator?**

**9) what is operator overloading ?**

**10) a = b + c what would happen in scala how is works?**

**11)what is groupbykey and reducebykey(\_.\_) ? how internally it works in scala explain?**

**12) how to print RDD?**

**Println(outrdd.collect.toList())**

**13) what is collect ? not action collect?**

**14) how a file read when you submit a spark application? what happens next?**

**15)what you did in spark SQL?**

**16) What is UDTF?**

**17) Explode ?**

**Anjireddy@dvs: Hive varaible**

**Object inspector**

**Cosildation in hive**

**Mapreduce n YARN difference**

**Spark n mapreduce difference**

**RDD n data frames in spark**

**Sqoop import**

**Hive views**

**Hive external n managed**

**HBase n Hive differences**

**Orderby,sortby n clustered by**

**Speculative execution**

**Alter column command in hive**

**IBM interview questions on 01-03-2017**

**What is lazy evaluation in pig?**

**What is dynamic partition and static partition in hive?**

**What is the use of partitions and bucketing in hive?**

**Explain the flow of mapreduce program?**

**What is default partition in mapreduce and how can we override it?**

**What is difference between key class and value class in mapreduce?**

**What is the level of sub queries in hive?**

**What is transformation and action in spark?**

**#################### ########################################################**

**[6/3, 5:26 PM] Anjireddy@dvs: Hive varaible**

**Object inspector**

**Hive uses ObjectInspector to analyze the internal structure of the row object and also the structure of the individual columns. ObjectInspector provides a uniform way to access complex objects that can be stored in multiple formats in the memory, including: •Instance of a Java class (Thrift or native Java) •A standard Java object (we use java.util.List to represent Struct and Array, and use java.util.Map to represent Map) •A lazily-initialized object (For example, a Struct of string fields stored in a single Java string object with starting offset for each field) A complex object can be represented by a pair of ObjectInspector and Java Object. The ObjectInspector not only tells us the structure of the Object, but also gives us ways to access the internal fields inside the Object.**

**Cosildation in hive**

**Mapreduce n YARN difference**

**Spark n mapreduce difference**

**RDD n data frames in spark**

**Sqoop import**

**Hive views**

**Hive external n managed**

**HBase n Hive differences**

**Orderby,sortby n clustered by**

**Speculative execution**

**Alter column command in hive**

**IBM interview questions on 01-03-2017**

**What is lazy evaluation in pig?**

**What is dynamic partition and static partition in hive?**

**What is the use of partitions and bucketing in hive?**

**Explain the flow of mapreduce program?**

**What is default partition in mapreduce and how can we override it?**

**What is difference between key class and value class in mapreduce?**

**What is the level of sub queries in hive?**

**What is transformation and action in spark?**

**##########################################################################**

**NEWT:**

**1) diff b/w class and object in SCALA**

**2) how do you remove duplicates in DATAFRAME or in RDD**

**3) what are triats ?**

**4) do you write classes in your program ?**

**5) How do you count records in Sqoop**

**6) diff btween hadoop 1.x and 2.x**

**1)Command to shut down hdfs?**

**$ stop-dfs.sh**

**$shut-dfs.sh**

**dfs.sh stop**

**2) process to communicate among different nodes in hadoop?**

**rest api,rpc,rmp, http**

**3)**

**ITC Infotech: 19 - Aug**

**1) How do you shedule oozie jobs ? where you see them ? how do you access them how to access them?**

**2) diff b/w mr1 and mr 2 ?**

**3)which cluster distribition u r using ? which version? which CM(cloud manger) and version?**

**4) what are the recent issues that you faced in hive ?**

**5) explain MR job flow?**

**6) How to see databses in Hbase ?**

**7) Limitations of Sqoop ?**

**8) how bid ur cluster? how mny NN?**

**4) diff b/w spark submit and spark shell**

**5) diff b/w spark 1.x and 2 ?**

**6)spark is involved in CDH?**

**5) what is broker and consumer?**

**6)explain streaming flow ?**

**JPMC :**

**1) how to get data from streaming to spark?**

**2) have you worked on Lamda architechture and what is is**

**2) which API u use to the data?**

**3) which tool you use to deploy the code in production?**

**4) what is offset mngment in KAfka**

**5) how do you achieve fault tolorent in Kafka**

**5) what is options in SCALa, you worked on implicits ?**

**6) what scala Case class statement, annomous fuctions, static you use in scala?**

**7) shared variable in scala**

**8) foreach - what will happen internally.**

**8) what is flatmap?**

**9) what is Map partitions**

**10) what is implicit in scala ?**

**10) What are the OOzie components.**

**11) Yarn architecture**

**12) What is collect and why you use it and what are its drawbacks.**

**13) YARN is required to install in all machines ? Why ?**

**14) what is Yarn in client mode and yarn cluster mode? w hy to go for it**

**15) what is executor memory how do you declare it, and no.of executor?**

**16) where to see the job logs?**

**17 ) Hive or spark to be installed in all machines?.**

**18) Cytalist optimizer.**

**Applied Materiels**

**1) When you submit scoop what happens internally**

**2) when there is no non numeric col how you import the data, is its possible or not? y ?**

**3) what is Monod in scala?**

**Second time:**

**1) How to create directory in hadoop?**

**2) How to manage offset after kafka goes down? where this offset is maintained**

**3) i have 10 gb file , col1 , col2, col3 and 4 , 4th column as name, i want to search name. how u search in spark sql**

**4) in slidind interval, there is a delay in processing between among them wat wud be the reason , wat you do on this case how you trouble shoot.**

**5) In sqoop how to import if there are no primary key?**

**and how to import if there is no numeric key in table ?**

**6) what are the different compression techniques?**

**7) diff between avro and parque file formats? how they are usefull**

**8) what format you use when you go for hive and when you go for spark?**

**9) currying in scala?**

**10) what is crud operations , in ORC format**

**All State:**

**1) what is functinal programming ? where do you use ?**

**2) how do u match 2 rdd's.**

**3) what is tail recurssion**

**4)**

## **Basic Interview Questions**

### **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.**

### **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.**

### **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 data.**

## **Advanced Interview Questions**

### **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.**

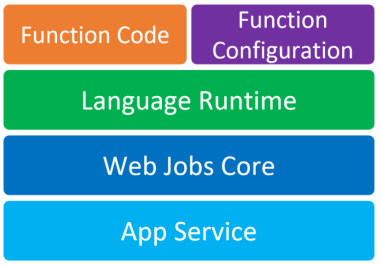
* **Role-Based Access Control (RBAC). RBAC includes built-in Azure roles such as reader, contributor, owner or custom roles. Typically, RBAC is assigned for two reasons. One is to specify who can manage the service itself (i.e., update settings and properties for the storage account). Another reason 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.**

### **What is Azure Active Directory and how it is used?**

**Ans. Microsoft offers Azure active directory, a fully managed multi-tenant service that implements identity and access capabilities for applications running in Azure as well as applications operating in the on-premises environment. It is used for providing single sign-on and multi-factor authentication to help users from protecting attacks.**

## **What is Azure Function?**

**Azure Function is a Serverless Compute Service that Runs code on Demand like Events or External-Invoke. Azure Functions can Scale up Automatically based on Demand. Azure functions are the evolution of Web Jobs. You can develop functions in C#, Node, Java, Python etc. Internally, Azure functions use App services.**

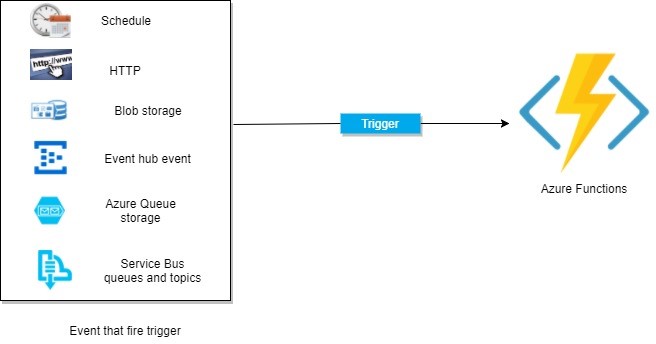
****

**We can use Functions for Backend Services, Event-based Processing like Data Table creation on File Upload, Scheduled Tasks etc.**

## **Azure Function Triggers**

**Triggers are what cause a function to run. A trigger defines how a function is invoked and a function must have exactly one trigger. Triggers have associated data, which is often provided as the payload of the function.**

## **Types of Triggers**

****

**Now let’s see some of the most common types of triggers available in Azure:**

### **Timer Trigger**

**This trigger is called on a predefined schedule. We can set the time for execution of the**[**Azure Function**](https://www.serverless360.com/azure-functions)**using this trigger.**

### **Blob Trigger**

**This trigger will get fired when a new or updated blob is detected. The blob contents are passed on as input to the function.**

### **Event Hub Trigger**

**This trigger is used for the application instrumentation, the user experience, workflow processing, and the**[**Internet of Things**](https://subscription.packtpub.com/search?released=Available&category=IoT%20%26%20Hardware)**( [IoT](https://subscription.packtpub.com/search?released=Available&category=IoT%20%26%20Hardware)). This trigger will get fired when any events are delivered to an Azure Event Hub.**

### **HTTP Trigger**

**This trigger gets fired when the HTTP request comes.**

### **Queue Trigger**

**This trigger gets fired when any new messages come in an Azure Storage Queue.**

### **Generic Webhook**

**This trigger gets fired when the Webhook HTTP requests come from any service that supports Webhooks.**

### **GitHub Webhook**

**This trigger is fired when an event occurs in your GitHub repositories. The GitHub repository supports events such as Branch created, delete branch, issue comment, and Commit comment.**

### **Service Bus Trigger**

**This trigger is fired when a new message comes from a service bus queue or topic.**

## **Logic Apps & Data Factory**

**While they both leverage a serverless orchestration I think they both shine in their own perspective: Azure Logic Apps is perfect for application integration while Data Factory is excellent for doing data integration.**

**Azure Data Factory allows you to interact with your data at scale by stitching together all your data stores together and build a data-centric platform inside your company ranging from copying data from one place to another, transforming data sets, loading data with bulk imports and much more. It is fully optimized for data processing and it ensures you don’t need to worry.**

**On the other hand, I see Azure Logic Apps being more focused on application integration where you can use it to unify all your internal & external services to create a unified infrastructure on which you can run your business processes and improve your company as a whole. The difference here is that it’s not focussed on the data itself but more the integration and connectivity with all these systems and how they communicate.**

**As with every Azure services it’s not about which service is better than the other, it’s about using the correct tool to get the job done.**

**The general rule here is that if it’s a data-centric workflow, Data Factory is probably your best bet. However, I think that combining Azure Logic Apps with Azure Data Factory is really the sweet spot.**

****

**By chaining these two orchestrators together you can create fully automated “pipelines” that join forces to achieve an end goal.**

**You can for example use an Azure Logic App that is in charge of data preparation by making data sets available in a specific data source which triggers Azure Data Factory for further processing. The triggered pipeline then picks up the data in the data source, processes it and when it’s finished it triggers another Azure Logic App to act on the gained business insights. This could be a Logic App that interprets the data, creates a summary of today’s sales and sendsout an email to the CEO so that he is aware of how his company is doing.**

**What is Azure Active Directory and what is its purpose?**

**Ans:**

**Azure Active directory is a comprehensive identity and access management Cloud solution; it combines directory services, advanced identity governance, application access management and a rich standards-based platform for you.**

**As you know, Windows Azure Active Directory is a multi-tenant Service, that provides an enterprise level identity and access management for the Cloud, built to support global scale, reliability and availability.**

**Some points are as follows about Windows Azure Active Directory, which are:**

* **For Azure Active Directory, you must have a Microsoft Account.**
* **Afterwards, I will create a new Windows Azure Active Directory.**
* **Subsequently, I’ll add the users to the directory as either a user or a global admin.**
* **The next step will be optionally enabling multi-factor authentication for the user.**
* **Afterwards, I’ll optionally add the user as a co-administrator for the subscription**

**What Is Azure Databricks?**

* **Ans:**
* **Azure Databricks is a fast, easy and collaborative Apache® Spark™ based analytics platform optimized for Azure. Designed in collaboration with the founders of Apache Spark, Azure Databricks combines the best of Databricks and Azure to help customers accelerate innovation with one-click setup; streamlined workflows and an interactive workspace that enables collaboration between data scientists, data engineers, and business analysts.**
* **As an Azure service, customers automatically benefit from native integration with other Azure services such as Power BI, SQL Data Warehouse, Cosmos DB as well as from enterprise-grade Azure security, including Active Directory integration, compliance, and enterprise-grade SLAs.**

**What are the different types of services offered in the cloud?**

|  |  |  |
| --- | --- | --- |
| **IAAS** | **PAAS** | **SAAS** |
| **In infrastructure as a service, you get the raw hardware from your cloud provider as a service i.e you get a server which you can configure with your own will** | **Platform as a Service, gives you a platform to publish without giving the access to the underlying software or OS** | **You get software as a service in Azure, i.e no infrastructure, no platform, simple software that you can use without purchasing it.** |
| **For Example: Azure VM, Amazon EC2.** | **For example: Web Apps, Mobile Apps in Azure.** | **For example: when you launch a VM on Azure, you are not buying the OS, you are basically renting it for the time you will be running that instance.** |

**22) What are the different cloud deployment models?**

* **Ans:**
* **Following are the three cloud deployment models:**
* **Public Cloud: The infrastructure is owned by your cloud provider and the server that you are using could be a multi-tenant system.**
* **Private Cloud: The infrastructure is owned by you or your cloud provider gives you that service exclusively. For eg: Hosting your website on your servers, or hosting your website with the cloud provider on a dedicated server.**
* **Hybrid Cloud: When you use both Public Cloud, Private Cloud together, it is called Hybrid Cloud. For Example: Using your in-house servers for confidential data, and the public cloud for hosting your company’s public facing website. This type of setup would be a hybrid cloud.**

# **Azure DevOps Interview Questions**

1. **What is the need for DevOps?**

**In Traditional software development, after completing the development part, the code deployment time was huge. And many times, we heard the common fights between the Development Team and Operations Team or deployment team that it works fine on our system, it's the sever causing problem and operation team defenses it's not your server it's your code, Right? Well, DevOps solves the Traditional Dev and Ops fights by breaking the wall of confusion.**

1. **How DevOps Works?**

**DevOps is the practice of operations and development engineers that work together in the entire project lifecycle, from the design and development process to production releases and support.**

**Starting from design and development to testing automation and from continuous integration to continuous delivery, the team works together to achieve the desired goal. People having both development and operations skill sets working together and use various tools for CI-CD and Monitoring to respond quickly to customer's need and fix issues and bugs.**

1. **What is Azure DevOps? What is the difference between Azure DevOps and VSTS Online?**

**Microsoft Visual Studio Team Services, now known as Azure DevOps having excellent application lifecycle management tool.**

**We can plan a project with Agile tools and templates, manage and run test plans, Version control source code and manage the branches, deploy the solution across all platform using Azure Pipelines, by implementing Continuous Instigation and Continuous Deployment.**

1. **What services Azure DevOps Provides?**

**Azure DevOps provides full application lifecycle management from planning to coding, and from testing to build and deploy.**

1. **What is Azure Boards?**

**Azure board provides service to manage your works, using the Agile Scrum and Kanban templates, Dashboard that we can customize and reporting.**

1. **What are Azure Repos?**

**Azure Repos is a code version control system that can manage your code and its version.**

**Using that we can track the changes, whenever team edits code it has all the version history so later, we can coordinate with the team and merge the changes.**

**The azure repo has both a centralized version control system as well as a distributed version control system.**

**Git: Distributed Version Control System**

**Team FoundationVersion Control (TFVC): Centralized Version Control System.**

1. **What are Azure Pipelines?**

**Azure Pipelines has all the features that are required for supporting Continuous Integration (CI) and Continuous Deployment (CD).**

**Using that we can constantly test and build the code and release it to any target.**

1. **What are Azure Test Plans?**

**Azure test plan provides browser-based test management using that we can manage all the testing like Exploratory & manual testing, Continuous testing, Unit & functional testing also we can ask or Request stakeholder to provide feedback.**

1. **What are Azure Test Plans?**

**Azure Artifacts is the service using that we create, host and share packages with teams. We can share code across teams, and manage all package types like NuGet, Marven, npm, Gradle etc.**

#### **List out the top DevOps tools**

1. **The most popular DevOps tools are given below:**

* **Git: Version Control System tool**
* **Jenkins: Continuous Integration tool**
* **Selenium: Continuous Testing tool**
* **Puppet, Chef, Ansible: Configuration Management and Deployment tools**
* **Nagios: Continuous Monitoring tool**
* **Docker: Containerization tool**