Introduction Azure dat afactory

1. Introduction

Azure Data Factory (ADF) is a fully managed, serverless data integration and ETL (Extract, Transform, Load) service provided by Microsoft Azure. It enables you to ingest, prepare, transform, and orchestrate data at scale from various on-premises and cloud data sources into centralized data stores for analytics, reporting, and machine learning workloads. Azure Data Factory plays a key role in modern data engineering by helping organizations build end-to-end data pipelines without managing any physical infrastructure.

2. Core Terminology and Concepts

2.1 Linked Services

A Linked Service in Azure Data Factory is similar to a connection string. It defines the connection information required for ADF to connect to external data sources and destinations. Linked Services store configuration such as server names, database names, authentication methods, keys, and endpoints. Every Dataset and activity that interacts with a data store must reference an appropriate Linked Service. Examples of Linked Services: - Azure SQL Database Linked Service using SQL authentication or Managed Identity. - Azure Blob Storage Linked Service using an access key or Shared Access Signature (SAS). - On-premises SQL Server Linked Service using a self-hosted Integration Runtime. - REST API Linked Service using a base URL and authentication details.

2.2 Datasets

A Dataset in Azure Data Factory represents the structure and location of the data that you want to work with. While Linked Services define how to connect to a data source, Datasets define what data within that source should be used. A Dataset can describe a table, a view, a file, a folder, or a set of files. Datasets also define schema information such as column names and data types when needed. Examples of Datasets: - A Dataset representing the dbo.Customers table in an Azure SQL Database. - A Dataset representing a CSV file named CustomerData.csv in an Azure Blob Storage container. - A Dataset representing a folder containing multiple parquet files in Azure Data Lake Storage Gen2.

2.3 Pipelines

A Pipeline is a logical container in Azure Data Factory that groups together one or more activities to accomplish a complete data operation. Pipelines help organize and manage complex workflows. You can design a pipeline to copy data, transform data, execute stored procedures, call REST APIs, or control execution flow using activities such as If Condition or ForEach. Example: A pipeline that first copies sales data from an on-premises SQL Server to Azure Data Lake Storage Gen2, then transforms the data using Mapping Data Flows, and finally loads the transformed data into Azure Synapse Analytics.

2.4 Activities

Activities represent individual tasks within a pipeline. Each activity performs a specific operation such as copying data, transforming data, calling a web service, or executing a stored procedure. Pipelines may contain one or many activities chained together. Common types of activities: - Copy Activity: Used for data movement from a source to a sink (destination). - Data Flow Activity: Used for visual data transformation using Mapping Data Flows. - Web Activity: Used to invoke a REST endpoint. - Stored Procedure Activity: Used to execute a stored procedure in a database. - Validation and Lookup Activities: Used to check and retrieve data before further processing.

2.5 Integration Runtime

Integration Runtime (IR) is the compute infrastructure that Azure Data Factory uses to perform data movement, transformation, and activity dispatch. It acts as the bridge between the Data Factory service and your data stores or compute services. Types of Integration Runtime: - Azure Integration Runtime: Used for data movement and transformation within Azure and between supported cloud data stores. - Self-hosted Integration Runtime: Installed on your on-premises machine or virtual network to connect to on-premises or private network data sources. - Azure-SSIS Integration Runtime: Used to lift and shift existing SQL Server Integration Services (SSIS) packages to Azure.

2.6 Triggers

Triggers in Azure Data Factory define when a pipeline should be executed. Instead of manually running a pipeline, you can configure triggers to schedule or automatically start pipelines based on specific events. Common trigger types: - Schedule Trigger: Runs a pipeline at a specific time or on a recurring schedule (for example, every day at 1:00 AM). - Tumbling Window Trigger: Processes data in fixed-size, non-overlapping time slices, often used for time-based batch processing. - Event-based Trigger: Starts a pipeline when a specific event occurs, such as a new file being created in a storage account.

2.7 Copy Activity

Copy Activity is one of the most frequently used activities in Azure Data Factory. It is responsible for copying data from a source dataset to a sink dataset. Internally, the Copy Activity reads data from the source by using the source Linked Service, optionally applies column mappings and type conversions, and then writes the data to the sink by using the sink Linked Service. Key features: - Supports a wide range of source and sink connectors (Azure SQL, Blob Storage, REST, Amazon S3, Oracle, and many more). - Supports performance optimization with parallel copies, batch sizes, and staging. - Supports column mapping and basic transformations during the copy process.

2.8 Mapping Data Flows

Mapping Data Flows provide a visual, code-free environment for designing complex data transformation logic within Azure Data Factory. You can build transformation graphs using drag-and-drop components such as joins, aggregates, pivots, and derived columns. Data Flows are executed on a Spark cluster managed by Azure, which allows transformations to scale automatically. Example usage: - Cleaning raw data from multiple sources. - Joining reference data with transactional data. - Aggregating and summarizing large datasets for reporting.

2.9 Source and Sink

In the context of Copy Activity and data movement, the term Source refers to the location from where data is read, while Sink refers to the destination where data is written. Both source and sink are defined through Datasets and Linked Services. ADF supports combinations such as cloud-to-cloud, on-premises-to-cloud, and cloud-to-on-premises data movement.

3. Step-by-Step Implementation Example

This section describes a detailed, practical implementation of an Azure Data Factory pipeline that copies data from an Azure SQL Database to Azure Blob Storage in CSV format. The scenario is a common use case where transactional data from a relational database is exported periodically to a data lake or storage account for downstream analytics.

3.1 Prerequisites

Before starting the implementation, ensure the following prerequisites are available: - An active Azure subscription. - An Azure SQL Database containing a sample table, for example dbo.Customers. - An Azure Storage Account with a Blob container, for example a container named output. - Appropriate permissions to create resources such as Azure Data Factory, Linked Services, and Storage Containers.

3.2 Create an Azure Data Factory

1. Sign in to the Azure portal. 2. In the left navigation pane, select "Create a resource". 3. Search for "Data Factory" and select Azure Data Factory. 4. Click "Create" and provide the required details: - Subscription: Choose your subscription. - Resource group: Select an existing resource group or create a new one. - Region: Select a region close to your data sources. - Name: Provide a unique name for your Data Factory instance. 5. Click "Review + create" and then click "Create" to deploy the Data Factory. 6. After deployment, open the Data Factory resource and click "Launch studio" to open the Data Factory Studio interface.

3.3 Create Linked Service for Azure SQL Database

1. In Data Factory Studio, switch to the "Manage" hub (gear icon). 2. Under the "Connections" section, select "Linked services". 3. Click on the "+ New" button to create a new Linked Service. 4. Choose "Azure SQL Database" as the connector type and click "Continue". 5. Provide a name, such as LS_AzureSQL. 6. Select the Azure subscription and choose your Azure SQL Server and database. 7. Configure authentication using SQL authentication, Managed Identity, or Service Principal. 8. Test the connection to ensure it is successful, then click "Create".

3.4 Create Linked Service for Azure Blob Storage

1. Still under "Linked services", click "+ New" again. 2. Select "Azure Blob Storage" as the connector type and click "Continue". 3. Provide a name, such as LS_BlobStorage. 4. Select your subscription and storage account name. 5. Choose an authentication method such as account key or Managed Identity. 6. Test the connection and click "Create" once validation is successful.

3.5 Create Source Dataset for Azure SQL Database

1. Switch to the "Author" hub in Data Factory Studio. 2. Under the "Factory Resources" pane, right-click "Datasets" and select "New dataset". 3. Choose "Azure SQL Database" as the dataset type and click "Continue". 4. Provide a name, for example DS_SQL_Customers. 5. Select the Linked Service LS_AzureSQL created earlier. 6. Choose the table dbo.Customers or specify a

query to select required columns. 7. Click "OK" or "Create" to save the dataset.

3.6 Create Sink Dataset for Azure Blob Storage (CSV)

1. Again, create a new dataset by right-clicking on "Datasets" and selecting "New dataset". 2. Select "Azure Blob Storage" as the type and then choose the "DelimitedText" (CSV) format. 3. Provide a name, such as DS_Blob_CustomersCsv. 4. Select the Linked Service LS_BlobStorage. 5. Specify the file path, for example: - Container: output - Directory: exported (optional) - File name: customers.csv 6. Configure the first row as header and specify the column delimiter (for example, comma ","). 7. Save the dataset configuration.

3.7 Create a Pipeline and Add Copy Activity

1. In the "Author" hub, click on "+" next to "Pipelines" and select "New pipeline". 2. Provide a meaningful name, such as PL_Copy_Customers_SQL_to_Blob. 3. From the "Activities" pane, drag and drop the "Copy data" activity onto the pipeline canvas. 4. Select the Copy Activity and go to the "Source" tab: - Choose the source dataset DS_SQL_Customers. - Optionally, specify a query if you want to filter or transform the data at the source. 5. Go to the "Sink" tab: - Select the sink dataset DS_Blob_CustomersCsv. - Configure additional options such as file naming behavior and whether to append or overwrite the file. 6. Optionally, go to the "Mapping" tab to explicitly map source columns to sink columns. 7. Click "Validate" to ensure that there are no errors in the pipeline configuration.

3.8 Publish and Trigger the Pipeline

1. After validation, click on the "Publish all" button to save and deploy the changes to the Data Factory. 2. Once publishing is complete, click on the "Add trigger" button at the top of the pipeline canvas. 3. Select "Trigger now" to run the pipeline manually, or associate an existing trigger for scheduled execution. 4. Confirm the settings and start the trigger. The pipeline will begin execution.

3.9 Monitor the Pipeline Run

1. Switch to the "Monitor" hub in Data Factory Studio. 2. Under the "Pipeline runs" section, you can see the status of the pipeline execution. 3. Click on a pipeline run to view detailed information such as activity runs, data volume copied, and execution duration. 4. If the run is successful, verify that the customers.csv file has been created in the specified Blob container by navigating to the Azure Storage Account in the Azure portal or using a tool such as Azure Storage Explorer.

4. Example Use Case Summary

Example Use Case: Daily Export of Customer Master Data Scenario: An organization maintains its customer master data in an Azure SQL Database. The analytics team needs a daily snapshot of this data in a centralized storage location for reporting and data science processing. Azure Data Factory is used to automate the daily export of the dbo.Customers table to Azure Blob Storage as a CSV file. Key steps implemented: - Created Linked Services for both Azure SQL Database and Azure Blob Storage. - Defined Datasets for the source SQL table and the destination CSV file. - Designed a pipeline with a Copy Activity to move data from the SQL table to the Blob file. - Configured a trigger (for example, a schedule trigger at midnight) to run the pipeline daily. - Monitored pipeline runs and validated data in the output storage container. This use case demonstrates how Azure Data Factory can be used to build automated, reliable, and repeatable data integration workflows with minimal operational overhead.