**1. What do Azure Data Factory's ARM Templates do? What do they serve?**

The infrastructure and setup for the data factory pipeline, such as linked services, pipeline activities, datasets, etc, are defined in an ARM template, which is a JSON (JavaScript Object Notation) file. The template's code will be nearly identical to that of our pipeline. When we want to move our pipeline code from Development to a higher environment, such as Production or Staging, after we are certain that the code is functioning properly, ARM templates come in handy.

**2. How can we in the Data Factory utilize code to higher environments?**

At a high level, the following series of actions will help us accomplish this:

* Make a feature branch where our code base will be kept.
* Once we're certain the code belongs in the Dev branch, start creating a request form to merge it.
* Publish the development branch's code to create ARM templates.
* As a result, code can be promoted to higher surroundings like Staging or Production using an automated [CI/CD DevOps pipeline](https://mindmajix.com/continuous-integration-in-devops).

**3. What are the three tasks that Microsoft Azure Data Factory supports?**

Data Factory supports the following activities: data movement, transformation, and control activities.

* **Movement of data activities**: As the name implies, these processes aid in the transfer of data.
* **Activities for data transformation**: These activities assist in data transformation as the data is loaded into the target or destination.
* **Control flow activities**: Control (flow) activities aid in regulating any activity's flow through a pipeline.

***[ Learn Complete***[***Azure Data Factory Tutorial***](https://mindmajix.com/azure-data-factory-tutorial)***]***

**4. What are the two categories of computing environments that Data Factory supports for the purposes of carrying out transform activities?**

The different compute environment types that Data Factory endorses for carrying out transformation activities are listed below: -

* On-Demand Computing Environment: ADF offers this completely managed environment. When performing this kind of calculation, a cluster is created to carry out the transformation activity, and it is automatically deleted once the task is finished.
* Bring Your Own Environment: If you have the infrastructure for on-premises services, you can use ADF to manage the computing environment in this scenario.

**5. What steps constitute an ETL process?**

The four main steps of the ETL also called as Extract, Transform, and Load process are as follows:

* **Integrate and Collect**: Attach to the data source(s) and transfer the data to crowdsourced and local data storage.
* Using computing services like HDInsight, [Hadoop](https://mindmajix.com/hadoop-tutorial), [Spark](https://mindmajix.com/apache-spark-tutorial), etc., for data transformation.
* **Publish**: To upload data to Azure Cosmos DB, Azure SQL databases, Azure Data Lake storage, etc.
* **Monitor**: Pipeline monitoring for Azure Data Factory is built-in and supported by Azure Monitor, API, [PowerShell](https://mindmajix.com/powershell-tutorial), Azure Monitor logs, and health panels on the Azure portal.

**6. Which activity should you use if you need to use the results from running a query?**

The output of a query or executable execution can be returned by a look-up activity. The outcome can be a singleton value, an array of attributes, or any transition or control flow activity like the ForEach activity. These outputs can be used in a subsequent copy data activity.

**7. Can we give a pipeline run parameters?**

Yes, in Data Factory, parameters are a first-class, top-level concept. When running the pipeline on demand or using a trigger, we can plate thickness at the pipeline level and pass arguments.

**8. Have you ever used Data Factory's Execute Notebook activity? How do I pass parameters to an activity in my notebook?**

To send code to the databricks cluster, we can use the execute notebook activity. Using the baseParameters property, we can supply parameters to a notebook activity. The default values from notebook are used if the parameters are not described or specified in the activity.

[￼](https://bit.ly/3if9dmk)

**9. What Data Factory constructs are available and useful?**

* **Parameter**: Using the @parameter construct, each activity in the pipeline can use the parameter value that was passed to it.
* **Coalesce**: To gracefully handle null values, we can use the @coalesce construct in the expressions.
* **Activity**: The @activity construct enables the consumption of an activity output in a subsequent activity.

**10. Can CI/CD or Continuous Integration and Continuous Delivery be used with ADF to push code?**

The CI/CD of the data pipelines utilizing Azure DevOps and GitHub is fully supported by Data Factory. Before publishing the final product, this allows you to refine and deliver the ETL processes incrementally. Load the data into Azure Cosmos DB, Azure SQL Azure Data Lake, Azure Data Warehouse, or whatever analytics engine the company uses and can point to from one‘s business intelligence tools once the raw data has been transformed into a form that can be consumed by businesses.

***[ Related Article:***[***GitHub CI/CD Tutorial***](https://mindmajix.com/github-ci-cd-tutorial)***]***

**Scenario-Based Azure Data Factory Interview Questions**

**11. What exactly do you mean when you refer to variables in Azure Data Factory?**

The Azure Data Factory pipeline's variables offer the capacity to store the values. They are available inside the pipeline and used for the same purposes as factors in any programming language.

Setting or modifying the values of the variables can be done using the set variable and trying to add variable activities. In a data factory, there are two different kinds of variables:

* **System variables**: these are constant variables that come from the Azure pipeline. The names of pipelines, pipeline IDs, triggers, etc. These are primarily required to obtain any network data that might be required for your use case.
* **User variable**: You must manually declare a user variable in your code.

**12. What are data flow maps?**

In Azure Data Factory, mapping flow of data are data transformations that are visually designed. Without writing any code, data engineers can create data transformation logic using data flows. The resulting data flows are carried out in weighted Apache Spark clusters by Azure Data Factory pipelines as activities. Utilizing the scheduling, control flow, and monitoring tools already available in Azure Data Factory, data flow activities could be operationalized.

Data flow mapping offers a completely visual experience without the need for coding. Scaled-out data processing is carried out using execution clusters that are managed by ADF. All of the path optimizations, data flow job execution, and code translation are handled by Azure Data Factory.

**13. What does the Azure Data Factory mean by copy activity?**

One of the most well-liked and frequently used activities in the Azure data factory is copy. It is employed in ETL, also known as lift and shift, which is the process of moving data through one data source to another. You can transform the data as you copy it. For instance, let's say you read data from a txt/csv file with 12 columns, but you only want to keep seven columns when writing it to the target data source. It can be transformed so that only the necessary number of columns are sent to the target data source.

**14. Can you provide more details about the Copy activity?**

At a high level, the copy activity completes the following actions:

Take information out of the source data store. Work with the data to perform the following tasks:

* Serialization/deserialization
* Compression/decompression
* Table mapping
* Enter data into the sink or destination data store.

**15. What various Azure Data Factory activities have you used?**

If you've used some of the key activities in your career, whether it be your job or a college project, you can share them here. Here are some of the most popular pursuits:

* To transfer data between datasets, use the Copy Data Activity.
* ForEach Activity is used to loop.
* Get Metadata Activity can give you information about any data source's metadata.
* To define and start variables within pipelines, set the variable activity.
* Lookup operation to retrieve values from a table or file.
* Wait Activity to wait for a predetermined period of time prior to or following a pipeline run.
* The existence of files inside the dataset will be verified by the validation activity.

**16. How do I plan a pipeline?**

A pipeline can be scheduled using either the window of time trigger or the scheduler trigger. The trigger utilizes a wall-clock calendar timetable that can schedule pipelines on a recurring basis or periodically. There are three trigger types that the service currently supports:

Trigger for tumbling windows: A trigger that keeps a state while operating at regular intervals.

* **Schedule trigger**: A trigger that starts a pipeline on a wall-clock timetable is known as a schedule trigger.
* **Event-based trigger**: A trigger that reacts to an event, such as a file being added to a blob, is known as an event-based trigger. The relationship between pipelines and triggers is many-to-many except for the tumbling window trigger. A single pipeline can be started by multiple triggers, or many pipelines can be started by a single trigger.

**17. When is Azure Data Factory the best option?**

Consider utilizing Data Factory:

* A data warehouse must be implemented when continuing to work with big data; you may need a cloud-based integrated platform like ADF for the same.
* Not all team members have coding experience, and some might find it easier to work with data using graphical tools.
* We would like to use a single analytics solution, such as ADF, to integrate all of the raw business data that is stored across various data sources, which may be on-premises or in the cloud.
* We prefer to manage our infrastructure lightly and use solutions for data motion and processing that are easily accessible.

**18. How are the remaining 90 dataset types in Data Factory used for data access?**

Azure Synapse Analytics, Azure SQL Database delimited text files from such an Azure storage account, or Azure Data Lake Storage Gen2 are all supported natively as the source and sink data sources by the mapping data flow feature. Parquet files from blob storage or Data Lake Storage Gen2 are also supported. Data from all other connectors should be staged using the Copy activity before being transformed using a Data Flow activity.

***[ Check out***[***Azure Analysis Services***](https://mindmajix.com/azure-analysis-services)***]***

**19. Is it possible to use the established column from mapping in ADF to determine a value for a new column?**

We can create a new column predicated on our desired logic by deriving transformations from the mapping data flow. When trying to generate a derived column, we have the option to add a new one or keep updating an existing one. In the Column textbox, type the name of the new column you're creating. To replace an existing column in the schema, use the column dropdown. To begin writing the expression for the derived column, click the Enter expression textbox. To create your logic, either input it or use the expression builder.

**20. What benefit does lookup activity in the Azure Data Factory provide?**

The Lookup activity in the ADF pipeline is frequently used for setup lookup needs, and the origin dataset is accessible. Additionally, it is used to extract the data from the source dataset and send it as the activity's output. The output of a lookup activity is typically utilized in the pipeline to make additional decisions or to present any resulting configuration. Simply put, the ADF pipeline uses lookup activity to fetch data. Your pipeline logic would determine how you would use it. Depending on the dataset or query, you may be able to retrieve just the first row or all of the rows.

**Advanced Azure Data Factory Interview Questions**

**21. Give more details about the Azure Data Factory's Get Metadata activity.**

Any data in an Azure Data Factory or Synapse pipeline can have its metadata retrieved using the Get Metadata activity. The Get Metadata activity's output can be used in conditional expressions to sample predictions or to consume the metadata in later activities. It receives a dataset as input and outputs metadata details. The following connectors are supported right now, along with the corresponding retrievable metadata. The returned metadata can only be up to 4 MB in size.

**22. How can an ADF pipeline be debugged?**

One of the most important components of any coding-related task is debugging, which is necessary to test the software for any potential bugs. It also offers the choice of debugging the pipeline without actually running it.

**23. What does "the breakpoint in the ADF pipeline" refer to?**

For instance, let's say you have a pipeline with three activities and want to focus on debugging the second action only. By setting the cut-off point at the second activity, you can achieve this. You can press the circle at the activity's top to add a breakpoint.

**24. What purpose does the ADF Service serve?**

The main function of an ADF is to coordinate data copying among numerous relational and non-relational sources of data that are hosted locally, in data centers, or the cloud. Additionally, you can use the ADF Service to transform the information that has been ingested to meet business needs. ADF Service is utilized as an ETL or ELT tool for loading data in the majority of Big Data solutions.

***[ Check out***[***Top Open Source ETL Tools***](https://mindmajix.com/open-source-etl-tools)***]***

**25. Describe the azure data factory's data source.**

The system from which the data will be used or executed is referred to as the data source. Data can be in binary, text, CSV, JSON, or any other format. It might be an appropriate database, but it could also be an image, video, or audio files.

**26. How do I copy data from multiple sheets in an Excel file?**

We must specify the name of the sheet from which we must load data when using an Excel connector inside of a data factory. When dealing with data from a single or small number of sheets, this approach is nuanced. However, if we have many sheets (say, 10+), changing the hard-coded sheet name repeatedly can become tedious. To accomplish this, we can use a data factory binary data format plug and point it at the excel file without having to specify which sheet(s) to use. The copy activity will allow us to copy the data from each and every sheet in the file.

**27. Is nested looping possible with Azure Data Factory?**

The data factory does not directly support nested looping for any looping action (for each / until). One for each and until loop activities, on the other hand, contain execute pipeline activities that may contain loop activities. In this manner, we can achieve nested looping because when we call the loop activity, it will inadvertently call another loop activity.

**28. How can I move several tables from one datacenter to another?**

An effective strategy for finishing this task would've been:

* Keep a lookup table or file that lists the tables that need to be copied along with their sources.
* After that, we can scan the list using the data retrieval activity and each loop activity.
* To copy multiple tables to the target datastore, we can employ a copy activity or a mapping data flow inside the for each loop activity.

**29. What are some of ADF's drawbacks?**

Excellent data movement and transition functionalities are offered by Azure Data Factory. There are, however, some restrictions as well.

* If we have enclosed looping activities in our pipeline, we cannot have them in the data factory and must find a solution. This covers all looping activities: Activities include If, Foreach, Switch, and Until.
* A maximum of 5000 rows can be retrieved at once by the lookup activity. Again, in order to achieve this type of organization in the pipeline, we must combine SQL with another loop activity.
* The total number of activities we can have in a single pipeline, along with all inner activities, and containers.

**30. Which assimilation runtime should we employ when using Azure Data Factory to copy data from a local SQL Server instance?**

We should have installed the self-hosted assimilation runtime on the onsite machine where the SQL Server Instance is offered to host in order to copy data from an on-premises SQL Database using Azure Data Factory.

--------------------------------------------------------------------------------------------

1. Briefly explain how you will create a Data Factory that copies data from one folder to another.

This can be done by creating a new Data Factory instance using ADF's copy data tool.

Creating Data Factory instances is quite a simple task. First, you must ensure that the user account in Azure is a member, either as a contributor, owner, or administrator of the Azure subscription. You can use a general-purpose Azure Storage account (specifically Blob storage) as both source and destination datastore. You will also need the name of your Azure Storage account.

Now, we will create a blob container, but before that, you have to ensure that the two folders involved are in the container you just created. Otherwise, create the relevant folders and upload the required files.

Next, you need to create a data factory by going to the Azure Portal. To do this follow these steps:

1. Go to the home page of Azure Data Factory and start the Copy Data tool.
2. Next, create a new connection using Azure Blob storage.
3. Select the file from your directory and also select where you want to copy the files in your blob storage container for the destination datastore.
4. And you're done!

Note: It is considered good practice to select Monitor on the complete deployment page to monitor the pipeline you created.

---------------------------------------------------------------------------------------------------

How can you create a schedule trigger in a Data Factory?

You can create a schedule trigger to schedule a pipeline to run periodically (hourly, daily, etc.).

When creating a schedule trigger, you specify a schedule (start date, recurrence, end date, etc.) for the trigger and associate it with a pipeline. Pipelines and triggers have a many-to-many relationship. This means multiple triggers can kick off a single pipeline, and a single trigger can kick off multiple pipelines.

For this, you have to follow the following steps:

1. Switch to the Edit tab in Data Factory or the Integrate tab in Azure Synapse
2. Select Trigger on the menu
3. Go to Add Triggers and select new.
4. Then select settings as per your requirements.
5. Select “Publish all” to publish the changes. The trigger doesn't start triggering the pipeline runs until you publish the changes.

Alternatively, you can even use Azure PowerShell to create, start, and monitor a schedule trigger. For this, the very first step is to create a JSON file:

{

"properties": {

"name": "MyTrigger",

"type": "ScheduleTrigger",

"typeProperties": {

"recurrence": {

"frequency": "Minute",

"interval": 15,

"startTime": "2017-12-08T00:00:00Z",

"endTime": "2017-12-08T01:00:00Z",

"timeZone": "UTC"

}

},

"pipelines": [{

"pipelineReference": {

"type": "PipelineReference",

"referenceName": "Adfv2QuickStartPipeline"

},

"parameters": {

"inputPath": "adftutorial/input",

"outputPath": "adftutorial/output"

}

}

]

}

}

The JSON file is then stored in the **C:\ADFv2QuickStartPSH\** folder.

Then, you can create a trigger by using the **Set-AzDataFactoryV2Trigger** cmdlet:

Set-AzDataFactoryV2Trigger -ResourceGroupName $ResourceGroupName -DataFactoryName $DataFactoryName -Name "MyTrigger" -DefinitionFile "C:\ADFv2QuickStartPSH\MyTrigger.json"

-----------------------------------------------------------------------------

Can you format the input and output format in Azure Data Factory?

Yes, users can specify the input and output format using data factory functions and system variables. For example:

{

"Type": "SqlSource",

"sqlReaderQuery": "$$Text.Format('SELECT \* FROM MyTable WHERE StartTime = [\\'{0:yyyyMMdd-HH}\\''](file://\\'{0:yyyyMMdd-HH}\\''), WindowStart)"

}

In the above example, the sqlReaderQuery property is assigned a value returned by the Text.Format function. The example also uses a WindowStart variable that specifies the start time of the activity in the running window.

-------------------------------------------------------------------------------

What is the Integration Runtime in Azure Data Factory?

The Azure Data Factory integration runtime is the compute infrastructure that enables the following data integration capabilities across a variety of network environments:

Data movement: The integration runtime handles data movement by moving data between the source and destination data stores while also supporting built-in connectors, format conversion, column mapping, and fast and scalable data transfer.

Data flow: You can execute a Data Flow in a managed Azure compute environment for data flow.

Dispatch activities: The integration runtime supports the native execution of SSIS packages for transformation.

Execute SSIS packages: The integration runtime natively executes SSIS packages in a managed Azure compute environment. The integration runtime also supports dispatching and monitoring transformation activities running on Azure HDInsight, Azure Machine Learning, SQL Database, and SQL Server, among other compute services.

To move and transform data, you can use one or more instances of the integration runtime. The integration runtime can run on either a public or private Azure network (on-premises, Azure Virtual Network, or Amazon Web Services virtual private cloud [VPC]).

The management hub as well as any activities, datasets, or data flows that reference them can be used to create integration runtimes in the Azure Data Factory UX.

----------------------------------------------------------------

Is it possible to create a send email workflow using the Azure Data Factory? If so, how?

Yes, it is possible to create a send email workflow using Azure Data Factory. For this, you will need both an Azure subscription and a Logic App to trigger sending an email in the pipeline.

1. Create the email workflow in your Logic App

First, you need to create an email workflow. To do this, you can use the Logic App to create a sample workflow named SendEmailFromPipeline. Next, you define a workflow trigger, say When an HTTP request is received, and add an action of Office 365 Outlook - Send an email (V2).

Now, you define an HTTP request trigger. Here is an example JSON for the Request Body JSON Schema:

{

"properties": {

"dataFactoryName": {

"type": "string"

},

"message": {

"type": "string"

},

"pipelineName": {

"type": "string"

},

"receiver": {

"type": "string"

}

},

"type": "object"

}

Finally, you can define the Send an email (V2) action. Using the Request Body JSON Schema, you can customize how you wish to format the email. Save this workflow, browse to find the workflow URL, and note it down for the next step.

2. Create a pipeline to trigger your Logic App email workflow

Now that you’ve created the Logic App workflow, you can trigger it using web activity.

First, a new pipeline needs to be created. You will then find the Web activity under the General category. Next, drag it onto the editing canvas. This will create a new activity called Web1.

Next, select the new Web1 activity and select the Settings tab. Here, add-in your workflow URL that was copied in the previous step and into the following example JSON:

{

"message": "This is a custom dynamic message from your pipeline with run ID @{pipeline().RunId}.",

"dataFactoryName" : "@{pipeline().DataFactory}",

"pipelineName" : "@{pipeline().Pipeline}",

"receiver" : "@{pipeline().parameters.receiver}"

}

You can also use dynamic expressions to generate different messages for events in your pipelines.

Next, select the background area of the pipeline designer to select the pipeline properties page. Here, you will add a new parameter called a receiver, where you will provide an email address as a default value.

Finally, you can publish your pipeline, and just as a test, trigger it manually to confirm whether the email is sent as expected.

-------------------------------------------------------------

What types of triggers does Azure Data Factory support?

ADF supports 3 types of triggers: schedule trigger, tumbling window trigger, and event-based trigger.

The scheduling trigger is a popular trigger that allows you to run a Data Pipeline on a predefined schedule. It provides numerous options and allows you to schedule triggers in minute(s), hour(s), day(s), week(s), or month intervals(s). Furthermore, you can specify the start and end dates to make the trigger run only for that period. Additionally, the Schedule Trigger can be set to run on future calendar days and hours.

The Tumbling Window Trigger runs Data Pipelines at a predetermined time interval or at a specific time slice. When copying or migrating historical data, it is substantially more advantageous than Schedule Triggers.

The Event-based Azure Data Factory Trigger runs Data Pipelines in response to blob-related events, such as generating or deleting a blob file present in an Azure Blob Storage. With the Event-based Triggers, you can schedule the Data Pipelines to execute in response to an event from Azure Blob Storage.

In addition, Event-based Triggers are not only compatible with blobs but also with ADLs. Similar to Schedule Triggers, Event Triggers can also work on many-to-many relationships. A single Event Trigger can run several Pipelines, and multiple Event Triggers can run a single Pipeline.

-------------------------------------------------------------------------------------

What is the difference between the Mapping Data Flow and Wrangling Data Flow transformation activities in Data Factory?

Mapping Data Flows

Data can be visually transformed using Mapping Data Flows without writing any code. You can concentrate on the transformations and logic while Azure Data Factory takes care of the heavy lifting. It converts your logic and transformations into code that runs on scaled-out Azure Databricks clusters for maximum performance.

Wrangling Data Flows

Using the Power Query Online mashup editor, Wrangling Data Flows are used to explore and prepare datasets visually. You can concentrate on the modeling and logic while Azure Data Factory takes care of the heavy lifting. It converts the underlying M code to Spark code that runs in a managed environment for maximum performance.

Key differences between Mapping and Wrangling Data Flows

Both Mapping and Wrangling Data Flows have some overlapping features. However, there are a few key differences:

Transformations:

With Mapping and Wrangling, you can do many of the same transformations, including filtering rows, merging/joining datasets, sorting, grouping, and adding and renaming columns. However, Mapping Data Flows can also handle inserts, delete, updates and upserts.

Schema Drift:

Mapping Data Flows can be used to handle schema drift if your source frequently changes. For example, if some columns are removed or added, the destination can be updated automatically to exclude or include those new columns. However, you have to make these changes manually in Wrangling Data Flows.

File and Table Handling:

File management, such as transferring files after they've been read, is built into Mapping Data Flows. During execution, you can also select to recreate sink tables. This eliminates the need to construct and execute T-SQL scripts before data loading.

------------------------------------------------------------------------------

How can you copy data from an Azure Blob Storage (text file) to an Azure SQL Database Table?

First of all, we create a data factory using simple C# code. Then we have to create Azure storage linked service by adding the following C# code to the main method:

// Create an Azure Storage linked service

Console.WriteLine("Creating linked service " + storageLinkedServiceName + "...");

LinkedServiceResource storageLinkedService = new LinkedServiceResource(

new AzureStorageLinkedService

{

ConnectionString = new SecureString(

"DefaultEndpointsProtocol=https;AccountName=" + storageAccount +

";AccountKey=" + storageKey

)

}

);

client.LinkedServices.CreateOrUpdate(

resourceGroup, dataFactoryName, storageLinkedServiceName, storageLinkedService

);

Console.WriteLine(

SafeJsonConvert.SerializeObject(storageLinkedService, client.SerializationSettings)

);

This is followed by creating an Azure SQL Database linked service by adding this code to the main method:

// Create an Azure SQL Database linked service

Console.WriteLine("Creating linked service " + sqlDbLinkedServiceName + "...");

LinkedServiceResource sqlDbLinkedService = new LinkedServiceResource(

new AzureSqlDatabaseLinkedService

{

ConnectionString = new SecureString(azureSqlConnString)

}

);

client.LinkedServices.CreateOrUpdate(

resourceGroup, dataFactoryName, sqlDbLinkedServiceName, sqlDbLinkedService

);

Console.WriteLine(

SafeJsonConvert.SerializeObject(sqlDbLinkedService, client.SerializationSettings)

);

You will now create a dataset for source Azure Blob as you define a dataset that represents the source data in Azure Blob, followed by creating a dataset for sink Azure SQL Database.

From here, things get relatively simple as we only have to create a pipeline with a copy activity, which takes in the Blob dataset as the source and the SQL dataset as the sink. Then we only have to monitor the pipeline and activity runs to ensure successful completion!

---------------------------------------------------

When copying data from or to an Azure SQL Database using Data Factory, what is the Firewall Option that one should enable to allow the Data Factory to access that database?

In this case, the “Allow Azure services and resources to access this server” option on the “Firewall and virtual networks” page should be selected.

Doing this permits Azure services to access SQL Database. It ensures that you allow access to Azure services in your server so that the Data Factory service can write data to SQL Database.

-------------------------------------------------------------------------

Is it possible to programmatically monitor Azure Data Factory via SDK?

Data Factory only stores pipeline run data for 45 days. When you query programmatically for data about Data Factory pipeline runs, there are no maximum dates for the optional LastUpdatedAfter and LastUpdatedBefore parameters.

A pipeline run has different statuses during its lifecycle, such as queued, InProgress, succeeded, failed, canceling, and canceled.

Now, we can monitor a pipeline run using .NET, Python, REST API, and the Azure Az Powershell.

Let’s take the example code of Python:

# Monitor the pipeline run

time.sleep(30)

pipeline\_run = adf\_client.pipeline\_runs.get(

rg\_name, df\_name, run\_response.run\_id)

print("\n\tPipeline run status: {}".format(pipeline\_run.status))

filter\_params = RunFilterParameters(

last\_updated\_after=datetime.now() - timedelta(1), last\_updated\_before=datetime.now() + timedelta(1))

query\_response = adf\_client.activity\_runs.query\_by\_pipeline\_run(

rg\_name, df\_name, pipeline\_run.run\_id, filter\_params)

print\_activity\_run\_details(query\_response.value[0])

----------------------------------------------------------

Imagine a scenario where you have to create an ETL process. How will you do it using Azure Data Factory?

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

For Creating an ETL process, one should follow these steps:

First, create a Linked Service (which is an SQL Server Database) for the source data store.

Assume that we have a car dataset.

Next, create a Linked Service (which is an Azure Data Lake Store) for the destination datastore.

Create a dataset for Data Saving.

Create the pipeline and add copy activity.

Schedule the pipeline by adding a trigger.

-----------------------------------------------------------------------------------

What are the two levels of security in ADLS Gen2?

Built on Azure Blob Storage, Azure Data Lake Storage Gen2 is a suite of capabilities targeted for big data analytics. The functionalities of Azure Data Lake Storage Gen1 and Azure Blob Storage are combined in Data Lake Storage Gen2.

There are two levels of security in ALDS Gen2:

Access Control by Role (RBAC)

Built-in Azure roles such as reader, contributor, owner, and custom roles are included in RBAC. RBAC is typically assigned for two reasons:

One is to identify who will be in charge of the service (i.e., update settings and properties for the storage account).

Another purpose is to allow users to use built-in data explorer tools that require reader access.

Access Control List (ACL)

ACLs define which data items a user can read, write, or execute (execute is required to browse the directory structure). ACLs are POSIX-compliant. Therefore people with Unix or Linux experience will be familiar with them.

-------------------------------------------------------------------------

Can you tell what the following JSON definition is trying to do?

​​{

"name": "AzureBlobStorageLinkedService",

"properties": {

"type": "AzureBlobStorage",

"typeProperties": {

"connectionString": "DefaultEndpointsProtocol=https;AccountName=;AccountKey="

},

"connectVia": {

"referenceName": "",

"type": "IntegrationRuntimeReference"

}

}

}

The code actually defines a linked service in JSON format that can allow you to establish a connection with your data stores.

The code has 4 properties: name, type, typeProperties, and connectVia.

As seen in the JSON, the name property holds the name of the linked service.

type contains the type of the linked service. For example, AzureBlobStorage (data store) or AzureBatch (compute).

Properties in the typeProperties depend on the type of activity, thus the typeProperties will be different for each data store or compute activity. The example above is only being utilized as storage for a connection string to the AzureBlobStorage.

connectVia stores the Integration Runtime to be used to connect to the data store. You can use Azure Integration Runtime or Self-hosted Integration Runtime (if your data store is located in a private network). If not specified, it uses the default Azure Integration Runtime.

---------------------------------------------------------------------------------------

What is the following C# code snippet doing?

DatasetResource blobDataset = new DatasetResource(

new AzureBlobDataset

{

LinkedServiceName = new LinkedServiceReference

{

ReferenceName = storageLinkedServiceName

},

FolderPath = new Expression { Value = "@{dataset().path}" },

Parameters = new Dictionary

{

{ "path", new ParameterSpecification { Type = ParameterType.String } }

}

}

);

client.Datasets.CreateOrUpdate(

resourceGroup, dataFactoryName, blobDatasetName, blobDataset);

Console.WriteLine(

SafeJsonConvert.SerializeObject(blobDataset, client.SerializationSettings));

This is the code that you can add to the main method to create an Azure blob dataset.

You create a dataset that represents the data that needs to be copied from a source to a sink. However, this example is incomplete and will first require the creation of a linked service to the blobDataset:

LinkedServiceResource storageLinkedService = new LinkedServiceResource(

new AzureStorageLinkedService

{

ConnectionString = new SecureString(

"DefaultEndpointsProtocol=https;AccountName=" + storageAccount +

";AccountKey=" + storageKey)

}

);

client.LinkedServices.CreateOrUpdate(

resourceGroup, dataFactoryName, storageLinkedServiceName, storageLinkedService);

Console.WriteLine(SafeJsonConvert.SerializeObject(

storageLinkedService, client.SerializationSettings));

After creating a linked service, the code in question is run which relates it to the Azure Storage we have just created.

The dataset has a parameter whose value is set in a dataset-consuming operation. The argument is then used to create the folderPath that points to the location of the data.

Note: Commonly asked Azure Data Factory interview question

--------------------------------------

Q2. How Azure Data Factory Makes the Process of Creating a Data Pipeline Easy?

ADF makes the process of creating a data pipeline easy by providing built-in connectors for data ingestion and orchestration, giving various activity options to perform operations such as copying data, for-each loop, look-up, etc., validating, publishing and monitoring pipelines, continuous integration, and continuous deployment support to the pipelines.

Q3. What are the Different Types of Activities Supported by Azure Data Factory?

Below are the different types of activities supported by ADF:

1. Data Movement Activities: Activities used to move data from one data source to another in a data pipeline are known as Data movement activities. For example, copy activity can be used to copy data from ADLS to Azure SQL.

2. Data Transformation Activities: Activities used to perform data transformation in a data pipeline are known as Data transformation activities. Data Flow Activity, Azure Functions Activity, Databricks Notebook Activity, etc., are examples of data transformation activities.

3. Control Activities: Activities used to build conditional, sequential, or iterative conditional logic in a data pipeline are known as control activities. Lookup Activity, Until Activity, For-Each Activity, etc., are examples of control activities.

Azure Data Factory

Source: docs.microsoft.com

Q4. Solve the Project Scenario based on Question 1.

Your data team is building an ETL pipeline for a client. You want to generate output files from Azure Data Factory which are optimized for read-heavy analytical workloads and support the columnar format. What should be the file format of output files?

The generated output files should have Parquet format as Parquet stores data in columns and are optimized for read-heavy analytical workloads.

Q5. What are Annotations in Azure Data Factory?

Annotations are additional informative tags that help in filtering and searching data factory resources such as datasets, pipelines, linked services, etc. For example, if you are working as a team lead for a large data processing project for a client ABC that uses ADF containing 10 pipelines. To avoid confusion in the data processing sequence, we can label each pipeline with its primary purpose: ingest, transform, or load using annotations. When we are monitoring pipelines, these annotations must be available to perform searching, grouping, and filtering.

Q6. Solve the Project Scenario based on Question 2.

A data science company handles data processing for different clients. Your team is building an ADF pipeline to move user logs generated based on users’ activities on an e-commerce platform from an ADLS container to a database inside Azure Synapse dedicated SQL pool. The user logs are stored in container users in the following folder structure./user/{YYYY}/{MM}/{DD}/{HH}/{mm}

The earliest folder is /user/2021/01/02/00/00. The latest folder is /user/2021/01/17/01/45.

How would you configure the pipeline to trigger so that existing data must be loaded every 30 minutes, and up to two minutes delay in data arrival must be included in the time at which the data should have arrived?We can configure the pipeline to trigger using a tumbling window trigger with Recurrence: 30 minutes, Start time: 2021-01-01T00:00, and Delay: 2 minutes to achieve the above scenario.

Q7. How Can Users Secure Their Data Store Credentials in ADF?

Users can secure their data store credentials in ADF by storing them in Azure Key Vault or encrypting them with certificates. Azure Key Vault is an Azure service used to securely store API keys, data store credentials, passwords, etc., to prevent unauthorized access. Developers can easily import or create keys, authorize users to access the key vault, and configure and manage the keys using Azure Key Vault.

Q8. State the Difference Between Pipeline Parameters and Variables in ADF.

Pipeline parameters are created using the “Parameters” tab in the pipeline and cannot be modified while a pipeline is running.

Pipeline Parameters

Source: learn.microsoft.com

Pipeline variables can be modified and set using Set variable activity during a pipeline run.

Variables in ADF

Source: learn.microsoft.com

Q9. Name Some Data Stores and File Formats Supported by Azure Data Factory.

Azure Data Factory supports various data stores such as Azure SQL, Azure Storage, Azure Databricks, HBase, Hive, Impala, MariaDB, Oracle, Cassandra, Amazon S3, MongoDB Atlas, etc. ADF supports various file formats such as Parquet, Avro, JSON, Delta, Excel, XML, Delimited text format, etc.

Q10. Which Activity of Azure Data Factory can be Used to Copy Data From Azure Blob Storage to Azure SQL?

Copy activity can be used in ADF to copy data from Azure Blob Storage to Azure SQL. Copy activity is used to copy data from between different data sources. Copy Activity reads data from the source store, performs column mapping, data compression/decompression based on data type and input and output dataset format, and writes the data into the destination data store.

Azure SQL

Source: learn.microsoft.com

Conclusion

Azure Data Factory (ADF) is a cloud-based data ingestion and ETL (Extract, Transform, Load) Azure service. The data-driven workflow in ADF orchestrates and automates data movement and data transformation. ADF helps developers to build complex ETL processes and scheduled event-driven workflows to process data which later can be used by various reporting tools for storytelling purposes. Below are some key points from the above article:

1. We have seen how ADF makes the process of creating a data pipeline easy.

2. We learned about approaches by which users can secure their data store credentials in ADF.

3. We have seen the differences between pipeline parameters and variables in ADF.

4. We got an understanding of how we can copy data from Azure Blob Storage to Azure SQL using ADF.

5. Apart from this, we also saw some scenario-based questions on ADF.