

First create resources

* A **Resource Group** is a **logical container** that holds related **resources** for an Azure solution.
* To create a resource group

1. Search for resource group
2. Click on create
3. Give the name, select the region
4. Review + create and create

* A Storage account in Azure is a fundamental container that provides unique namespace for your storage data.

(A **namespace** is basically a **naming system** that helps you **organize and manage resources or objects** without name conflicts.)

* To create a storage account

1. Search for storage account
2. Select the resource group
3. Give the storage account name and select the region
4. Select the primary service as Azure blob storage or Azure Data Lake Storge Gen 2
5. Creating a storage account only creates a blob storage and does not create a data lake. (A blob Storage is Microsoft Azure’s object storage solution for the cloud. Blob stands for binary large object; it can store any type of data)
6. The main difference between blob storage and data lake is that blob is a general-purpose object storage while data lake is built on blob with hierarchical namespace and added features for data analytics.
7. To create a data lake, enable the hierarchical namespace
8. Set access tier to hot. Review + create and create

* **Azure Data Factory (ADF)** is a **cloud-based data integration service** that lets you **create, schedule, and manage data pipelines** to **move and transform data** from various sources to destinations.
* To create data factory

1. Search for data factories
2. Select the resource group, Give the name
3. Review + create -> create

* A container is like a folder inside a storage account that holds your blobs(files).
* To Create containers

1. Go to data lake
2. Under data storage, click on containers
3. Click on + Container on the top
4. Give the name and click on create

* A **pipeline** in Azure Data Factory is a **logical group of activities** that **perform a task together** — like **moving**, **transforming**, or **loading data** from one place to another.
* To create a pipeline

1. Go to data factory
2. under factory resources click the 3 dots beside pipeline and new pipeline and give it a name
3. now click on move and transform and drag and drop copy data

* Now we need to create a linked service, this is basically a connection between data lake to source and data lake to destination (http)
* To create a linked service

1. Go to manage services, click on linked services-> new
2. As this is a http, search and select http and click on continue
3. Give a name, base URL and authentication type to anonymous
4. Click on test connection to test your connection
5. Click on create

* Now create a data lake linked service
* To create it

1. Go to manage services, click on linked services-> new
2. As this is a data lake select azure data lake storage gen2
3. Give a name and also select the storage account name
4. Test the connection and click on create

* Now go to pipeline and set the source
* To do it

1. Beside source click on + New
2. Select http as data store
3. Select the file format (csv)
4. Give it a name and select the linked service
5. Give the base URL and click on OK.

* Now we have to set the sink
* For that

1. Beside source click on + New
2. Select azure data lake storage gen 2
3. Select the file format (csv)
4. Give the name and linked service as data storage
5. Give the file path as the name of the container (bronze) and give any name.
6. Click on OK

* Select mapping if u want any schemas and click on debug to check the pipeline status
* To save pipeline click on publish all and then publish
* We have to create dynamic pipelines for that we’ll be using loop.
* The things that’ll change for every file is that:

1. Relative URL
2. Folder in which the file should be stored
3. File itself

* Instead of hardcoding we’ll change the parameters in every iteration
* For that we need to create a for each activity
* To create a dynamic pipeline initial steps are same but instead of giving the base URL.

1. Click on Advanced -> Open this dataset
2. A new window opens. Under the base URL, click on the add dynamic content option
3. Click on + under the parameters option
4. Give it a name and click on save.
5. You’ll be asked to give the expression, click on the parameter name below and click on OK.

* Now set the sink

1. Initial steps same
2. Instead of giving the file path, click on advanced
3. Our container is same, Add parameter to the directory
4. Select the parameter and click OK
5. Do same for file name as well

* Now we have to give 3 parameters one for source and 2 for sink. To pass them we need to create a for each activity
* From activities click on Iteration & conditionals and select ForEach
* Give it a name
* Upload the json that contains the sets of parameters in a separate container.
* A look up activity in Azure is used to retrieve data from an external source.

1. Add lookup activity from activities and in settings click + symbol beside source dataset
2. Select azure data lake storage Gen2 and select file format as json
3. Give it a name and select the file from parameters container
4. Uncheck the First row only box to run through all entities
5. Debug the lookup activity by deactivating the other activities
6. To refer to activities select success from nodes and connect it to for each

* Now we want to use the output of Lookup activity in the for each activity

1. Go to the settings of ForEach activity
2. Click on Add Dynamic content -> Select the LookUp activity
3. As we want the value of the output type .value beside output.

(Because the output passes the array and we want only the value)

1. Click on OK

* Now we have to embed the copy activity in For each and pass the parameter values into it.
* For that cut the copy activity

1. Go to for each activity and click on activities tab
2. Click on the pencil symbol
3. Paste the copy activity
4. Go to the source select the foe each item and beside @item add .relative url variable
5. Do the same thing for source (folder and filename)
6. Debug the for each activity

* Challenge: Gave the parameter and key in the json file the same name. In that case the ADF gets confused because it's like you're asking it to **assign a value to itself** — it ends up as a **circular reference**, which can break the activity. So just change the value name.
* Go to home and create a databricks resource and click on create

1. Select the resource group, give it a name
2. In data bricks we perform all the transformation steps on clusters, these will be handled by databricks and these cluster configurations are contained in managed resource group
3. After that just launch the databricks, click on compute -> create compute
4. Give it a name and select single node as this is a small project
5. Select the runtime version, Node type and termination time
6. Click on create compute

* Now we have to linked databricks and datalake to access data
* For that we create a service level application which will have access to datalake and will be used by databricks
* For that

1. Search for Microsoft Entra ID
2. Under manage tab click on app registrations -> New registration
3. Give a name and click on register.
4. Copy the application ID.

(App\_ID = ”b5daa302-647c-45c5-8ff8-c3ef65a1460e”)

(Object\_ID = “fdb82a5c-6767-4025-b2b2-d24d0a459de0”)

* Now we need to create a secret to pull the data into Data bricks.

1. Click on certificates & secrets
2. Click on New Client Secret
3. Give it a name and click on Add
4. Copy the value

(Value = “6uJ8Q~eBzF4RyOBV\_-TfAsMalmMphGHH8uVKkbx0”)

(Secret\_ID = “9edf7492-2c0f-40f7-92a7-82541b558cf5”)

(Tenant\_ID=”a51559b3-6de4-4efc-9b92-86974e465540”)

* We have created an application, now we have assign a role to the application so that it can access the datalake.
* To do that

1. Go to Access control (IAM) -> Add -> Add role assignment
2. The role we want to assign is storage blob Data contributor which gives both read and write options.
3. Select it and click on next
4. Click on select members, search for our application
5. Click on Review + Assign

* Now go to data bricks

1. Click on workspace -> create -> Folder
2. Give it a name and create
3. Create a notebook in the folder
4. Click on connect and select the cluster to turn it on

* Now we write the spark script for transformation

1. First write the configuration script and give all the storage account name, app\_id and all.

Challenger: couldn’t get the dataset because instead of tenant\_id I copied the object\_id so the configurations failed. (**the "Tenant ID"** refers to the unique **identifier (GUID)** of your **Azure Active Directory (Azure AD)** instance.)

* df.withColumn(‘columnname’, month(col(Date)))
  + - extracts month from a time stamp

(if we give existing column name it updates or else if we give a new one it creates one)

* Now we push this data to silver layer and we do this in a parquet format

(**Parquet** is a **file format** — specifically designed for **big data** processing.

It is:

* + - **Columnar** → stores data **column by column** (not row by row like CSV)
    - **Efficient** → saves a lot of storage space
    - **Fast** → speeds up reading and processing
    - **Compressed** → automatically stores data in a compact way
    - **Structured** → keeps schema (field names + types) inside the file itself)
* **df\_cal.write.format(‘parquet’)**

**.mode(‘append’) – To mention the updation mode, there are 4 modes**

* + - append() – To add to the existing data like a union
    - overwrite() – To replace the existing data
    - error() – Fails if the output path already exists
    - ignore() – Skips writing if data already exists

.option() – In this we specify the path where we want to store the data

* Transformations

1. For df\_calendar we created 2 separate columns for month and year

[code - df\_cal = df\_cal.withColumn('Month',month(col('Date'))).withColumn('Year',year(col('Date'))]

1. For df\_cus we concatenated Prefix, firstname and lastname and created a column called full name.

[code - df\_cus.withColumn("fullName",concat(col('Prefix'),lit(' '),col('FirstName'),lit(' '),col('LastName'))).display()]

1. For df\_pro we split the ProductSKU and ProductName

[code - df\_pro = df\_pro.withColumn('ProductSKU',split(col('ProductSKU'),'-')[0])\

    .withColumn('ProductName',split(col('ProductName'),' ')[0])]

1. For df\_sales we created timestamp and performed multiplication

[code - df\_sales = df\_sales.withColumn('StockDate',to\_timestamp('StockDate'))]

[code - df\_sales = df\_sales.withColumn('multiply',col('OrderLineItem')\*col('OrderQuantity'))]

* Now we to create a Azure synapse analytics resource
* For that

1. Click in create resource and search for azure synapse analytics
2. Click on create, select the resource group
3. Give it a name
4. Also select the data lake account
5. Give a sql server password
6. Review + create -> create

* **Azure Synapse Analytics** is a **powerful data analytics platform** from Microsoft that combines:
  + - **Data Warehousing**
    - **Big Data Analytics**
    - **Data Integration (ETL)**
    - **SQL + Spark processing**
    - **Power BI integration**
    - **Real-time analytics**
* Now create a storage blob data contributor role to azure synapse to access the data
* Go to synapse analytics and develop a SQL script and we need a SQL database
* There are 2 SQl databases available in azure dedicated and serverless
* Serverless SQL in Azure lets you query data on-demand and pay per use, while Dedicated SQL requires pre-allocated resources and is better for consistent, high-performance workloads.
* Now write the SQL scripts First create gold view

(Query = CREATE SCHEMA GOLD)

* Now create views to access the data by using openrowset()

(OPENROWSET enables you to:Query data from **external files** (like Excel, CSV), Access remote **databases,** Import or export data **on the fly)**

**[ Query = CREATE VIEW gold.subcategories**

**AS**

**SELECT \***

**FROM**

**OPENROWSET**

**(**

**BULK 'https://awstoragedatalakepps.dfs.core.windows.net/silver/AdventureWorks\_SubCategories/',**

**FORMAT = 'PARQUET'**

**)as QUER1]**

* **To create external tables there are 3 steps**

1. **We have to create credentials**
2. **External data source**
3. **External file format**

* **Use [CREATE MASTER KEY ENCRYPTION BY PASSWORD = 'password'] to create credentials. It is just database master key**
* **A database-scoped credential is a credential that is contained within a database and is used to authenticate to external resources**

**[Query - CREATE DATABASE SCOPED CREDENTIAL cred\_pps**

**WITH**

**IDENTITY = 'Managed Identity']**

* **Now we point external database to our silver container**

**[Query - CREATE EXTERNAL DATA SOURCE source\_silver**

**WITH**

**(**

**LOCATION = 'https://awstoragedatalakepps.dfs.core.windows.net/silver',**

**CREDENTIAL = cred\_pps**

**)]**

* **Create one or gold too**

**[Query - CREATE EXTERNAL DATA SOURCE source\_gold**

**WITH**

**(**

**LOCATION = 'https://awstoragedatalakepps.dfs.core.windows.net/gold',**

**CREDENTIAL = cred\_pps**

**)]**

* **Create external file format**

**[Query - CREATE EXTERNAL FILE FORMAT format\_parquet**

**WITH**

**(**

**FORMAT\_TYPE = PARQUET,**

**DATA\_COMPRESSION = 'org.apache.hadoop.io.compress.SnappyCodec'**

**)]**

* **We have created a view on top of silver layer and now we create an external table which will push the data in the silver layer to the gold and layer and create a external table on top of it. We will be doing this using CETAS (create external table as select)**
* **In case of view we store the query but in external table we actually store the data.**
* **Install Power BI**
* **To establish a connection between synapse and Power BI we need a SQL Server endpoint**
* **To do that**

1. **First copy the name of the SQL Server endpoint**
2. **Click on get data -> more**
3. **Select Azure synapse analytics SQL and click on connect**
4. **Paste the name of SQL server endpoint in the server box. Click on OK.**
5. **Give your database credentials.**
6. **Prepare dashboards according to your need.**