Microsoft Azure Tutorial

Create an Account on Microsoft Azure

- Visit the Azure Portal and sign up for an account if you don't already have one. Follow the sign-up process, which includes verifying your identity and setting up a subscription.

Setting Up Data Pipelines and Data Lake in Azure with Static and Dynamic Pipelines

1. Create a Resource Group

- After logging into Azure, go to the **Resource Groups** section.
- Click on **Create Resource Group**.
- Provide a **Name** and choose a **Region** (where your resources will be hosted).
- Click **Review + Create** and then **Create**. This resource group will house all related resources, such as storage accounts, Data Factory, and data lake.

2. Create a Storage Account

- Go to **Storage Accounts** on the Azure Portal.
- Click **Create** and choose your previously created **Resource Group**.
- Specify a unique **Storage Account Name** and select your **Region**.
- Choose **Standard** for performance and **Hot** for access tier.
- By default, this storage account will support **Blob** storage.

How to Create a Data Lake Instead of Blob Storage

- In the **Advanced** tab of the storage account creation:
  - Enable **Hierarchical Namespace** (a necessary setting to create a Data Lake).
- Click **Review + Create** and then **Create**.

Difference Between Blob Storage and Data Lake

- **Blob Storage**: Optimized for storing massive amounts of unstructured data. It supports flat storage but does not natively support directory structures.
- **Data Lake Storage**: Built on top of Blob storage, but with the **Hierarchical Namespace** feature, allowing directory and file-level organization. It supports advanced data organization, subfolder structures, and efficient file management, making it ideal for big data processing.

3. Create Azure Data Factory

- Navigate to **Data Factory** on the Azure portal.
- Click **Create** and choose the **Resource Group** you previously created.
- Provide a **Name**, **Region**, and **Git Configuration** if applicable.
- Click **Review + Create** and then **Create** to set up the Data Factory, which will manage your data pipelines.

4. Set Up Containers in Data Lake

- Go to **Storage Accounts** and select the **Data Lake** storage you created.
- Open the **Containers** section.
- Create three new containers:
    - **Bronze** for raw data (unprocessed).
    - **Silver** for transformed data.
    - **Gold** for serving layer (final, processed data).

5. Return to Azure Data Factory and Set Up Static Pipeline

- Go to **Home**, select your **Resource Group**, and then choose **Data Factory**.
- Open the **Author** tab and click **Create Pipeline** to make a static pipeline.

Pipeline Requirements

- **Source**: The source of your data (e.g., HTTP from GitHub).
- **Destination**: The destination for your data (e.g., Data Lake containers).

6. Create a New Pipeline with Linked Services

1. In **Author**, select **New Pipeline**.
2. Go to the **Manage** tab and create linked services for the source and destination.
    - **Source (HTTP)**: Connect to GitHub by setting the **Base URL** to `https://raw.githubusercontent.com`.
    - **Destination (Data Lake)**: Connect to the previously created Data Lake.

7. Configure Source and Sink

1. Go back to the **Author** tab.
2. Add a **Source**:
    - Choose **HTTP** as the source type.
    - Select the previously created **HTTP linked service** and specify the **relative path** to your data.
3. Add a **Sink**:
    - Choose **Data Lake** as the sink type.

    - Select the Data Lake linked service and specify the **container** as **Bronze** (for raw data).
4. **Debug** the pipeline to test your setup.
5. After confirming the data is being copied as expected, **Publish All** to save changes.

---

8. Create a Dynamic Pipeline with Parameters

In this pipeline, you'll set up parameters to dynamically change the source and destination, allowing for flexibility and automation.

1. **Pipeline Setup**:
    - Create a new pipeline and set the **base URL** in the source link service to the same GitHub URL (`https://raw.githubusercontent.com`).
    - Go to **Advanced Parameters** and add **dynamic content** in the **Relative URL**.
2. **Parameter Creation**:
    - Create a parameter to dynamically update the relative URL.
    - In the **Source Dataset** (e.g., `ds_source_dynamic`), add parameters for the dynamic portion of the URL.
    - In the **Sink Dataset** (e.g., `ds_sink_dynamic`), add parameters for the destination subfolder and file name.

9. Pass Values to Parameters Using ForEach

1. Search for **ForEach** activity under **Iteration & Conditionals**.
2. Set **Sequential** to ensure data is processed one step at a time.
3. To pass parameter values, configure them as a dictionary in **VSCode** or another editor.
4. Upload the parameters file (e.g., a JSON file) to the **Data Lake** storage under a new container if necessary.

---

10. Pull Data Using Lookup Activity

1. In the **Dynamic Pipeline**, add a **Lookup** activity.
    - Uncheck **First Row Only** to retrieve all rows from the lookup table.
2. **Debug** this activity to verify it retrieves data as expected.
3. Use **Activity Output** and **LookupGit.value** (specifying `.value` to access the array stored in the lookup output).

---

11. Set Up Dynamic Copy Activity in ForEach Loop

1. Copy the **DynamicCopy** activity.
2. Edit the **ForEach** activity, open the **Activities** tab, and paste **DynamicCopy** into it.
3. Configure **Source** and **Sink** to use the dynamic parameters:
    - In **Source**: Set the iterator to retrieve the **source key** from **ForEachGit**.
    - In **Sink**: Set the iterator to retrieve the **sink key** from **ForEachGit**.

At this point, you have successfully set up data ingestion in your **Bronze (Raw)** container using both static and dynamic pipelines. The Data Lake now contains raw data organized by the defined containers and pipelines for further data transformations.

Creating an Azure Databricks Resource for Data Transformation

1. Create an Azure Databricks Resource

- Log into your Azure portal.
- Search for **Databricks** in the search bar.
- Select **Azure Databricks** and click **Create**.
- Fill in the necessary details, including the subscription, resource group, workspace name, and location.
- Choose the pricing tier that best suits your needs, and click **Review + Create** to finalize.

2. Launch Azure Databricks

- After the resource is created, navigate to the Azure Databricks workspace.
- Click **Launch Workspace** to open Databricks in a new tab.

3. Create a Basic Cluster in Databricks

- In the Databricks workspace, go to the **Compute** section.
- Click **Create Cluster**.
- Configure the cluster with the following settings:
    - **Policy**: Set to **Unrestricted**.
    - **Cluster Mode**: Select **Single Node**.
    - **Access Mode**: Choose **No Isolation Shared**.
    - **Databricks Runtime Version**: Set to **14.3 LTS** for optimal performance.
    - **Node Type**: Use **General-Purpose Standard_DS3_v2**.
    - **Auto Termination**: Specify a termination time in minutes to shut down the cluster automatically if idle, to save costs.
- Click **Create Cluster** to start the cluster.

4. Set Up Access Between Azure Databricks and Azure Data Lake

- To access raw data stored in Azure Data Lake from Databricks, you need to create a Service Principal with the required permissions.

Steps to Create a Service Principal:

1. In the Azure portal, search for **Microsoft Entra ID**.
2. Navigate to **Manage > App registrations**, and click **New registration**.
3. Fill in the application details (name, redirect URL) and click **Register**.
4. Save the **Application (Client) ID** and **Directory (Tenant) ID**:
    - **Application (Client) ID**: `Your_application_id`
    - **Directory (Tenant) ID**: `Your_directory_id`

Steps to Create a Secret for the Application:

1. In your app registration, go to **Manage > Certificates & Secrets**.
2. Click **New client secret** to generate a secret for the Service Principal.
3. Save the **Secret Value** (it's only displayed once):
    - **Secret Value**: `Your_Secret_value`
    - **Secret ID**: `Your_secret_id`

5. Assign Storage Blob Data Contributor Role to the Service Principal

- Navigate to your **Storage Account** where the raw data is stored.
- Go to **Access Control (IAM)** and click **Add Role Assignment**.
- Assign the **Storage Blob Data Contributor** role to the Service Principal.
- Under **Members**, select the application you registered.

6. Set Up a Notebook in Databricks

- Open the Databricks workspace.
- Create a **Workspace Folder** and add a **Notebook**.
- Connect this notebook to the cluster you created.

7. Configure Access to Azure Data Lake in the Notebook

- Write the code in the notebook to configure access using the Service Principal credentials:

python
Copy code

```
# Store the service credential in a secret scope (Databricks secret
management)
```

```
service_credential = dbutils.secrets.get(scope="<secret-scope>",
key="<service-credential-key>")

# Configure Spark to access Azure Data Lake with OAuth using Service
Principal
spark.conf.set("fs.azure.account.auth.type.<storage-account>.dfs.core.
windows.net", "OAuth")
spark.conf.set("fs.azure.account.oauth.provider.type.<storage-account>
.dfs.core.windows.net",
"org.apache.hadoop.fs.azurebfs.oauth2.ClientCredsTokenProvider")
spark.conf.set("fs.azure.account.oauth2.client.id.<storage-account>.df
s.core.windows.net", "<application-id>")
spark.conf.set("fs.azure.account.oauth2.client.secret.<storage-account
>.dfs.core.windows.net", service_credential)
spark.conf.set("fs.azure.account.oauth2.client.endpoint.<storage-accou
nt>.dfs.core.windows.net",
"https://login.microsoftonline.com/<directory-id>/oauth2/token")
```

- Replace `<storage-account>` with the name of your Azure Data Lake Storage account.
- Replace `<application-id>` with your Application (Client) ID.
- Replace `<directory-id>` with your Directory (Tenant) ID.
- Replace `<secret-scope>` and `<service-credential-key>` with the appropriate Databricks secret scope and key for storing the service credential.

This setup will allow your Databricks notebook to authenticate securely to Azure Data Lake using the Service Principal for data transformation tasks.

Using Transformed Data in the Serving Layer with Azure Synapse and Power BI

Step 1: Create a New Synapse Resource

1. **Navigate to Azure Synapse Analytics**:
    - Go to the Azure portal and select *Azure Synapse Analytics* to create a new Synapse workspace.
2. **Select the Resource Group**:
    - Choose the resource group you've been working with, or create a new one.

3. **Use a Separate Storage Account (Recommended)**:
    - It’s advisable to create or use a different storage account for Synapse Analytics to manage security and data organization effectively.
4. **Create SQL Credentials**:
    - Define the SQL credentials required for Synapse to connect to data sources.
5. **Disable Managed Virtual Network**:
    - Disabling this can simplify access to certain external storage resources.
6. **Use an External Storage Account for Synapse**:
    - Synapse will create a default storage account, but we'll use the external data lake account created earlier.
7. **Open the Synapse Workspace**:
    - Once Synapse is created, open it to access the *Azure Synapse Analytics* workspace.

Step 2: Explore Azure Synapse Analytics Capabilities

Azure Synapse offers a unified platform with integrations:

- **Azure Data Factory**: Used for creating and managing data pipelines.
- **Databricks (PySpark)**: To perform large-scale data processing.
- **Warehousing and Analytics**: Synapse allows for both dedicated and serverless SQL pools to manage and query data.
- **Integrate Tab**: Used for creating and managing pipelines.
- **Develop/Scripts Tab**: Create SQL scripts, Spark scripts, and notebooks.
- **Data Tab**: Manage and query databases like SQL or Lake databases (optimized for Spark processing).

Step 3: Grant Synapse Access to Data Lake

1. **Use System Managed Identity**:
    - System-managed identity is needed to authorize Synapse’s access to the data lake.
2. **Assign IAM Role to Synapse**:
    - Go to your storage account's *Access Control (IAM)* settings.
    - Add *Storage Blob Data Contributor* role for the Synapse workspace’s managed identity.

Step 4: Configure SQL Pools in Synapse

1. **Create a New SQL Script**:
    - Go to the *Develop* tab in Synapse, and create a new SQL script.
2. **Select SQL Pool Type**:
    - **Dedicated SQL Pool**: Stores data traditionally in databases (e.g., MySQL, PostgreSQL).

    - **Serverless SQL Pool**: Doesn't store data in a database; instead, it uses a lakehouse approach, ideal for scalable and flexible storage.
3. **Note**: Serverless SQL Pool is cost-effective as data is stored in the data lake, which is cheaper than traditional data warehouses.

Step 5: Pull Transformed Data from Data Lake Using SQL

1. **Assign Required Roles**:
    - Ensure your Azure Synapse user has the *Storage Blob Data Contributor* role for querying data from the data lake.
2. **Query Data with OPENROWSET**:
    - Use `OPENROWSET` to apply an abstraction layer for querying data from the transformed container in your data lake.

sql
Copy code
```sql
SELECT * FROM
OPENROWSET(
    BULK 'your_data_url_from_Transformed_container',  -- Replace with
data lake URL up to the folder level
    FORMAT = 'PARQUET'
) AS query1;
```

3.
4. **Creating Views for Serving Layer**:
    - Create views for easy access to the transformed data:

sql
Copy code
```sql
CREATE SCHEMA schema_name;

CREATE VIEW schema_name.table_name AS
SELECT * FROM
OPENROWSET(
    BULK 'your_data_url_from_Transformed_container',
    FORMAT = 'PARQUET'
) AS query1;
```

5.

Step 6: Create External Tables

1. **Create Credentials**:

    - Use managed identity, SAS token, or access keys to authenticate.
2. **Define External Data Source**:
    - Define a reusable data source for frequent access:

sql
Copy code
```
CREATE EXTERNAL DATA SOURCE source_transformed
WITH (
    LOCATION =
'https://datalake_name.blob.core.windows.net/container_name',
    CREDENTIAL = credential_name
);

CREATE EXTERNAL DATA SOURCE source_servinglayer
WITH (
    LOCATION =
'https://datalake_name.blob.core.windows.net/container_name',
    CREDENTIAL = credential_name
);
```

3.
4. **Specify External File Format**:
    - Define the file format (e.g., Parquet) for data retrieval:

sql
Copy code
```
CREATE EXTERNAL FILE FORMAT format_parquet
WITH (
    FORMAT_TYPE = PARQUET,
    DATA_COMPRESSION = 'org.apache.hadoop.io.compress.SnappyCodec'
);
```

5.
6. **Create External Table for Data Storage**:
    - Create an external table based on the transformed data views:

sql
Copy code
```
CREATE EXTERNAL TABLE schema_name.ext_table_name
WITH (
    LOCATION = 'Folder_name',
    DATA_SOURCE = source_transformed,
    FILE_FORMAT = format_parquet
)
AS SELECT * FROM schema_name.table_name;
```

7.

Step 7: Connect Synapse with Power BI

1. **Retrieve Serverless SQL Endpoint**:
   - Go to *Synapse Workspace -> Overview*, copy the *Serverless SQL Endpoint*.
2. **Connect Power BI to Synapse**:
   - Open *Power BI Desktop*.
   - Select *Get Data -> Azure Synapse Analytics SQL* and enter the SQL endpoint URL to connect.
3. **Visualize Data**:
   - Use Power BI to create visualizations based on the external table created in Synapse.

This guide will help you set up Azure Synapse, pull transformed data from Azure Data Lake, and connect it with Power BI for visualization, enabling you to leverage Azure Synapse as a robust, scalable analytics solution.