Azure Data Lake Storage (ADLS) and Snowflake Stages

This project outlines the design and implementation of a modern, scalable, and secure data ingestion pipeline. The solution leverages **Azure Data Lake Storage (ADLS) Gen2** as the primary, cost-effective landing zone for raw data from various source systems. This data is then efficiently loaded into the **Snowflake** cloud data warehouse for high-performance analytics and business intelligence.

The architecture decouples storage and compute, utilizing Snowflake's external stages to read data directly from ADLS, thereby optimizing data transfer costs and performance.

Objectives

The primary objectives of this project are:

- **Centralized Raw Data Repository:** To establish ADLS Gen2 as the single source of truth for all raw, unstructured, and structured data within the organization.
- **Efficient Data Loading:** To implement a robust and automated mechanism for loading data from ADLS into Snowflake using Snowflake Stages and the COPY command.
- **Scalability and Performance:** To design a system that can seamlessly scale to handle growing data volumes without compromising performance.
- **Security and Governance:** To ensure data is secured at rest and in transit using Azure and Snowflake security features, including Managed Identity and access controls.
- **Cost Optimization:** To minimize data transfer costs and leverage the most cost-effective storage and compute resources.

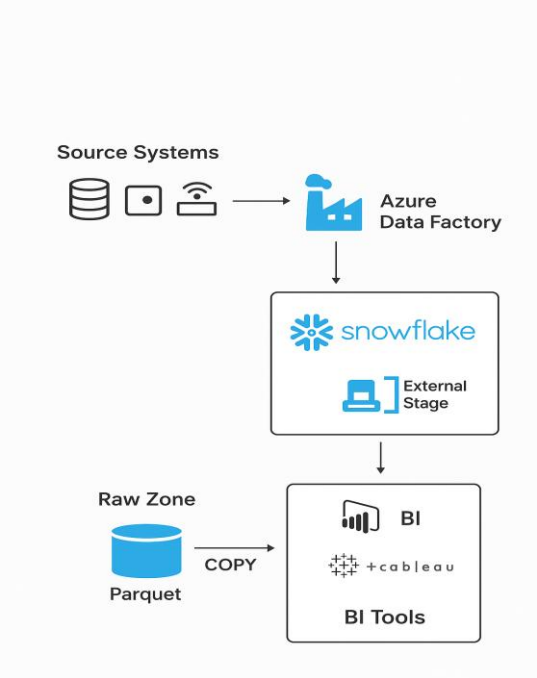
System Architecture & Design

1. Technology Stack

- **Cloud Platform:** Microsoft Azure
- **Data Lake Storage:** Azure Data Lake Storage Gen2
- **Data Warehouse:** Snowflake
- **Orchestration:** Azure Data Factory (ADF) / Apache Airflow
- **File Format:** Parquet (Recommended for performance and schema evolution)
- **Security:** Azure Managed Identity, Snowflake Role-Based Access Control (RBAC)

2. Data Flow

The end-to-end data flow is depicted in the diagram below and follows these steps:



1. **Data Ingestion:** Source systems (e.g., on-premises SQL Server, SaaS applications, IoT streams) land their data into the ADLS Raw Zone. This is orchestrated by Azure Data Factory, which handles connectivity, extraction, and landing.
2. **Data Staging:** Data is stored in ADLS in a partitioned directory structure and in a columnar format like Parquet.
3. **Data Loading:** A Snowflake external stage is configured to point to the specific ADLS container and path. An internal Snowflake task or an external orchestrator (like ADF) executes a COPY INTO command, which reads the data from the external stage and loads it into the target Snowflake table.
4. **Data Consumption:** Analytics teams and BI tools query the processed data directly within Snowflake for reporting, dashboards, and advanced analytics.

3. Data Model

The data model in Snowflake typically follows a layered approach:

- **Raw/Landing Layer (STG):** Mirrors the raw data structure from ADLS. Tables in this layer are often transient and used for initial loading.
- **Cleaned/Integrated Layer (CLEAN/INT):** Data is cleansed, standardized, and integrated from multiple sources.
- **Business Layer (MART):** Data is modeled into dimensional models (Star/Snowflake schemas) optimized for business reporting.

This project focuses on populating the **Raw/Landing Layer** from ADLS.

Implementation Guide

1. Prerequisites

- An active **Azure subscription** with owner privileges.
- An active **Snowflake** account with the ACCOUNTADMIN role or a custom role with sufficient privileges.
- Basic knowledge of SQL, Azure Portal, and Snowflake web interface.

2. Step 1: Azure Data Lake Storage (ADLS) Gen2 Configuration

1. **Create a Storage Account:**
    - Navigate to the Azure Portal and create a new Storage Account.
    - Ensure the "Hierarchical namespace" (for ADLS Gen2) is enabled.
2. **Create Containers:**
    - Create containers to act as logical boundaries. For example:
        - raw-data: For initial data landing.
        - processed-data: For any intermediate processed files (if needed).
3. **Configure Security with Managed Identity:**
    - This is the recommended, secure method for Snowflake to access ADLS without storing access keys.
    - In Azure Active Directory (AAD), register an App Registration (this represents Snowflake).
    - Navigate to your Storage Account > **Access Control (IAM)**.
    - Click "Add role assignment" and assign the **Storage Blob Data Contributor** role to the registered application.

3. Step 2: Snowflake Configuration

1. **Create a Storage Integration:**
    - This object in Snowflake securely stores the Azure tenant and service principal information.

```
CREATE STORAGE INTEGRATION azure_data_lake_int
  TYPE = EXTERNAL_STAGE
  STORAGE_PROVIDER = 'AZURE'
  ENABLED = TRUE
  AZURE_TENANT_ID = '<your-azure-tenant-id>'
  STORAGE_ALLOWED_LOCATIONS = ('azure://youraccount.blob.core.windows.net/raw-data/');
```

    - After creation, run DESC STORAGE INTEGRATION azure_data_lake_int; to

retrieve the AZURE_CONSENT_URL and AZURE_MULTI_TENANT_APP_NAME. You must grant admin consent in the Azure portal using the provided URL.

2. **Create an External Stage:**
    - The stage points to the location in your ADLS container.

```
CREATE STAGE adls_raw_stage
  URL = 'azure://youraccount.blob.core.windows.net/raw-data/sales/'
  STORAGE_INTEGRATION = azure_data_lake_int
  FILE_FORMAT = (TYPE = PARQUET COMPRESSION = SNAPPY);
```

    - Verify the stage works: LIST @adls_raw_stage;
3. **Create Target Table:**
    - Create a table in Snowflake with a schema that matches the Parquet files in ADLS.

```
CREATE OR REPLACE TABLE raw_sales_data (
  SaleID INTEGER,
  ProductName STRING,
  SaleAmount NUMBER(10,2),
  SaleDate DATE
);
```

4. Step 3: Creating the Data Ingestion Pipeline

The core of the loading process is the COPY INTO command. It is idempotent, meaning it will only load new files that haven't been loaded already, based on the file metadata.

```
CREATE OR REPLACE TABLE raw_sales_data (
  SaleID INTEGER,
  ProductName STRING,
  SaleAmount NUMBER(10,2),
  SaleDate DATE
);
```

5. Step 4: Orchestration and Scheduling

To automate this process, you can use **Azure Data Factory**:

1. Create a new ADF pipeline.

2. Add a "Web" activity to execute the COPY INTO command via Snowflake's SQL API, or use a "Copy Data" activity with Snowflake as a sink (though the external stage method is often more performant for large datasets).
3. Configure a schedule trigger to run the pipeline at a required frequency .