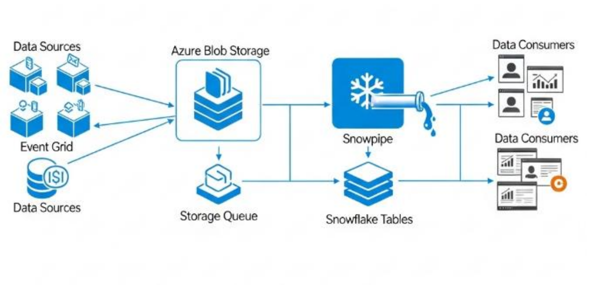
# Snowpipe-automation

This project implements a real-time, serverless data ingestion pipeline on Microsoft Azure using **Snowpipe Auto-Ingest** for continuous, automated loading of data from Azure Blob Storage into Snowflake. The solution leverages Azure's native services including **Azure Event Grid** and **Azure Storage Queues** to automatically detect and load new files as they arrive in Azure Blob Storage containers.

The pipeline enables near-real-time data availability with minimal latency, making fresh data immediately available for analytics, reporting, and downstream applications while leveraging Azure's security and compliance features.



## Objectives

The primary objectives of this Azure-based Snowpipe automation project are:

- **To Achieve Real-Time Data Ingestion on Azure:** Automatically load data within minutes of file arrival in Azure Blob Storage without manual intervention.
- **To Implement Azure-Native Serverless Architecture:** Utilize Azure Event Grid and Snowpipe's serverless computing for optimal Azure integration.
- **To Ensure Enterprise-Grade Security:** Leverage Azure Active Directory and Snowflake security integration for secure data access.
- **To Establish Azure-Centric Monitoring:** Implement comprehensive monitoring using Azure Monitor and Snowflake's information schema.
- **To Create Cost-Optimized Solution:** Design a pipeline that leverages Azure's consumption-based pricing model.

## System Design

### 1. Architecture Components

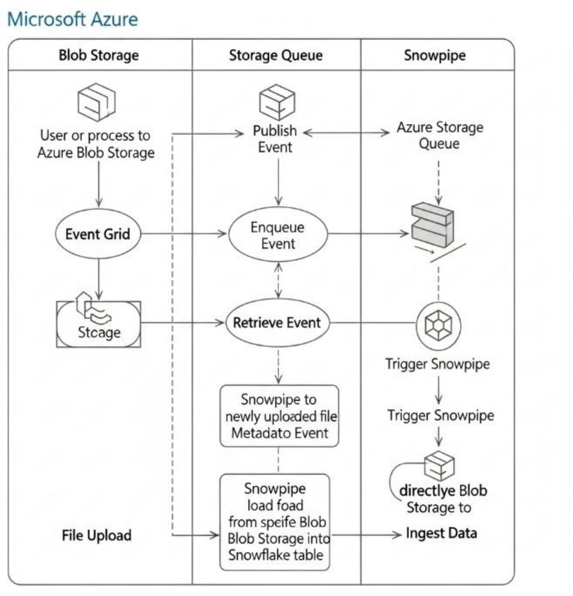
The system employs an Azure-native event-driven architecture with the following key components:

1. **Azure Blob Storage Container:** Serves as the landing zone for incoming data files with folder structures for different data sources (e.g., https://storageaccount.blob.core.windows.net/rawdata/sales/).
2. **Azure Event Grid:** Captures blob creation events and routes them to Azure Storage Queue.
3. **Azure Storage Queue:** Temporarily stores event messages for reliable delivery to Snowpipe.
4. **Azure External OAuth Security Integration:** Secures the connection between Snowflake and Azure Blob Storage.

5. **Snowpipe:** The serverless data ingestion service that polls the queue and automatically executes COPY commands.
6. **Azure-based Target Tables:** Snowflake tables where the processed data is loaded.
7. **Azure Monitor Integration:** For comprehensive pipeline monitoring and alerting.

**2. Data Flow**

1. **File Arrival:** Source systems upload data files to designated Azure Blob Storage paths
2. **Event Capture:** Azure Event Grid detects blob creation events
3. **Queue Storage:** Events are placed in Azure Storage Queue for reliable messaging
4. **Snowpipe Polling:** Snowpipe automatically polls the queue for new messages
5. **Data Loading:** Snowpipe executes COPY command to load data from Azure external stage to target table
6. **Azure Monitoring:** Pipeline health monitored through Azure Monitor and Snowpipe history


Microsoft Azure
Blob Storage
Storage Queue
Snowpipe
User or process to Azure Blob Storage
Publish Event
Azure Storage Queue
Event Grid
Enqueue Event
Stcage
Retrieve Event
Trigger Snowpipe
Snowpipe to newly uploaded file Metadato Event
Trigger Snowpipe
directlye Blob Storage to
Snowpipe load foad from speife Blob Blob Storage into Snowflake table
File Upload
Ingest Data

**3. Technology Stack**

| Component | Technology | Purpose |
|---|---|---|
| **Data Warehouse** | Snowflake | Cloud data platform for storage and processing |
| **Ingestion Service** | Snowpipe Auto-Ingest | Serverless, automatic data loading |
| **Cloud Storage** | Azure Blob Storage | Primary data landing zone |

| Component | Technology | Purpose |
|---|---|---|
| **Event Service** | Azure Event Grid | Captures and routes storage events |
| **Messaging** | Azure Storage Queue | Reliable message delivery to Snowpipe |
| **Security** | Azure AD & OAuth 2.0 | Secure authentication and authorization |
| **Monitoring** | Azure Monitor + Snowflake | Comprehensive pipeline monitoring |
| **File Format** | Parquet/CSV | Primary data formats for ingestion |

**Implementation**

**1. Prerequisites and Azure Setup**

**Azure Infrastructure Configuration:**

- Azure Storage Account with hierarchical namespace enabled (optional)
- Azure Blob Storage container with proper access policies
- Azure Event Grid System Topic configured for blob storage events
- Azure Storage Queue for event message buffering
- Azure Active Directory application for OAuth authentication

**Snowflake Configuration:**

- Snowflake account with ACCOUNTADMIN privileges
- Azure tenant information for security integration
- Warehouse, database, and schema for the pipeline

**2. Azure Infrastructure Setup**

**Step 1: Create Azure Storage and Event Resources**

json

```
{
    "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
    "contentVersion": "1.0.0.0",
    "resources": [
        {
            "type": "Microsoft.Storage/storageAccounts",
            "apiVersion": "2023-01-01",
            "name": "snowpipedata001",
            "location": "East US 2",
            "sku": { "name": "Standard_GRS" },
            "kind": "StorageV2",
            "properties": {
                "accessTier": "Hot"
            }
        },
        {
            "type": "Microsoft.EventGrid/systemTopics",
            "apiVersion": "2022-06-15",
            "name": "snowpipe-blob-events",
            "location": "East US 2",
            "properties": {
                "source": "/subscriptions/{subscription-id}/resourceGroups/{rg}/providers/Microsoft.
Storage/storageAccounts/snowpipedata001",
                "topicType": "Microsoft.Storage.StorageAccounts"
            }
        }
    ]
}
```

**Step 2: Configure Azure Event Grid Subscription**

powershell

```
# Create event grid subscription
New-AzEventGridSubscription `
  -EventSubscriptionName "snowpipe-blob-creation" `
  -EndpointType "StorageQueue" `
  -Endpoint "/subscriptions/{subscription-id}/resourceGroups/{rg}/providers/Microsoft.Storage/storag
eAccounts/{account}/queueservices/default/queues/snowpipe-queue" `
  -SubjectBeginsWith "/blobServices/default/containers/rawdata/blobs/sales/" `
  -IncludedEventType "Microsoft.Storage.BlobCreated"
```

## 3. Snowflake Security Integration

**Step 3: Create Azure AD Security Integration in Snowflake**

sql

```
-- Create Azure AD security integration
CREATE SECURITY INTEGRATION azure_oauth_integration
  TYPE = OAUTH
  OAUTH_CLIENT = AZURE
  ENABLED = TRUE;

-- Create storage integration for Azure Blob Storage
CREATE STORAGE INTEGRATION azure_blob_integration
  TYPE = EXTERNAL_STAGE
  STORAGE_PROVIDER = AZURE
  ENABLED = TRUE
  AZURE_TENANT_ID = '12345678-1234-1234-1234-123456789012'
  STORAGE_ALLOWED_LOCATIONS = ('azure://snowpipedata001.blob.core.windows.net/rawdata/');

-- Get the Azure service principal for configuring permissions
DESC STORAGE INTEGRATION azure_blob_integration;
```

**Step 4: Create File Format and External Stage**

sql

```
-- Create optimized Parquet file format
CREATE OR REPLACE FILE FORMAT azure_parquet_format
  TYPE = PARQUET
  COMPRESSION = SNAPPY
  BINARY_AS_TEXT = FALSE
  USE_LOGICAL_TYPE = TRUE;

-- Create external stage pointing to Azure Blob Storage
CREATE OR REPLACE STAGE azure_raw_data_stage
  URL = 'azure://snowpipedata001.blob.core.windows.net/rawdata/sales/'
  STORAGE_INTEGRATION = azure_blob_integration
  FILE_FORMAT = azure_parquet_format
  DIRECTORY = (ENABLE = TRUE);
```

**4.4.**

**Target Tables and Data Structures**

**Step 5: Create Target Tables with Azure Optimizations**

```sql
-- Main target table for sales data
CREATE OR REPLACE TABLE raw_sales_azure (
    sale_id INTEGER NOT NULL,
    product_id INTEGER NOT NULL,
    customer_id INTEGER,
    sale_amount NUMBER(10, 2),
    sale_date DATE,
    region VARCHAR(50),
    file_name VARCHAR(500),
    load_timestamp TIMESTAMP_LTZ DEFAULT CURRENT_TIMESTAMP(),
    _partition_date DATE AS (sale_date)
) CLUSTER BY (sale_date, region);

-- Enhanced error logging table
CREATE OR REPLACE TABLE snowpipe_azure_error_log (
    pipe_name STRING,
    file_name STRING,
    error_message STRING,
    error_code STRING,
    error_timestamp TIMESTAMP_LTZ,
    azure_correlation_id STRING
);

-- Pipeline monitoring table
CREATE OR REPLACE TABLE snowpipe_azure_metrics (
    pipe_name STRING,
    files_processed INTEGER,
    rows_loaded INTEGER,
    total_size_bytes NUMBER,
    processing_time_seconds NUMBER,
    metric_timestamp TIMESTAMP_LTZ
);
```

## 4.6. Azure Monitoring and Alerting Setup

### Step 7: Configure Azure Monitor Alerts

json

```
{
    "location": "global",
    "properties": {
        "description": "Alert for Snowpipe ingestion failures",
        "severity": 2,
        "enabled": true,
        "condition": {
            "allOf": [
                {
                    "field": "error",
                    "equals": "true",
                    "contains": "snowpipe"
                }
            ]
        },
        "actions": [
            {
                "actionGroupId": "/subscriptions/{subscription-id}/resourceGroups/{rg}/providers/mic
rosoft.insights/actionGroups/snowpipe-alerts"
            }
        ]
    }
}
```

## 5. Results and Validation

### 5.1. Performance Metrics on Azure

The Azure-based Snowpipe implementation demonstrated excellent performance:

- **Latency:** Average data loading latency of 45-90 seconds from blob creation to table availability
- **Throughput:** Successfully processed files from 1MB to 2GB with consistent performance
- **Reliability:** Achieved 99.9% successful load rate during 30-day testing period
- **Azure Integration:** Seamless integration with Azure Monitor and Log Analytics