Pull Past 1 Year Azure Cost Data into Databricks - Step-by-Step Guide

# PART 1: Setup Azure Cost API Access (via Azure UI)

1.1 Go to Azure Portal

- Visit: https://portal.azure.com  
- Login with your account.

1.2 Create an App Registration (Service Principal)

- Go to Azure Active Directory > App registrations > New registration.  
- Name: databricks-cost-access  
- Supported account types: Single tenant (or as needed)  
- Redirect URI: Leave blank.  
- Click Register.

1.3 Capture Important Details

- Note down Application (client) ID and Directory (tenant) ID.

1.4 Create a Client Secret

- Go to Certificates & secrets > New client secret.  
- Description: databricks-cost-secret  
- Copy the Value immediately.

1.5 Assign Permissions to the App

- Go to Subscriptions > your subscription > Access control (IAM) > Add role assignment.  
- Role: Cost Management Reader.  
- Assign access to: User, group, or service principal.  
- Select your app: databricks-cost-access.

# PART 2: Understand the Azure Cost API (Cost Management Query API)

Endpoint: POST https://management.azure.com/subscriptions/{subscriptionId}/providers/Microsoft.CostManagement/query?api-version=2023-08-01

Authorization: Bearer token (from your app).

Request Body to Pull Past 1 Year Data:

{  
 "type": "ActualCost",  
 "timeframe": "Custom",  
 "timePeriod": {  
 "from": "2024-04-28T00:00:00Z",  
 "to": "2025-04-28T00:00:00Z"  
 },  
 "dataset": {  
 "granularity": "Daily",  
 "aggregation": {  
 "totalCost": {  
 "name": "PreTaxCost",  
 "function": "Sum"  
 }  
 }  
 }  
}

# PART 3: In Databricks - Pull the Past 1 Year Cost Data

3.1 Install Required Libraries  
%pip install azure-identity azure-mgmt-costmanagement

3.2 Authenticate using Client Credentials

from azure.identity import ClientSecretCredential  
  
TENANT\_ID = "<your-tenant-id>"  
CLIENT\_ID = "<your-client-id>"  
CLIENT\_SECRET = "<your-client-secret>"  
  
credential = ClientSecretCredential(  
 tenant\_id=TENANT\_ID,  
 client\_id=CLIENT\_ID,  
 client\_secret=CLIENT\_SECRET  
)

3.3 Get Access Token

token = credential.get\_token("https://management.azure.com/.default")  
access\_token = token.token

3.4 Call Azure Cost API and Handle Pagination

import requests  
import pandas as pd  
  
subscription\_id = "<your-subscription-id>"  
url = f"https://management.azure.com/subscriptions/{subscription\_id}/providers/Microsoft.CostManagement/query?api-version=2023-08-01"  
  
headers = {  
 "Authorization": f"Bearer {access\_token}",  
 "Content-Type": "application/json"  
}  
  
body = {  
 "type": "ActualCost",  
 "timeframe": "Custom",  
 "timePeriod": {  
 "from": "2024-04-28T00:00:00Z",  
 "to": "2025-04-28T00:00:00Z"  
 },  
 "dataset": {  
 "granularity": "Daily",  
 "aggregation": {  
 "totalCost": {  
 "name": "PreTaxCost",  
 "function": "Sum"  
 }  
 }  
 }  
}  
  
# Initialize  
all\_rows = []  
columns = []  
  
# First API call  
response = requests.post(url, headers=headers, json=body)  
data = response.json()  
  
# Capture data  
rows = data.get('properties', {}).get('rows', [])  
columns = [col['name'] for col in data.get('properties', {}).get('columns', [])]  
all\_rows.extend(rows)  
  
# Handle pagination  
nextLink = data.get('properties', {}).get('nextLink', None)  
  
while nextLink:  
 response = requests.post(f"https://management.azure.com{nextLink}", headers=headers)  
 data = response.json()  
 rows = data.get('properties', {}).get('rows', [])  
 all\_rows.extend(rows)  
 nextLink = data.get('properties', {}).get('nextLink', None)  
  
# Convert to DataFrame  
df = pd.DataFrame(all\_rows, columns=columns)

3.5 Save the Cost Data to Delta Table

# Convert to Spark DataFrame  
spark\_df = spark.createDataFrame(df)  
  
# Write to Delta table  
spark\_df.write.format("delta").mode("overwrite").saveAsTable("azure\_cost\_data\_full\_year")