Accessing Harmonized Landsat Sentinel-2 (HLS) Data on Azure: Tutorial Guide

This guide demonstrates how to access and visualize NASA's Harmonized Landsat Sentinel-2 (HLS) data hosted on Azure Blob Storage using Python. You'll learn how to:

- Generate a SAS token for secure access
- Query available HLS tiles by location and date
- Construct blob URLs for downloading imagery
- Visualize HLS imagery using GDAL and rasterio

Prerequisites

- Python 3.7+
- The following Python packages:
  - `requests`
  - `pandas`
  - `numpy`
  - `matplotlib`
  - `rasterio`
  - `azure-storage-blob`

Install them with:

```
pip install requests pandas numpy matplotlib rasterio azure-storage-blob
```

---

1. Generate a SAS Token

To securely access HLS data, request a SAS token from the Planetary Computer API.

```
import requests

def get_token_from_url(url):
    response = requests.get(url)
    response.raise_for_status()
    return response.json()["token"]

sasurl = "https://planetarycomputer.microsoft.com/api/sas/v1/token/hlssa/hls"
token = get_token_from_url(sasurl)
```

---

2. Set Up Azure Blob Storage Access

Configure the Azure Blob Storage client using the SAS token.

```
from azure.storage.blob import ContainerClient

container_name = 'hls'
storage_account_name = 'hlssa'
storage_account_url = f'https://{storage_account_name}.blob.core.windows.net/'

hls_container_client = ContainerClient(
    account_url=storage_account_url,
    container_name=container_name,
    credential=token
)
```

---

3. Find the HLS Tile for a Location

Use the provided tile extents file to look up the HLS tile ID for your latitude and longitude.

```
import pandas as pd
import io

hls_tile_extents_url =
'https://ai4edatasetspublicassets.blob.core.windows.net/assets/S2_TilingSystem2-
1.txt'
s = requests.get(hls_tile_extents_url).content
hls_tile_extents = pd.read_csv(io.StringIO(s.decode('utf-8')), delimiter=r'\s+')

def lat_lon_to_hls_tile_id(lat, lon):
    for _, row in hls_tile_extents.iterrows():
        if (lat >= row.MinLat and lat <= row.MaxLat and
            lon >= row.MinLon and lon <= row.MaxLon):
            return row.TilID
    return None

lat, lon = 41.89655047211277, -111.4132464403312  # Example: Bear Lake, UT
tile_id = lat_lon_to_hls_tile_id(lat, lon)
```

---

4. List Available Tiles for a Date

```
def list_available_tiles(prefix):
    return [blob.name for blob in
hls_container_client.list_blobs(name_starts_with=prefix)]

year = '2019'
daynum = '001'
product = 'S30'  # 'S30' for Sentinel-2, 'L30' for Landsat
```

```
prefix = f"{product}/HLS.{product}.T{tile_id}.{year}{daynum}"
matches = list_available_tiles(prefix)
```

---

5. Construct the Blob URL and Visualize the Image

```
hls_blob_root = storage_account_url + container_name
sband = '_01'
version = 'v1.4'

blob_name = f"{product}/HLS.{product}.T{tile_id}.{year}{daynum}.{version}
{sband}.tif"
sentinel_url = f"{hls_blob_root}/{blob_name}?{token}"

import rasterio
import numpy as np
import matplotlib.pyplot as plt

with rasterio.open(sentinel_url, 'r') as raster:
    img = raster.read(1)
    plt.imshow(img, cmap='gray')
    plt.title('HLS Image')
    plt.axis('off')
    plt.show()
```

---

6. Find and Visualize a Landsat Tile from the Same Location

Increment the day number until a matching Landsat tile is found, then visualize as above.

References

- HLS Data on Azure
- Planetary Computer Documentation
- Ag-Analytics HLS API

This tutorial demonstrates best practices for accessing Azure-hosted geospatial data using SAS tokens and the Azure SDK for Python.