GEMS Tile Store Example Usage -- MSI Users

This notebook illustrates how to identify and obtain data from Sentinel 2 images of interest using the GEMS Tile Store. This is meant for Minnesota Supercomputing Institute (MSI) users. A separate notebook will be available for external users.

Set up an HTTP client using Python's request library

We use a `Session` object to store our API key and automatically include it in the header for each request.

Note that we have a `rapid_api_key.py` file in the same directory as this notebook. The file contains only the below line.

```
rapid_api_key = 'SECRET'
```

In [1]:
```
import pandas as pd
import rasterio
from rasterio.plot import show
from requests import Session
import warnings
warnings.filterwarnings('ignore')

from rapid_api_key import rapid_api_key

s = Session()
s.headers.update({'x-rapidapi-key': rapid_api_key})

base = 'https://gems-tile-store.p.rapidapi.com'
```

Search for local images

We use the `/sentinel2/msi/local/search` endpoint to find available Level-2A images for an arbitrary search area and date range.

In [2]:
```
area_of_interest = {
    "type": "Polygon",
    "coordinates": [
        [
            [
                -95.00976562499999,
                44.653024159812
            ],
            [
                -93.8671875,
                44.653024159812
            ],
            [
                -93.8671875,
                45.19752230305682
            ],
            [
                -95.00976562499999,
                45.19752230305682
            ],
            [
                -95.00976562499999,
                44.653024159812
            ]
        ]
    ]
}

start_date = '2020-08-15T15:53:00'
end_date = '2020-08-20T15:53:00'
params = {
    'start_date': start_date,
    'end_date': end_date,
    'processing_level': 'Level-2A'
}
res = s.post(f'{base}/sentinel2/msi/local/search', json=area_of_interest, params=params)
images = res.json()
pd.json_normalize(images)[['s3', 'http', 'meta.title', 'meta.uuid']]
```

Out[2]:

	s3	http	meta.title	meta.uuid
0	s3://gems-satellite-imagery/S2A_MSIL2A_2020081...	https://s3.msi.umn.edu/gems-satellite-imagery/...	S2A_MSIL2A_20200815T171901_N0214_R012_T15TUK_2...	7e01612d-5706-4332-9dbc-d442d37b1bee
1	s3://gems-satellite-imagery/S2B_MSIL2A_2020081...	https://s3.msi.umn.edu/gems-satellite-imagery/...	S2B_MSIL2A_20200817T170849_N0214_R112_T15TUK_2...	95d25534-6c0a-4ab7-a830-a46863ef88b2
2	s3://gems-satellite-imagery/S2B_MSIL2A_2020081...	https://s3.msi.umn.edu/gems-satellite-imagery/...	S2B_MSIL2A_20200817T170849_N0214_R112_T15TVK_2...	46004e58-210d-4a7e-825f-bd30468195dd
3	s3://gems-satellite-imagery/S2A_MSIL2A_2020081...	https://s3.msi.umn.edu/gems-satellite-imagery/...	S2A_MSIL2A_20200819T165901_N0214_R069_T15TVK_2...	81a01e6b-6843-40cf-a13c-9606d67a3a08
4	s3://gems-satellite-imagery/S2A_MSIL2A_2020081...	https://s3.msi.umn.edu/gems-satellite-imagery/...	S2A_MSIL2A_20200815T171901_N0214_R012_T15TUL_2...	6819737b-a523-4ffb-97b7-3f4f90048dcb
5	s3://gems-satellite-imagery/S2B_MSIL2A_2020081...	https://s3.msi.umn.edu/gems-satellite-imagery/...	S2B_MSIL2A_20200817T170849_N0214_R112_T15TUL_2...	45ac01fb-fbb6-4c7d-a86b-ef021f59ee7e
6	s3://gems-satellite-imagery/S2A_MSIL2A_2020081...	https://s3.msi.umn.edu/gems-satellite-imagery/...	S2A_MSIL2A_20200819T165901_N0214_R069_T15TVL_2...	94891905-b750-4b9c-8d6f-d53f322d3073
7	s3://gems-satellite-imagery/S2B_MSIL2A_2020081...	https://s3.msi.umn.edu/gems-satellite-imagery/...	S2B_MSIL2A_20200817T170849_N0214_R112_T15TVL_2...	9056dc6a-75b8-4f72-aada-a98839704b00

Access and plot data

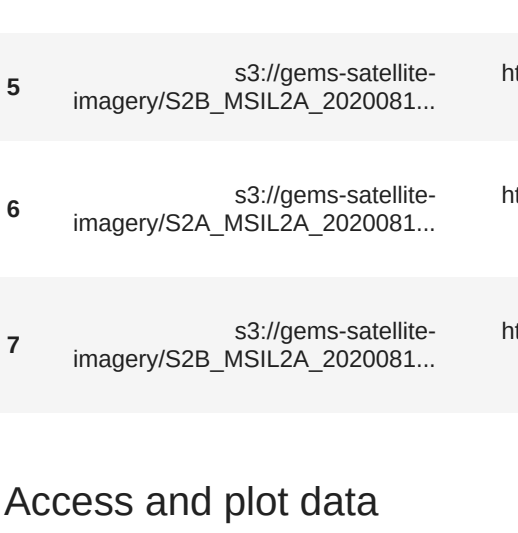
We use `rasterio` to open the file without writing to disk.

In [3]:
```
image = images[1]
image_path = image['http']
method = '/vsizip//vsicurl/'
with rasterio.open(f'{method}{image_path}') as ds:
    sub_dss = ds.subdatasets
    image_to_plot = [x for x in sub_dss if 'TCI' in x][0]
```

In [4]:
```
# Write to local file
with s.get(image_path) as ds:
    # write to local file
    with open('/home/mason/some_file.tif', 'wb') as f:
        f.write(ds.content)
```

To plot an image we use `rasterio`. We identify the True Color Image (TCI) subdataset path within the downloaded zip file then open and display the image.

In [5]:
```
with rasterio.open(image_to_plot) as img:
    show(img)
```



In [ ]:
```
```