Accessing MODIS temperature data with the Planetary Computer STAC API

The planetary computer hosts three temperature-related MODIS 6.1 products:

- Land Surface Temperature/Emissivity Daily (11A1)
- Land Surface Temperature/Emissivity 8-Day (11A2)
- Land Surface Temperature/3-Band Emissivity 8-Day (21A2)

For more information about the products themselves, check out the User Guides at the bottom of this document.

Environment setup

This notebook works with or without an API key, but you will be given more permissive access to the data with an API key. The Planetary Computer Hub is pre-configured to use your API key.

In [1]:

```
import odc.stac
import planetary_computer
import pystac_client
import rich.table
```

Data access

The datasets hosted by the Planetary Computer are available from Azure Blob Storage. We'll use pystac-client to search the Planetary Computer's STAC API for the subset of the data that we care about, and then we'll load the data directly from Azure Blob Storage. We'll specify a `modifier` so that we can access the data stored in the Planetary Computer's private Blob Storage Containers. See Reading from the STAC API and Using tokens for data access for more.

In [2]:

```
catalog = pystac_client.Client.open(
    "https://planetarycomputer.microsoft.com/api/stac/v1",
    modifier=planetary_computer.sign_inplace,
)
```

Query for available data

MODIS is a global dataset with a variety of products available within each larger category (vegetation, snow, fire, temperature, and reflectance). The MODIS group contains a complete listing of available collections. Each collection's format follows `modis-{product}-061`, where `product` is the MODIS product id. The `-061` suffix indicates that all of the MODIS collections are part of the MODIS 6.1 update.

Let's access Land Surface Temperature/Emissivity Daily (11A1) data over Sacramento, CA in 2022 We'll get four images for the midseasonal months: March, June, September, and December. In the cell below we save the first granule for each month in the items dictionary

In [13]:

```
# Sacramento, CA
latitude = 38.6
longitude = -121.5
buffer = 2
bbox = [longitude - buffer, latitude - buffer, longitude + buffer, latitude
year = "2022"
months = {
    "March": "03",
    "June": "06",
    "September": "09",
    "December": "12",
}
items = dict()
all_items = dict()

print("list. the number of images for each month\n")
# Fetch the collection of interest and print available items
for name, number in months.items():
    datetime = f"{year}-{number}"
    search = catalog.search(
        collections=["modis-11A1-061"],
        bbox=bbox,
        datetime=datetime,
    )
    all_items[name] = search.item_collection()
    print(f"Available granules: {name}: {len(all_items[name])}")
    items[name] = all_items[name][0]

print("\ncloud fraction for first image\n")
for key, value in items.items():
    print(f"{key=}: {value.properties['eo:cloud_cover']=}")
```

```
list. the number of images for each month

Available granules: March: 184
Available granules: June: 124
Available granules: September: 120
Available granules: December: 112

cloud fraction for first image

key='March': value.properties['eo:cloud_cover']=58
key='June': value.properties['eo:cloud_cover']=20
key='September': value.properties['eo:cloud_cover']=15
key='December': value.properties['eo:cloud_cover']=53
```

Available assets

Each item has several available assets, including the original HDF file and a Cloud-optimized GeoTIFF of each subdataset.

Print all the assets for the March image

```
In [4]:
t = rich.table.Table("Key", "Title")
for key, asset in items["March"].assets.items():
    t.add_row(key, asset.title)
t
```

```
Out[4]:
┏━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━
┃ Key              ┃ Title
┡━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━
│ hdf              │ Source data containing all bands
│ QC_Day           │ Quality control for daytime LST and emissivity
│ Emis_31          │ Band 31 emissivity
│ Emis_32          │ Band 32 emissivity
│ QC_Night         │ Quality control for nighttime LST and emissivity
│ metadata         │ Federal Geographic Data Committee (FGDC) Metadata
│ LST_Day_1km      │ Daily daytime 1km grid Land-surface Temperature
│ Clear_day_cov    │ Day clear-sky coverage
│ Day_view_angl    │ View zenith angle of daytime Landsurface Temperatu
│ Day_view_time    │ (local solar) Time of daytime Land-surface Tempera
│ LST_Night_1km    │ Daily nighttime 1km grid Land-surface Temperature
│ Clear_night_cov  │ Night clear-sky coverage
│ Night_view_angl  │ View zenith angle of nighttime Landsurface Tempera
│ Night_view_time  │ (local solar) Time of nighttime Landsurface Temper
│ tilejson         │ TileJSON with default rendering
│ rendered_preview │ Rendered preview
└──────────────────┴──────────────────────────────────────────────────
```

Loading the data

For this example, we'll visualize the temperature data over Boise, Idaho. Let's grab each fire mask cover COG and load them into an xarray using odc-stac. The MODIS coordinate reference system is a sinusoidal grid, which means that views in a naïve XY raster look skewed. For visualization purposes, we reproject to a spherical Mercator projection for intuitive, north-up visualization.

Reproject on to spherical mercator

see https://epsg.io/3857

this will transform all 4 images at the same time

```
In [5]:
data = odc.stac.load(
    items.values(),
    crs="EPSG:3857",
    bands="LST_Day_1km",
    resolution=500,
    bbox=bbox,
)
```

```
raster = items["March"].assets["LST_Day_1km"].extra_fields["raster:bands"]
data = data["LST_Day_1km"] * raster[0]["scale"]
```

In [6]:
```
raster
```

Out[6]:
```
[{'unit': 'Kelvin',
  'scale': 0.02,
  'data_type': 'uint16',
  'spatial_resolution': 1000}]
```

Displaying the data

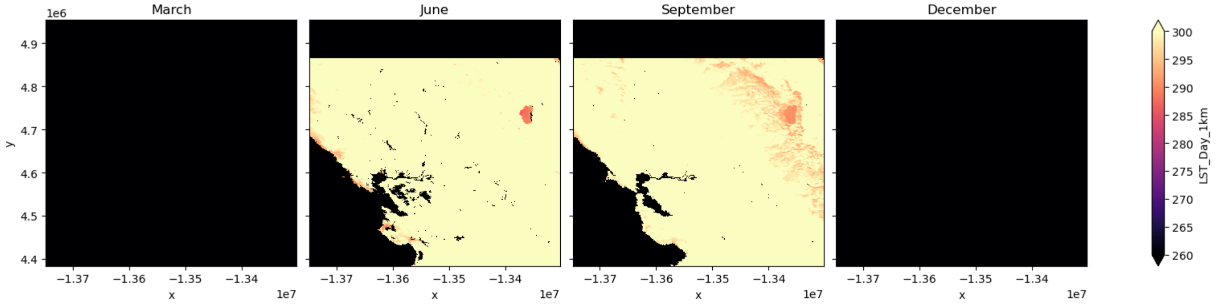
Let's display the temperature for each month.

In [7]:
```
print(F"{type(data)=}, {data.shape=}")
```

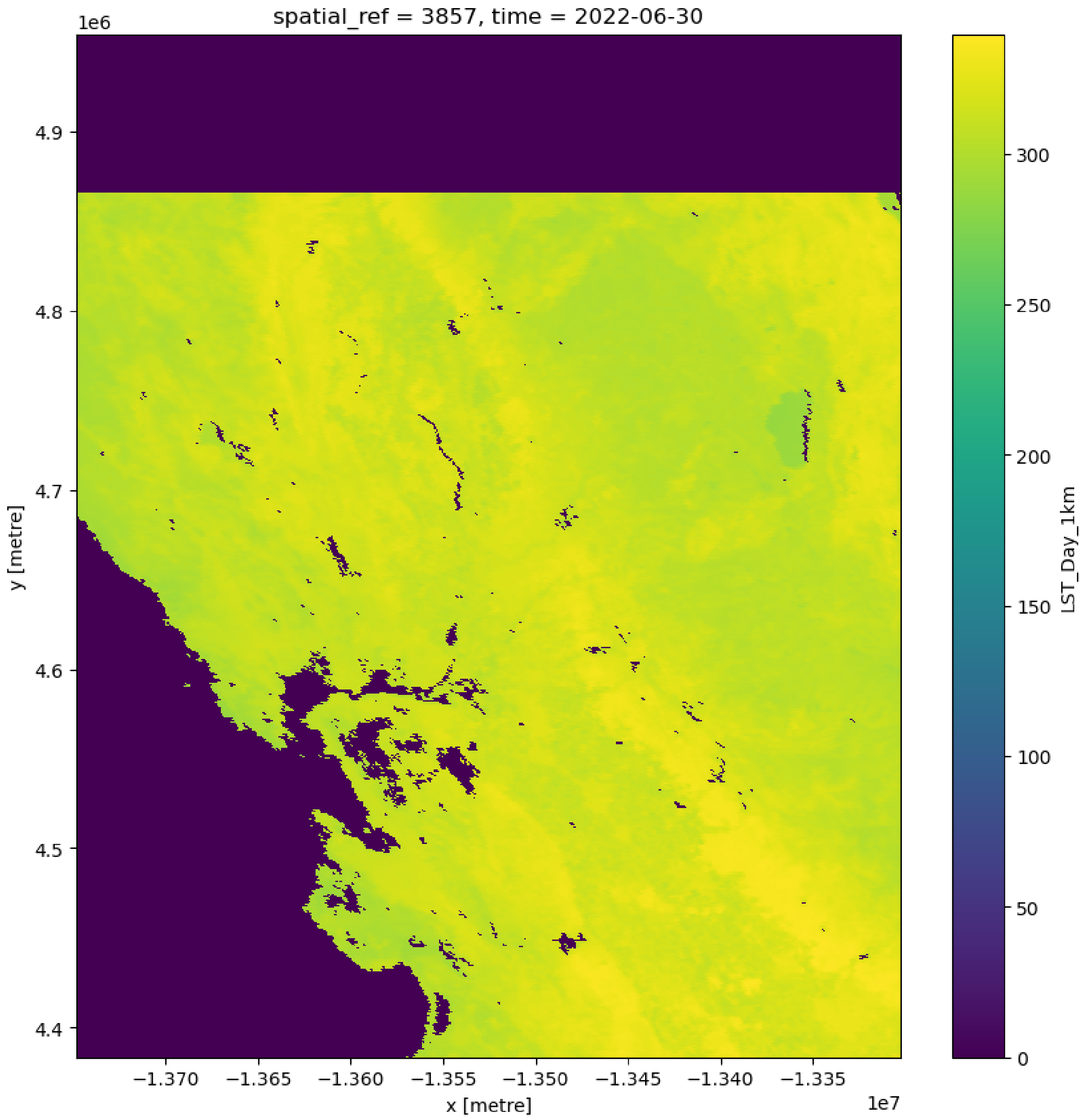
```
type(data)=<class 'xarray.core.dataarray.DataArray'>, data.shape=(4, 1141, 89
1)
```

In [8]:
```
g = data.plot.imshow(cmap="magma", col="time", vmin=260, vmax=300, size=4)
datetimes = data.time.to_pandas().dt.strftime("%B")

for ax, datetime in zip(g.axs.flat, datetimes):
    ax.set_title(datetime)
```



In [9]:
```
june_image = data[1]
from matplotlib import pyplot as plt
fig, ax = plt.subplots(1,1,figsize=(10,10))
june_image.plot.imshow(ax=ax);
```



Download the June geotif

Click on the link below to get the geotif in your downloads folder

```
In [10]: items['June'].assets['LST_Day_1km'].href
```

```
Out[10]: 'https://modiseuwest.blob.core.windows.net/modis-061-cogs/MYD11A1/08/05/202
2181/MYD11A1.A2022181.h08v05.061.2022182170719_LST_Day_1km.tif?st=2024-02-1
6T16%3A58%3A26Z&se=2024-02-24T16%3A58%3A26Z&sp=rl&sv=2021-06-08&sr=c&skoid=
c85c15d6-d1ae-42d4-af60-e2ca0f81359b&sktid=72f988bf-86f1-41af-91ab-2d7cd011
db47&skt=2024-02-17T16%3A58%3A25Z&ske=2024-02-24T16%3A58%3A25Z&sks=b&skv=20
21-06-08&sig=zycH%2BFSWv/V/AnvpDoQK3OLWZLG/xEOuAffxpUHXI6E%3D'
```

User guides

- MOD11: https://lpdaac.usgs.gov/documents/715/MOD11_User_Guide_V61.pdf
- MOD21: https://lpdaac.usgs.gov/documents/1398/MOD21_User_Guide_V61.pdf