→ EARTH OBSERVATION FOR SUSTAINABLE DEVELOPMENT

Climate Resilience

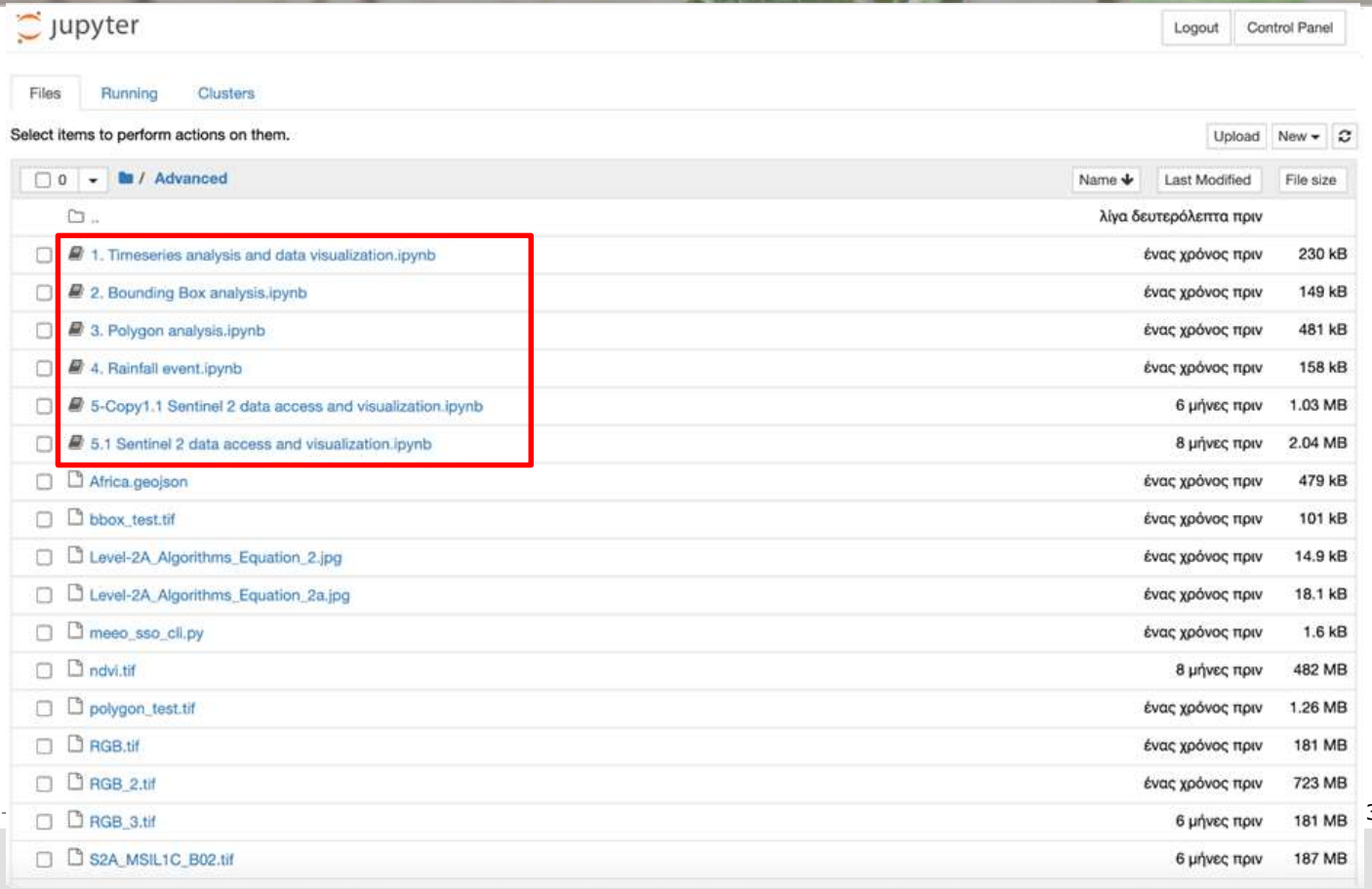
Webinar Series on how to use Earth Observation to tackle Climate Change

Webinar 06: How-to' Session: Using the EO4SD CR Platform to access EO data (hands-on)

4) Examples of pre-existing Notebooks

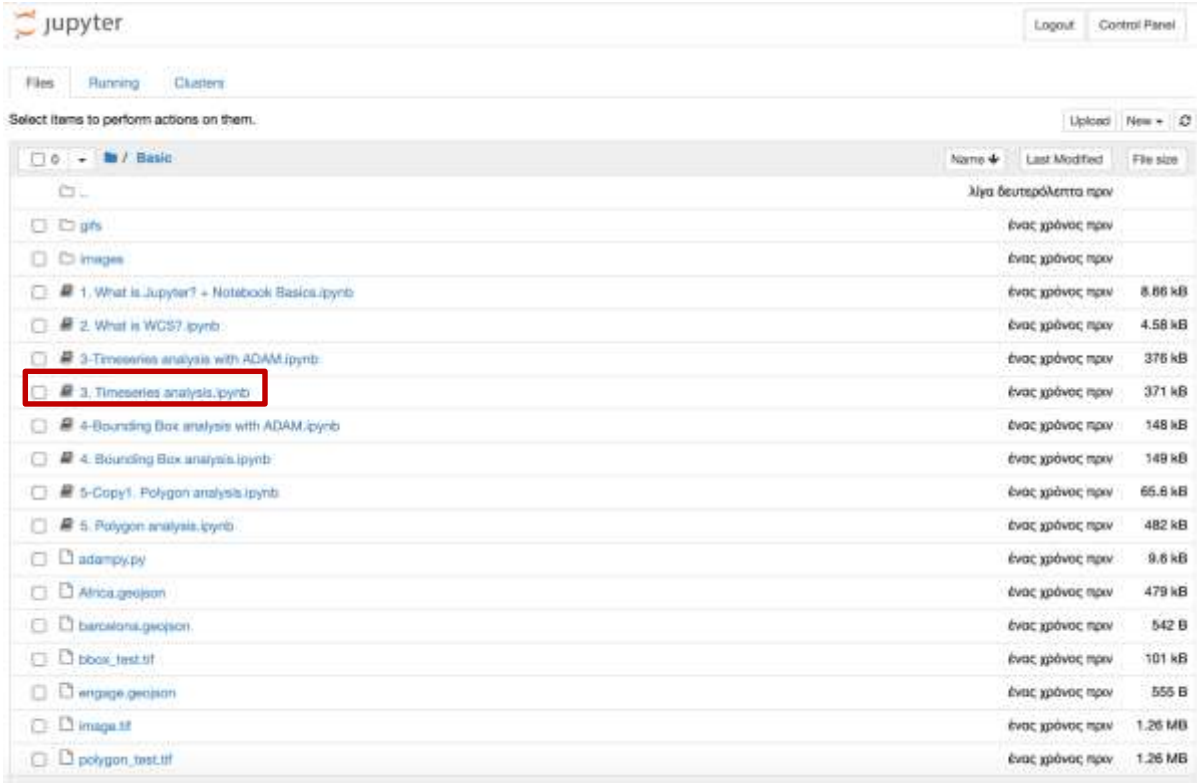
Dora Perrou, Research Associate, NOA

Examples of pre-existing Notebooks



Examples of pre-existing Notebooks

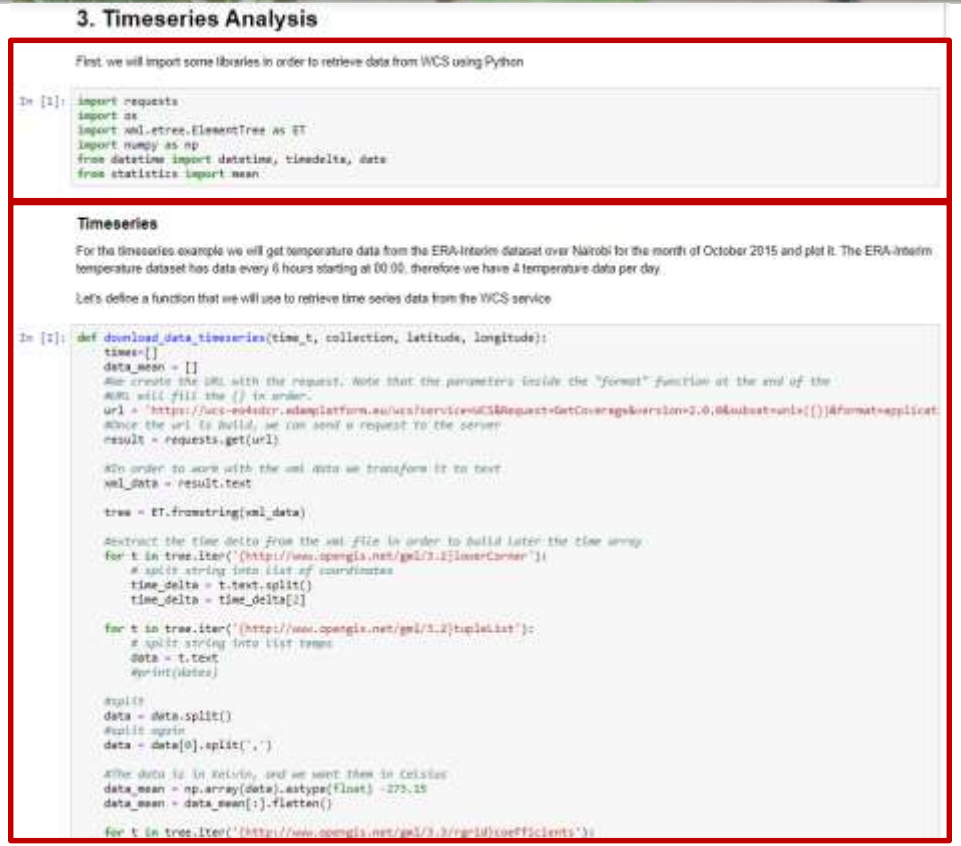
1. Time series analysis of temperature



Examples of pre-existing Notebooks

Timeseries analysis of temperature

- For the timeseries example we will get **temperature** data from the **ERA-Interim** dataset over **Nairobi** for the month of **October 2015** and **plot it**
- The ERA-Interim temperature dataset has data every 6 hours starting at 00:00, therefore we have 4 temperature data per day



3. Timeseries Analysis

First, we will import some libraries in order to retrieve data from WCS using Python

In [1]:
```
import requests
import os
import xml.etree.ElementTree as ET
import numpy as np
from datetime import datetime, timedelta, date
from statistics import mean
```

Timeseries

For the timeseries example we will get temperature data from the ERA-Interim dataset over Nairobi for the month of October 2015 and plot it. The ERA-Interim temperature dataset has data every 6 hours starting at 00:00, therefore we have 4 temperature data per day

Let's define a function that we will use to retrieve time series data from the WCS service

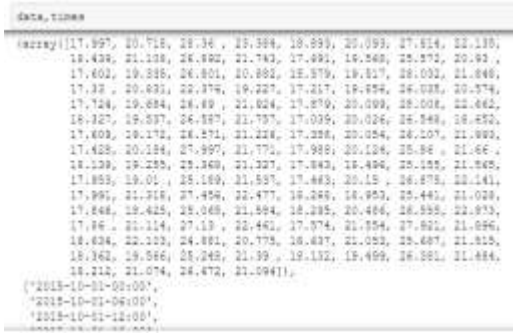
In [2]:
```
def download_data_timeseries(time_t, collection, latitude, longitude):
    times=[]
    data_mean = []
    #We create the URL with the request. Note that the parameters inside the "format" function at the end of the
    #URL will fill the {} in order.
    url = 'https://wcs-ea4ddcr.adamplatform.eu/wcs?service=WCS&Request=GetCoverage&version=2.0.0&subset=unix({})&format=applicat
    #Once the url is build, we can send a request to the server
    result = requests.get(url)

    #In order to work with the xml data we transform it to text
    xml_data = result.text

    tree = ET.fromstring(xml_data)

    #extract the time delta from the xml file in order to build later the time array
    for t in tree.iter('{http://www.opengis.net/gml/3.2}lowerCorner'):
        # split string into list of coordinates
        time_delta = t.text.split()
        time_delta = time_delta[2]

    for t in tree.iter('{http://www.opengis.net/gml/3.2}tupleList'):
        # split string into list temps
        data = t.text
        #print(dates)

    #split
    data = data.split()
    #split again
    data = data[0].split(',')

    #The data is in Kelvin, and we want them in Celsius
    data_mean = np.array(data).astype(float) -273.15
    data_mean = data_mean[:].flatten()

    for t in tree.iter('{http://www.opengis.net/gml/3.3/rgrid}coefficients'):
```

Timeseries analysis of temperature

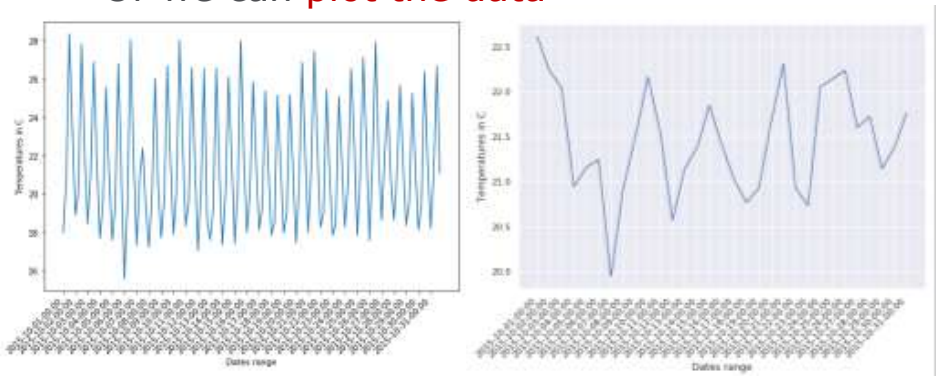
- We define the time range, the latitude and longitude of the point and the data collection

```
time_t = '2015-10-01T00:00:00,2015-10-31T23:59:59'
latitude = -1.286046
longitude = 36.820959
collection = 'ERA-Interim_temp2m_4326_05'
```

- We can print the raw data



- Or we can plot the data

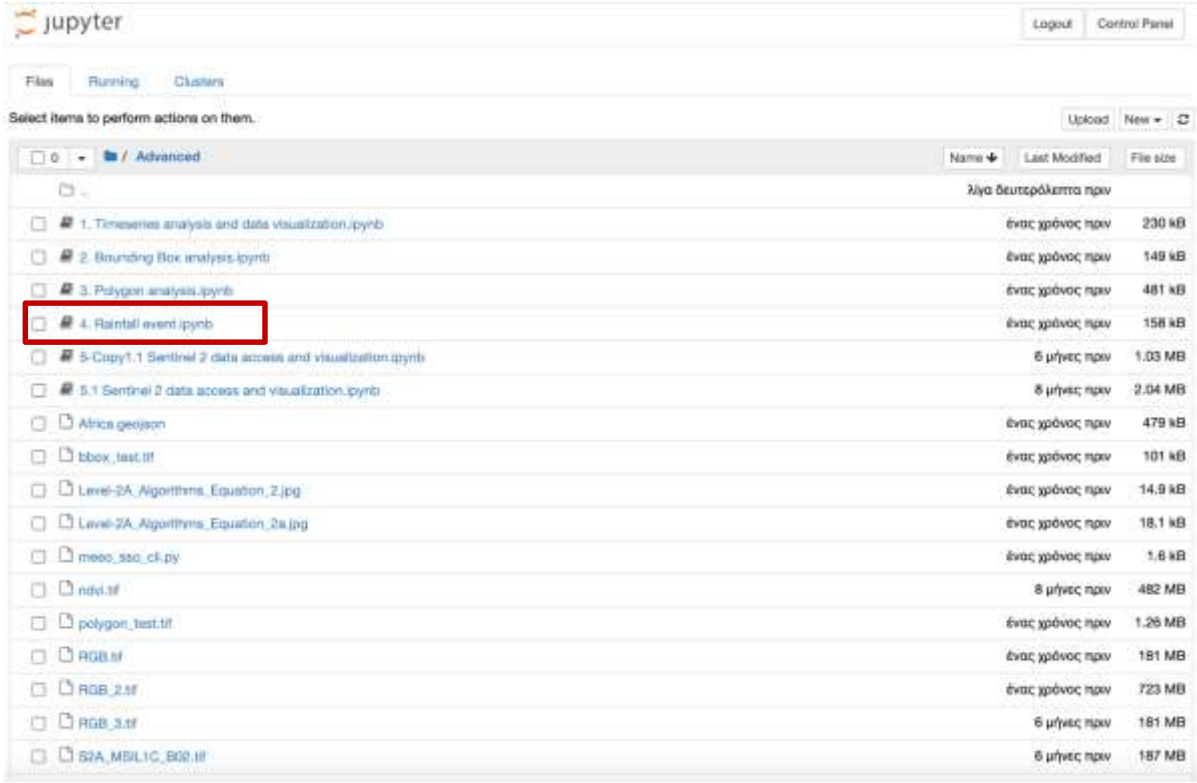


- We can also get the max, min and mean temperature

```
Maximum temperature: 28.360 degrees celsius
Minimum temperature: 15.579 degrees celsius
Mean temperature: 21.452 degrees celsius
```

Examples of pre-existing Notebooks

2. Rainfall event detection



Rainfall event detection

- We will check precipitation data and detect when a rainfall event occurs.
- This is very useful for Agriculture research in order to analyze historical data and detect potential risks in the fields

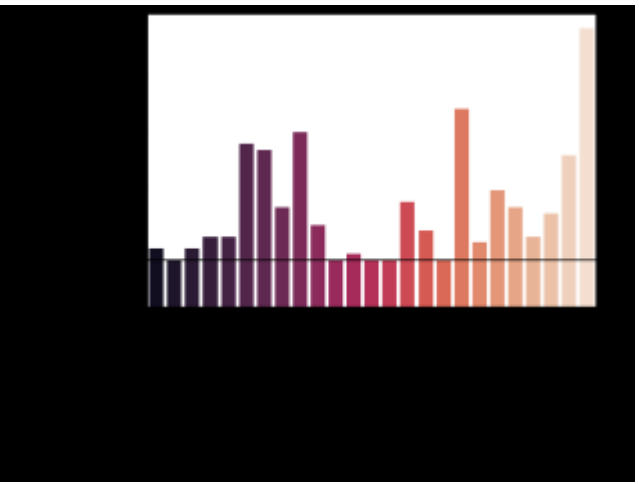
- We can check rainfall events during 2015 and **aggregate them monthly**
- As an example, we define a rainfall event: if the cumulative rain over **3 days is greater than 80 mm**

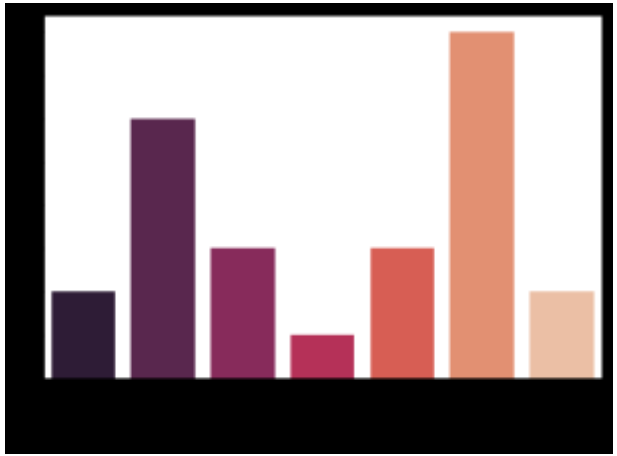
```
time_t = '2015-01-01T00:00:00,2015-12-31T23:59:59'
agg = 'm'
min_lat = max_lat = -1.286046
min_long = max_long = 36.820959
days = 3
amount = 80
```

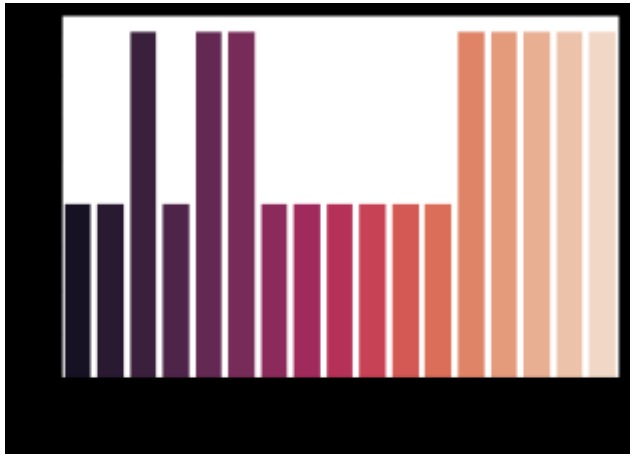
Examples of pre-existing Notebooks

Rainfall event detection

- We can check at the precipitation events and see in a barplot graph the amount of cumulative precipitation during the event
- We can also check at the data by the aggregation period we asked for, in this case the number of times:
  - per month or
  - per week

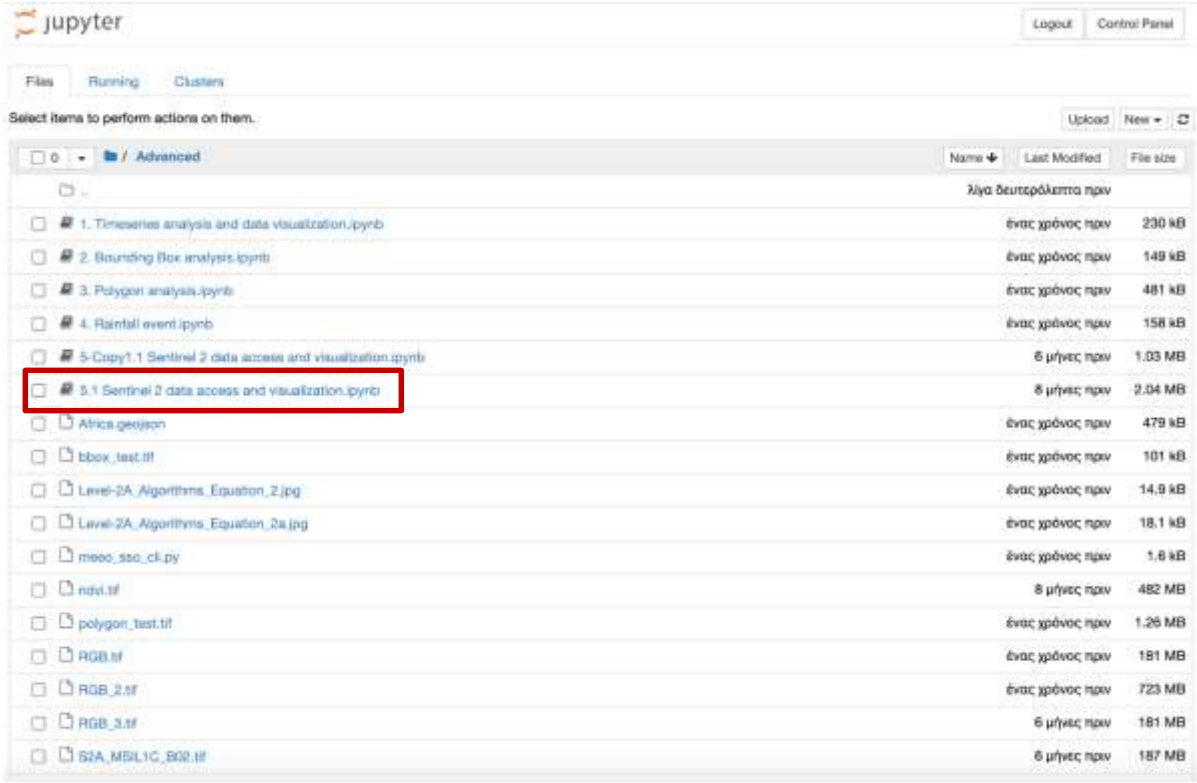






Examples of pre-existing Notebooks

3. Sentinel 2 data access and visualization



Examples of pre-existing Notebooks

In [4]:
```
import mgrs
```

In [5]:
```
#we should define a latitude and longitude (this is Nairobi's centroid coordinates)
latitude = 41.28
longitude = 0.57

#then we initialize mgrs
m = mgrs.MGRS()
#and pass the coordinates as arguments
c = m.toMGRS(latitude, longitude)
#since the response is a byte object, we need to decode it into a string
c = c.decode("utf-8")
#We finally add the 'T' and select only the first 5 characters so it matches the WCS format
tiles = 'T'+c[:5]
tiles
```

Out[5]:
```
'T31TBF'
```

In [6]:
```
bbox = ['-2','2','35','37'] #we are not going to use them now
#The names match the collection names in the server
band_list = ['S2A_MSIL1C_B11', 'S2A_MSIL1C_B08', 'S2A_MSIL1C_B02']
#This is the most recent cloud free image of Nairobi
time_t = '2019-06-30T00:00:00,2019-06-30T23:59:59'
```

Now we can download the 3 bands. We will loop the band_list array to make it easier.

In [7]:
```
access_token = connect_to_server()
for band in band_list:
    download(time_t,band,tiles, access_token)

print('Done!!')
```

```
https://creodias01.eodataservice.org/wcs?service=WCS&Request=GetCoverage&version=2.0.0&&format=image/tiff&filter=false&subset=uuid(2019-06-30T00:00:00,2019-06-30T23:59:59)&CoverageId=S2A_MSIL1C_B11&mgrs_tile=T31TBF&token=9c9e77466ba74f489f69d0d4d04350d3
https://creodias01.eodataservice.org/wcs?service=WCS&Request=GetCoverage&version=2.0.0&&format=image/tiff&filter=false&subset=uuid(2019-06-30T00:00:00,2019-06-30T23:59:59)&CoverageId=S2A_MSIL1C_B08&mgrs_tile=T31TBF&token=9c9e77466ba74f489f69d0d4d04350d3
https://creodias01.eodataservice.org/wcs?service=WCS&Request=GetCoverage&version=2.0.0&&format=image/tiff&filter=false&subset=uuid(2019-06-30T00:00:00,2019-06-30T23:59:59)&CoverageId=S2A_MSIL1C_B02&mgrs_tile=T31TBF&token=9c9e77466ba74f489f69d0d4d04350d3
Done!!
```

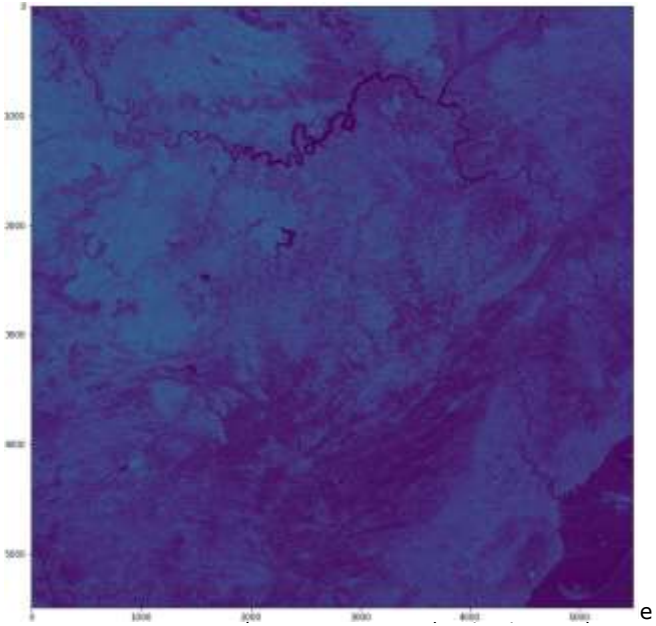
Once we have the images downloaded we can use Rasterio to stack the 3 images together and create and RGB image.

In [8]:
```
band_list = ['S2A_MSIL1C_B11.tif', 'S2A_MSIL1C_B08.tif', 'S2A_MSIL1C_B02.tif']

# Read metadata of first file
with rasterio.open(band_list[0]) as src0:
    meta = src0.meta

# Update meta to reflect the number of layers
meta.update(count = len(band_list))

# Read each layer and write it to stack
with rasterio.open('RGB_3.tif', 'w', **meta) as dst:
    for id, layer in enumerate(band_list, start=1):
        with rasterio.open(layer) as src1:
            dst.write_band(id, src1.read(1))
```

[min_lat,max_lat,min_long,max_long] that we can

Sentinel 2 data access and visualization

Build RGB



Sentinel 2 data access and visualization

Build NDVI

- The Normalized Difference Vegetation Index (NDVI) is a simple graphical indicator that can be used to assess whether an image being observed contains live green vegetation or not
- The formula for NDVI for Sentinel 2: $$\text{Normalized Difference Vegetation Index (NDVI)} = \frac{\text{Band 8} - \text{Band 4}}{\text{Band 8} + \text{Band 4}}$$
- Depending on the sensor we are working on, the bands might change
- For Sentinel 2, the European Space Agency (ESA) has a dedicated website explaining how to build different indexes

- First, we define which bands we need to download, in this case band 4 and band 8

```
ndvi_list = ['S2A_MSIL1C_B04', 'S2A_MSIL1C_B08']
access_token = connect_to_server()
for band in ndvi_list:
    download(time_t,band,tiles, access_token)

print('Done!!')
```

- Then we load the two bands with rasterio

In this way what we have are 2 numpy arrays

```
src = rasterio.open('S2A_MSIL1C_B04.tif')
b04 = src.read(1)
src.close()

src = rasterio.open('S2A_MSIL1C_B08.tif')
b08 = src.read(1)
src.close()
```

Sentinel 2 data access and visualization

Build NDVI

- We can compute the NDVI image

```
# Allow division by zero
np.seterr(divide='ignore', invalid='ignore')

# Calculate NDVI
#ndvi = (b04.astype(float) - b08.astype(float)) / (b04 + b08)
ndvi = (b08.astype(float) - b04.astype(float)) / (b08 + b04)
```

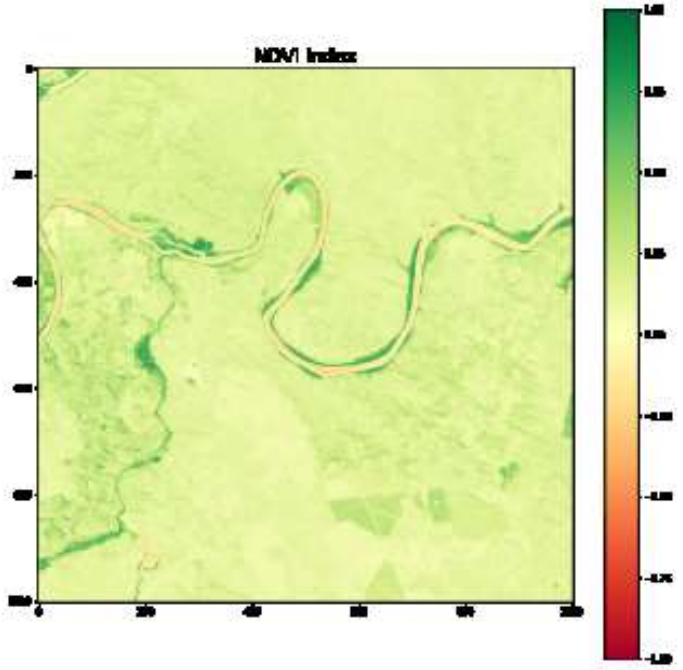
- We can pass all the spatial characteristics as arguments to the function

```
# Define spatial characteristics of output object
#(basically they are analog to the input)
kwargs = src.meta

# Update kwargs (change in data type)
kwargs.update(
    dtype=rasterio.float32,
    count = 1)

# Let's see what is in there
print(kwargs)
```

```
with rasterio.open('ndvi.tif', 'w', **kwargs) as dst:
        dst.write_band(1, ndvi.astype(rasterio.float32))
```

- And finally, we can write out the image

```
# Take a spatial subset of the ndvi layer produced
ndvi_sub = ndvi[2000:3000, 2000:3000]

# Plot
plt.subplots(figsize=(13,13))
plt.imshow(ndvi_sub, cmap='RdYlGn',vmin=-1, vmax=1)
plt.colorbar()
plt.title('NDVI index', size=20)
```



Build ad-hoc examples

Sentinel 2 data access and visualization

NDVI Timeseries

- We can access precomputed NVDI products using a WCS query

```
def download_timeseries(time_t,access_token):
    #url='https://creodias01.eodataservice.org/wcs?service=WCS&Request=GetCoverage&version=2.0.0&format=image/tiff&filter=falsea
    url='https://creodias01.eodataservice.org/wcs?service=WCS&Request=GetCoverage&version=2.0.0&format=application/xml&filter=fa
    print(url)
    #Once the url is build, we can send a request to the server
    result = requests.get(url)

    xml_data = result.text

    tree = ET.fromstring(xml_data)

    #extract the time delta from the xml file in order to build later the time array
    for t in tree.iter('{http://www.opengis.net/gml/3.2}lowerCorner'):
        # split string into list of coordinates
        time_delta = t.text.split()
        time_delta = time_delta[2]

    for t in tree.iter('{http://www.opengis.net/gml/3.2}tupleList'):
        # split string into list tuples
        data = t.text
        #print(data)

    times = []
    #splitsuper xml.etree.ElementTree as ET
    data = data.split()
    #split again
    data = data[0].split(',')

    data = [float(i) for i in data]
    for t in tree.iter('{http://www.opengis.net/gml/3.3/rgrid}coefficients'):
        # extract dates
        dates = t.text

    dates = dates.split()

    for i in range(0,len(dates)):
        dates[i] = int(dates[i]) + int(time_delta)

    for i in range(0,len(dates)):
        times.append(datetime.fromtimestamp(int(dates[i])).strftime('%Y-%m-%d')) #add hour min

    return times, data
```

- We define a time range e.g. period of one month (19.01.2019 – 18.02.2019)

```
time_t = '2019-01-19T00:00:00,2015-02-18T23:59:59'
access_token = connect_to_server()
times, data = download_timeseries(time_t, access_token)
```

- and we plot the data

```
f, ax = plt.subplots(figsize=(9, 6))
plt.plot(times, data)
ax.set_xticklabels(labels=times, rotation=45, ha='right')
ax.set(xlabel='Dates range', ylabel='NDVI')
```