

DRAGONHACK 2021

sentinelhub-py

Sinergise Team, May 22 - 23, 2021



ABOUT

- a Python API for SH services



ABOUT

- a Python API for SH services
- requests offer a wide functionality
 - images, raw data, extracted info
 - small or large area applications
 - single- or multi-date results
 - choose from multiple collections



ABOUT THIS PRESENTATION

- a set of examples for basic/advanced stuff
- examples of code and results
- based on `sentinelhub-py/examples/*`



FIRST STEPS

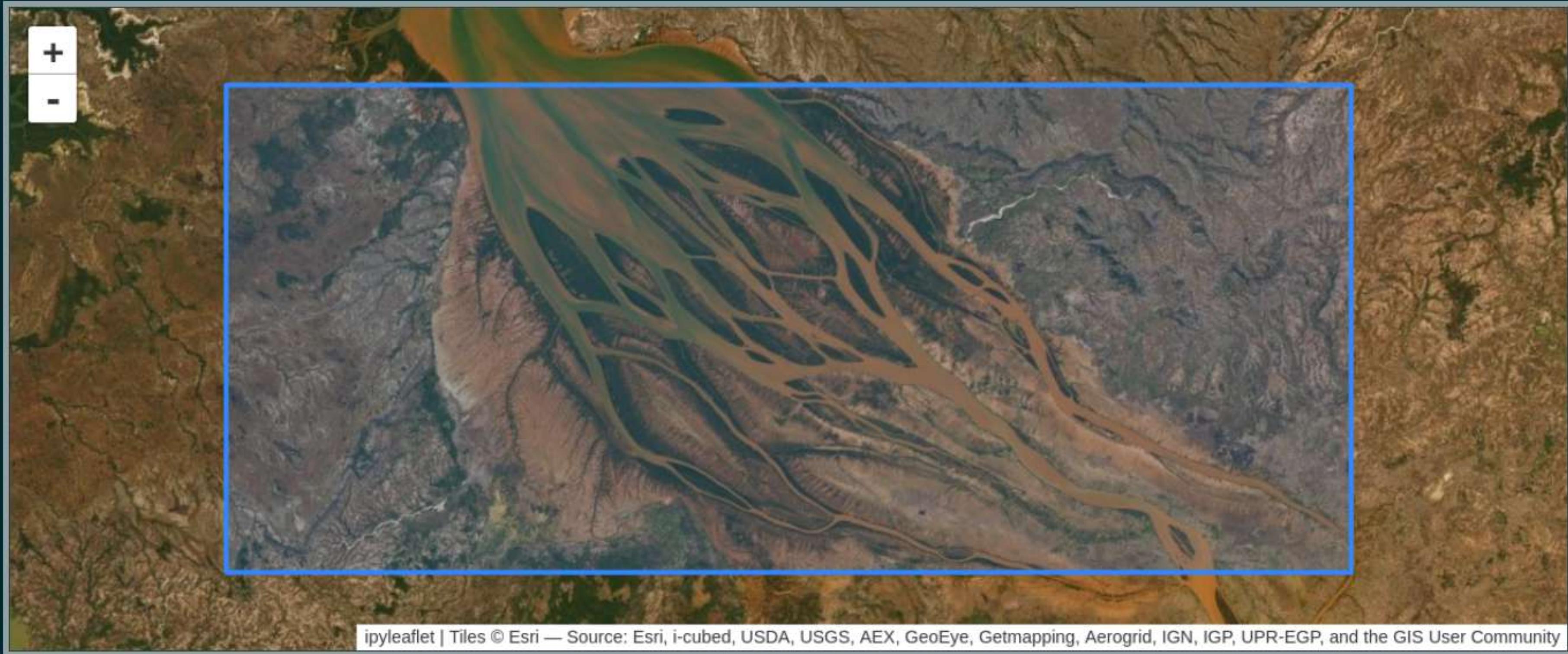
1. create a SH account
2. create an OAuth client
3. install `sentinelhub-py` (available via pip)
4. set up credentials

More info on [readthedocs](#).



SETTING UP THE SCENE

Let's focus on the area around the [Betsiboka estuary](#).



```
from sentinelhub import BBox, CRS

crs = CRS.WGS84
bounds = [46.16, -16.05, 46.64, -15.85]
resolution = 30

bbox = BBox(bbox=bounds, crs=crs)
size = bbox_to_dimensions(
    bbox,
    resolution=resolution
)
```

```
from sentinelhub import BBox, CRS

crs = CRS.WGS84
bounds = [46.16, -16.05, 46.64, -15.85]
resolution = 30

bbox = BBox(bbox=bounds, crs=crs)
size = bbox_to_dimensions(
    bbox,
    resolution=resolution
)
```

Basic imports

```
from sentinelhub import BBox, CRS

crs = CRS.WGS84
bounds = [46.16, -16.05, 46.64, -15.85]
resolution = 30
```

```
bbox = BBox(bbox=bounds, crs=crs)
size = bbox_to_dimensions(
    bbox,
    resolution=resolution
)
```

Define CRS, bounds and resolution



```
from sentinelhub import BBox, CRS

crs = CRS.WGS84
bounds = [46.16, -16.05, 46.64, -15.85]
resolution = 30

bbox = BBox(bbox=bounds, crs=crs)
size = bbox_to_dimensions(
    bbox,
    resolution=resolution
)
```

Image shape at 30 m resolution:
(726, 1718) pixels



EXAMPLE 1 - BASIC

- image from single date
- top-of-atmosphere (Sentinel-2 L1C)
- true color image



STEP 1 - EVALSCRIPT

```
//VERSION=3
function setup() {
    return {
        input: [
            bands: ["B02", "B03", "B04"] // B, G, R
        ],
        output: {
            bands: 3
        }
    };
}
function evaluatePixel(sample) {
    return [sample.B04, sample.B03, sample.B02];
}
```



STEP 1 - EVALSCRIPT

```
//VERSION=3
function setup() {
    return {
        input: [
            bands: ["B02", "B03", "B04"] // B, G, R
        ],
        output: {
            bands: 3
        }
    };
}
function evaluatePixel(sample) {
    return [sample.B04, sample.B03, sample.B02];
}
```

Tag for version

STEP 1 - EVALSCRIPT

```
//VERSION=3
function setup() {
    return {
        input: [
            bands: ["B02", "B03", "B04"] // B, G, R
        ],
        output: {
            bands: 3
        }
    };
}
function evaluatePixel(sample) {
    return [sample.B04, sample.B03, sample.B02];
}
```

Setup the input bands and the output



STEP 1 - EVALSCRIPT

```
//VERSION=3
function setup() {
    return {
        input: [
            bands: ["B02", "B03", "B04"] // B, G, R
        ],
        output: {
            bands: 3
        }
    };
}
function evaluatePixel(sample) {
    return [sample.B04, sample.B03, sample.B02];
}
```

Manipulate band data



```
evalscript_true_color = """
    //VERSION=3
    function setup() {
        return {
            input: [
                bands: ["B02", "B03", "B04"]
            ],
            output: {
                bands: 3
            }
        };
    }
    function evaluatePixel(sample) {
        return [sample.B04, sample.B03, sample.B02];
    }
"""

```

Just wrap the whole thing in a multi-line string



STEP 2 - REQUEST

```
request = SentinelHubRequest(  
    evalscript=evalscript_true_color,  
    input_data=[  
        SentinelHubRequest.input_data(  
            data_collection=DataCollection.SENTINEL2_L1C,  
            time_interval=('2020-06-12', '2020-06-13'))  
    ],  
    responses=[  
        SentinelHubRequest.output_response(  
            'default', MimeType.PNG)],  
    bbox=bbox,  
    size=size)  
  
data = request.get_data()
```



STEP 2 - REQUEST

```
request = SentinelHubRequest(  
    evalscript=evalscript_true_color,  
    input_data=[  
        SentinelHubRequest.input_data(  
            data_collection=DataCollection.SENTINEL2_L1C,  
            time_interval=('2020-06-12', '2020-06-13'))  
    ],  
    responses=[  
        SentinelHubRequest.output_response(  
            'default', MimeType.PNG)],  
    bbox=bbox,  
    size=size)  
  
data = request.get_data()
```

Set the evalscript



STEP 2 - REQUEST

```
request = SentinelHubRequest(  
    evalscript=evalscript_true_color,  
    input_data=[  
        SentinelHubRequest.input_data(  
            data_collection=DataCollection.SENTINEL2_L1C,  
            time_interval=('2020-06-12', '2020-06-13'))  
    ],  
    responses=[  
        SentinelHubRequest.output_response(  
            'default', MimeType.PNG)],  
    bbox=bbox,  
    size=size)  
  
data = request.get_data()
```



Set the data collection (top of atmosphere)

STEP 2 - REQUEST

```
request = SentinelHubRequest(  
    evalscript=evalscript_true_color,  
    input_data=[  
        SentinelHubRequest.input_data(  
            data_collection=DataCollection.SENTINEL2_L1C,  
            time_interval=('2020-06-12', '2020-06-13'))  
    ],  
    responses=[  
        SentinelHubRequest.output_response(  
            'default', MimeType.PNG)],  
    bbox=bbox,  
    size=size)  
  
data = request.get_data()
```

Set the time interval



STEP 2 - REQUEST

```
request = SentinelHubRequest(  
    evalscript=evalscript_true_color,  
    input_data=[  
        SentinelHubRequest.input_data(  
            data_collection=DataCollection.SENTINEL2_L1C,  
            time_interval=('2020-06-12', '2020-06-13'))  
    ],  
    responses=[  
        SentinelHubRequest.output_response(  
            'default', MimeType.PNG)],  
    bbox=bbox,  
    size=size)  
  
data = request.get_data()
```

Set the location and image size



STEP 2 - REQUEST

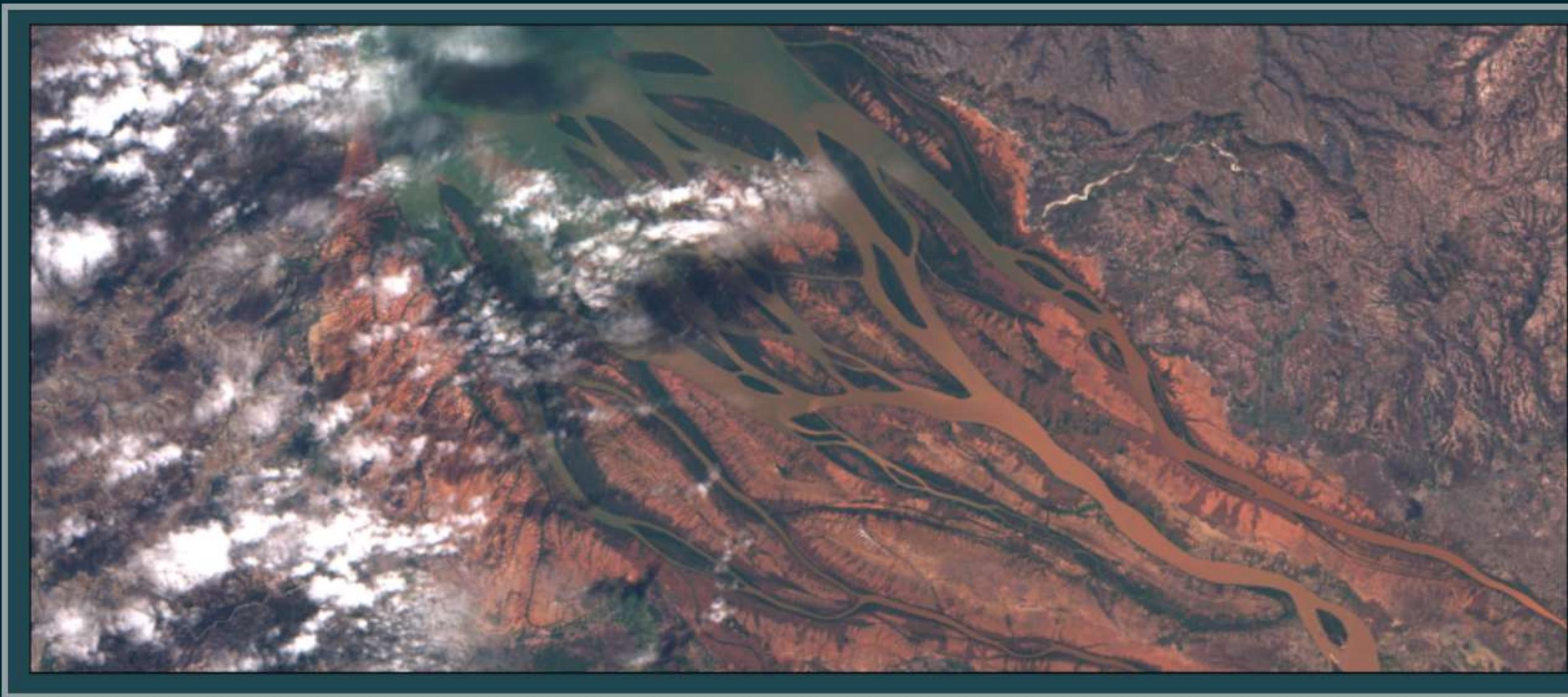
```
request = SentinelHubRequest(  
    evalscript=evalscript_true_color,  
    input_data=[  
        SentinelHubRequest.input_data(  
            data_collection=DataCollection.SENTINEL2_L1C,  
            time_interval=('2020-06-12', '2020-06-13'))  
    ],  
    responses=[  
        SentinelHubRequest.output_response(  
            'default', MimeType.PNG)],  
    bbox=bbox,  
    size=size)  
  
data = request.get_data()
```

Download data



HOW DOES IT LOOK?

- the output is of length 1 (single image returned)
- the image shape is (726, 1718, 3)



EXAMPLE 2 - CLOUD DETECTION

- can we detect clouds?
 - use precomputed CLM band
- atmospherically corrected data (Sentinel-2 L2A)



EVALSCRIPT

```
//VERSION=3
function setup() {
  return {
    input: ["B02", "B03", "B04", "CLM"],
    output: { bands: 3 }
  }
}
function evaluatePixel(sample) {
  if (sample.CLM == 1) {
    return [0.55 + sample.B04, sample.B03, sample.B02]
  }
  return [3.5*sample.B04, 3.5*sample.B03, 3.5*sample.B02];
}
```



EVALSCRIPT

```
//VERSION=3
function setup() {
    return {
        input: ["B02", "B03", "B04", "CLM"],
        output: { bands: 3 }
    }
}
function evaluatePixel(sample) {
    if (sample.CLM == 1) {
        return [0.55 + sample.B04, sample.B03, sample.B02]
    }
    return [3.5*sample.B04, 3.5*sample.B03, 3.5*sample.B02];
}
```

Add CLM band



EVALSCRIPT

```
//VERSION=3
function setup() {
    return {
        input: ["B02", "B03", "B04", "CLM"],
        output: { bands: 3 }
    }
}
function evaluatePixel(sample) {
    if (sample.CLM == 1) {
        return [0.55 + sample.B04, sample.B03, sample.B02]
    }
    return [3.5*sample.B04, 3.5*sample.B03, 3.5*sample.B02];
}
```

Increase red if clouds detected



REQUEST

```
request = SentinelHubRequest(  
    evalscript=evalscript_true_color,  
    input_data=[  
        SentinelHubRequest.input_data(  
            data_collection=DataCollection.SENTINEL2_L2A,  
            time_interval=('2020-06-12', '2020-06-13'))  
    ],  
    responses=[  
        SentinelHubRequest.output_response(  
            'default', MimeType.PNG)],  
    bbox=bbox,  
    size=size)
```



REQUEST

```
request = SentinelHubRequest(  
    evalscript=evalscript_true_color,  
    input_data=[  
        SentinelHubRequest.input_data(  
            data_collection=DataCollection.SENTINEL2_L2A,  
            time_interval=('2020-06-12', '2020-06-13'))  
    ],  
    responses=[  
        SentinelHubRequest.output_response(  
            'default', MimeType.PNG)],  
    bbox=bbox,  
    size=size)
```

Change data collection



RESULT

- clouds highlighted in red
- image more colorful (atmospheric corrections)



EXAMPLE 3 - MOSAIC

- longer time span (1 month)
- sort by tile-level cloud coverage
- same evalscript



REQUEST

```
request = SentinelHubRequest(  
    evalscript=evalscript_true_color,  
    input_data=[  
        SentinelHubRequest.input_data(  
            data_collection=DataCollection.SENTINEL2_L2A,  
            time_interval=('2020-06-01', '2020-06-30'),  
            mosaicking_order='leastCC')],  
    responses=[  
        SentinelHubRequest.output_response(  
            'default', MimeType.PNG)],  
    bbox=bbox,  
    size=size)
```



REQUEST

```
request = SentinelHubRequest(  
    evalscript=evalscript_true_color,  
    input_data=[  
        SentinelHubRequest.input_data(  
            data_collection=DataCollection.SENTINEL2_L2A,  
            time_interval=('2020-06-01', '2020-06-30'),  
            mosaicking_order='leastCC')],  
    responses=[  
        SentinelHubRequest.output_response(  
            'default', MimeType.PNG)],  
    bbox=bbox,  
    size=size)
```

Longer time span



REQUEST

```
request = SentinelHubRequest(  
    evalscript=evalscript_true_color,  
    input_data=[  
        SentinelHubRequest.input_data(  
            data_collection=DataCollection.SENTINEL2_L2A,  
            time_interval=('2020-06-01', '2020-06-30'),  
            mosaicking_order='leastCC')],  
    responses=[  
        SentinelHubRequest.output_response(  
            'default', MimeType.PNG)],  
    bbox=bbox,  
    size=size)
```

Take least cloudy tiles first



RESULT



EXAMPLE 4 - DEM

- how about elevation?
- digital elevation model (DEM) collection
- by default `uint8`
- need to return `float32` due to negative values



EVALSCRIPT

```
//VERSION=3
function setup() {
    return {
        input: ["DEM"],
        output: {
            bands: 1,
            sampleType: SampleType.FLOAT32
        }
    }
}
function evaluatePixel(sample) {
    return [sample.DEM]
}
```



EVALSCRIPT

```
//VERSION=3
function setup() {
    return {
        input: ["DEM"],
        output: {
            bands: 1,
            sampleType: SampleType.FLOAT32
        }
    }
}
function evaluatePixel(sample) {
    return [sample.DEM]
}
```

Specify output type



REQUEST

```
request = SentinelHubRequest(  
    evalscript=evalscript_dem,  
    input_data=[  
        SentinelHubRequest.input_data(  
            data_collection>DataCollection.DEM  
        )],  
    responses=[  
        SentinelHubRequest.output_response(  
            'default',  
           MimeType.TIFF)],  
    bbox=bbox,  
    size=size)
```



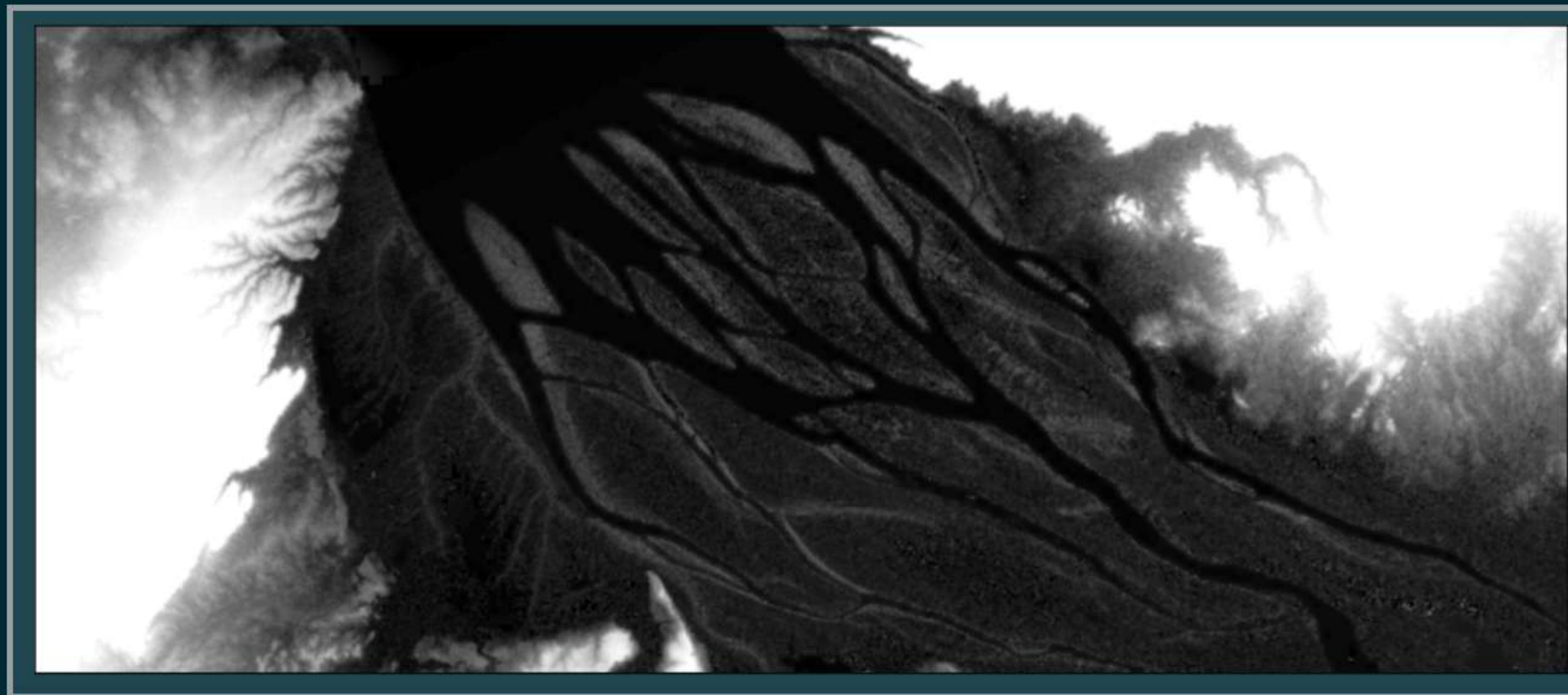
REQUEST

```
request = SentinelHubRequest(  
    evalscript=evalscript_dem,  
    input_data=[  
        SentinelHubRequest.input_data(  
            data_collection=DataCollection.DEM  
        )],  
    responses=[  
        SentinelHubRequest.output_response(  
            'default',  
            MimeType.TIFF)],  
    bbox=bbox,  
    size=size)
```

Same as before, but a different data collection



RESULT



EXAMPLE 5 - MONTHLY

- multiple images
- define monthly time ranges for 2019



REQUESTS

```
from sentinelhub import SentinelHubDownloadClient

.

.

.

req_list = [req1, req2, ...]
req_list = [req.download_list[0] for r in req_list]

# download data with multiple threads
data = SentinelHubDownloadClient().download(
    req_list,
    max_threads=5)
```



REQUESTS

```
from sentinelhub import SentinelHubDownloadClient

.
.

req_list = [req1, req2, ...]
req_list = [req.download_list[0] for r in req_list]

# download data with multiple threads
data = SentinelHubDownloadClient().download(
    req_list,
    max_threads=5)
```

Prepare a list of requests



REQUESTS

```
from sentinelhub import SentinelHubDownloadClient

.
.

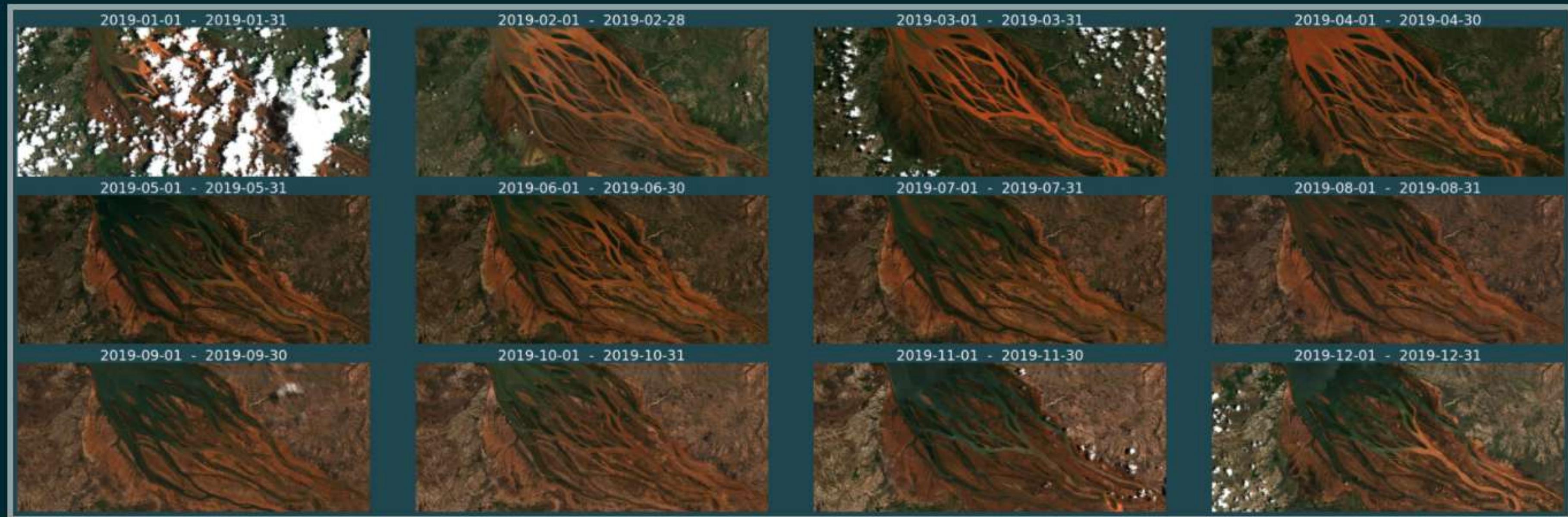
req_list = [req1, req2, ...]
req_list = [req.download_list[0] for r in req_list]

# download data with multiple threads
data = SentinelHubDownloadClient().download(
    req_list,
    max_threads=5)
```

Download in parallel



RESULT



EXAMPLE 6 - NDWI

- uncover the power of evalscripts
- extract normalized diff. water index (NDWI)
- max value over time interval

$$NDWI = \frac{B_3 - B_8}{B_3 + B_8}$$



EVALSCRIPT

```
//VERSION=3
function setup() {
    return {input: ["B03", "B08"],
            output: {bands: 1, sampleType: "FLOAT32"},
            mosaicking: "ORBIT"
    };
}
function evaluatePixel(samples) {
    let ndwi_array = [];
    for (i = 0; i < samples.length; i++) {
        ndwi_array.push(index(sample[i].B03, sample[i].B08));
    }
    max_ndwi = Math.max(...ndwi_array);
    return [max_ndwi];
}
```



EVALSCRIPT

```
//VERSION=3
function setup() {
    return {input: ["B03", "B08"],
            output: {bands: 1, sampleType: "FLOAT32"},
            mosaicking: "ORBIT"
    };
}
function evaluatePixel(samples) {
    let ndwi_array = [];
    for (i = 0; i < samples.length; i++) {
        ndwi_array.push(index(sample[i].B03, sample[i].B08));
    }
    max_ndwi = Math.max(...ndwi_array);
    return [max_ndwi];
}
```

Split data per each orbit



EVALSCRIPT

```
//VERSION=3
function setup() {
    return {input: ["B03", "B08"],
            output: {bands: 1, sampleType: "FLOAT32"},
            mosaicking: "ORBIT"
    };
}
function evaluatePixel(samples) {
    let ndwi_array = [];
    for (i = 0; i < samples.length; i++) {
        ndwi_array.push(index(sample[i].B03, sample[i].B08));
    }
    max_ndwi = Math.max(...ndwi_array);
    return [max_ndwi];
}
```

Loop over acquisitions



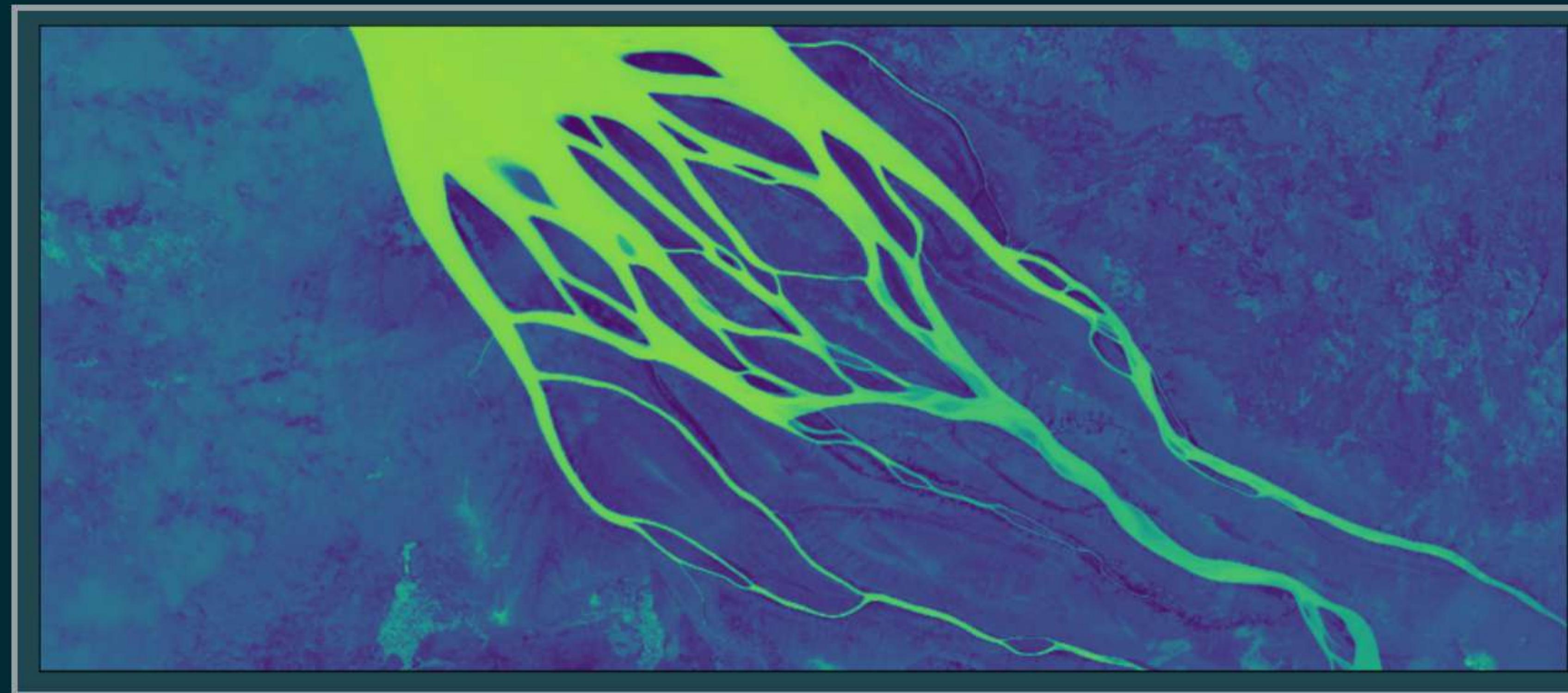
EVALSCRIPT

```
//VERSION=3
function setup() {
    return {input: ["B03", "B08"],
            output: {bands: 1, sampleType: "FLOAT32"},
            mosaicking: "ORBIT"
    };
}
function evaluatePixel(samples) {
    let ndwi_array = [];
    for (i = 0; i < samples.length; i++) {
        ndwi_array.push(index(sample[i].B03, sample[i].B08));
    }
    max_ndwi = Math.max(...ndwi_array);
    return [max_ndwi];
}
```

Return max for each pixel



RESULT



EXAMPLE 7 - SCL

- multipart response
- get scene classification layer (SCL)
- get dictionary of coverage per SCL value
- available in L2A



EVALSCRIPT

```
...
var val_names = ["no data", "defective", "dark area",
  "cloud shadow", "vegetation", "bare soils", "water",
  "clouds low", "clouds mid", "clouds hi", "cirrus", "snow"];
var colorMap = {
  0: [0, 0, 0], // No Data
  1: [1, 0, 0.016], // Saturated / Defective
  2: [0.525, 0.525, 0.525], // Dark Area Pixels
  3: [0.467, 0.298, 0.043], // Cloud Shadows
  4: [0.063, 0.827, 0.176], // Vegetation
  5: [1, 1, 0.325], // Bare Soils
  6: [0, 0, 1], // Water
  ...
};
...
```

Prepare a color map for transforming pixel values to colors



```
...
var scl_dict = {};
var count = 0;
function evaluatePixel(sample) {
    scl = sample.SCL;
    if (!(val_names[scl] in scl_dict)) {
        scl_dict[val_names[scl]] = 0;
    } else {
        scl_dict[val_names[scl]] += 1;
    }
    count += 1;
    return colorMap[scl];
}
...
```



```
...
var scl_dict = {};
var count = 0;
function evaluatePixel(sample) {
    scl = sample.SCL;
    if (!(val_names[scl] in scl_dict)) {
        scl_dict[val_names[scl]] = 0;
    } else {
        scl_dict[val_names[scl]] += 1;
    }
    count += 1;
    return colorMap[scl];
}
...
```

Initiate a dict and a counter outside of function



```
...
var scl_dict = {};
var count = 0;
function evaluatePixel(sample) {
    scl = sample.SCL;
    if (!(val_names[scl] in scl_dict)) {
        scl_dict[val_names[scl]] = 0;
    } else {
        scl_dict[val_names[scl]] += 1;
    }
    count += 1;
    return colorMap[scl];
}
...
```

Populate the dict and increase counter



```
...
var scl_dict = {};
var count = 0;
function evaluatePixel(sample) {
    scl = sample.SCL;
    if (!(val_names[scl] in scl_dict)) {
        scl_dict[val_names[scl]] = 0;
    } else {
        scl_dict[val_names[scl]] += 1;
    }
    count += 1;
    return colorMap[scl];
}
...
```

Return the color for each value



```
...
function updateOutputMetadata(scenes, inputMetadata,
    outputMetadata) {
    for (let scl in scl_dict) {
        scl_dict[scl] = scl_dict[scl]/count;
    }
    outputMetadata.userData = { "scl": scl_dict}
}
...
```



```
...
function updateOutputMetadata(scenes, inputMetadata,
    outputMetadata) {
    for (let scl in scl_dict) {
        scl_dict[scl] = scl_dict[scl]/count;
    }
    outputMetadata.userData = { "scl": scl_dict}
}
...
```

Divide values by count and return via userData



REQUEST

```
request = SentinelHubRequest(  
    evalscript=evalscript,  
    input_data=[  
        SentinelHubRequest.input_data(  
            data_collection=DataCollection.SENTINEL2_L2A,  
            time_interval=('2019-06-01', '2019-7-31'))  
    ],  
    responses=[  
        SentinelHubRequest.output_response(  
            'default', MimeType.TIFF),  
        SentinelHubRequest.output_response(  
            'userdata', MimeType.JSON)],  
    bbox=bbox,  
    size=size)
```



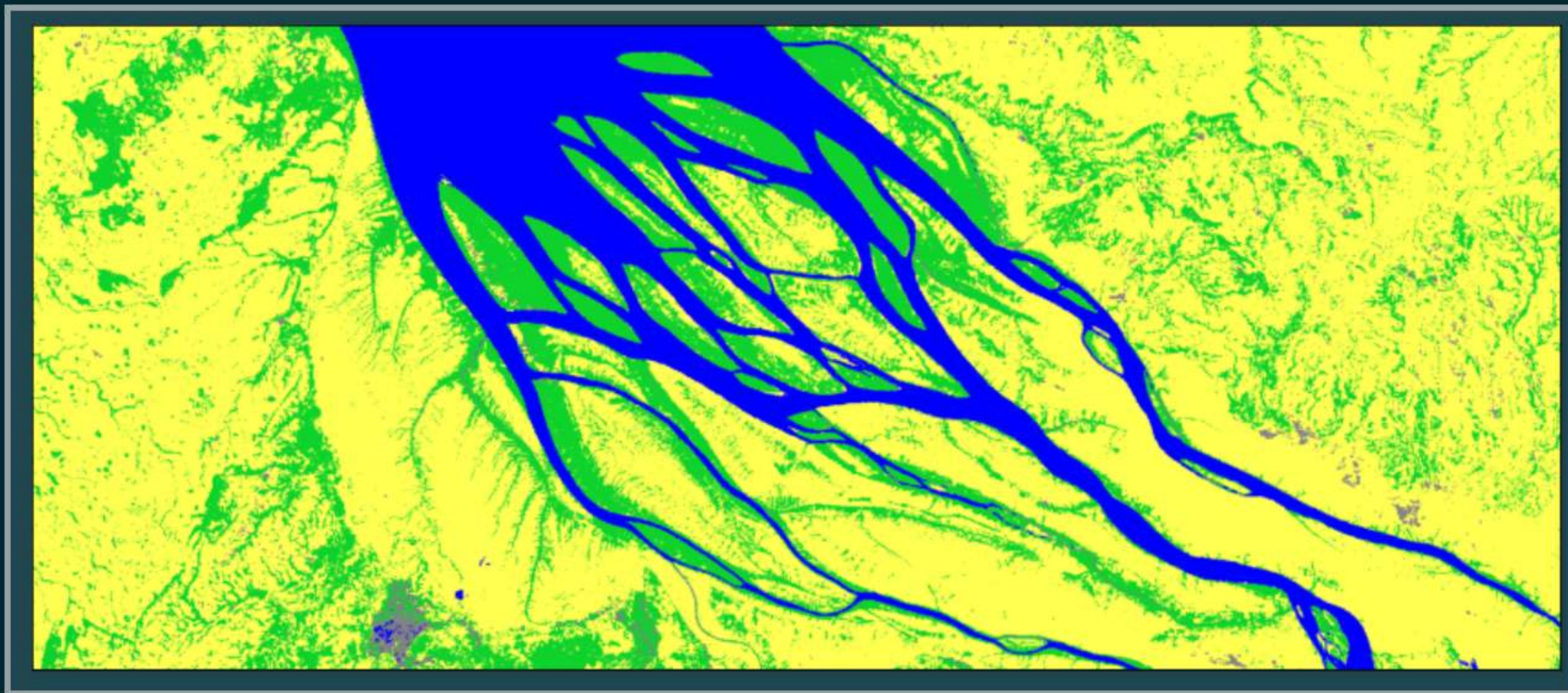
REQUEST

```
request = SentinelHubRequest(  
    evalscript=evalscript,  
    input_data=[  
        SentinelHubRequest.input_data(  
            data_collection=DataCollection.SENTINEL2_L2A,  
            time_interval=('2019-06-01', '2019-7-31'))  
    ],  
    responses=[  
        SentinelHubRequest.output_response(  
            'default', MimeType.TIFF),  
        SentinelHubRequest.output_response(  
            'userdata', MimeType.JSON)],  
    bbox=bbox,  
    size=size)
```

Responses must match the output



RESULT



Water	Vegetation	Bare soils	Other
14.36%	21.79%	62.61%	1.24%

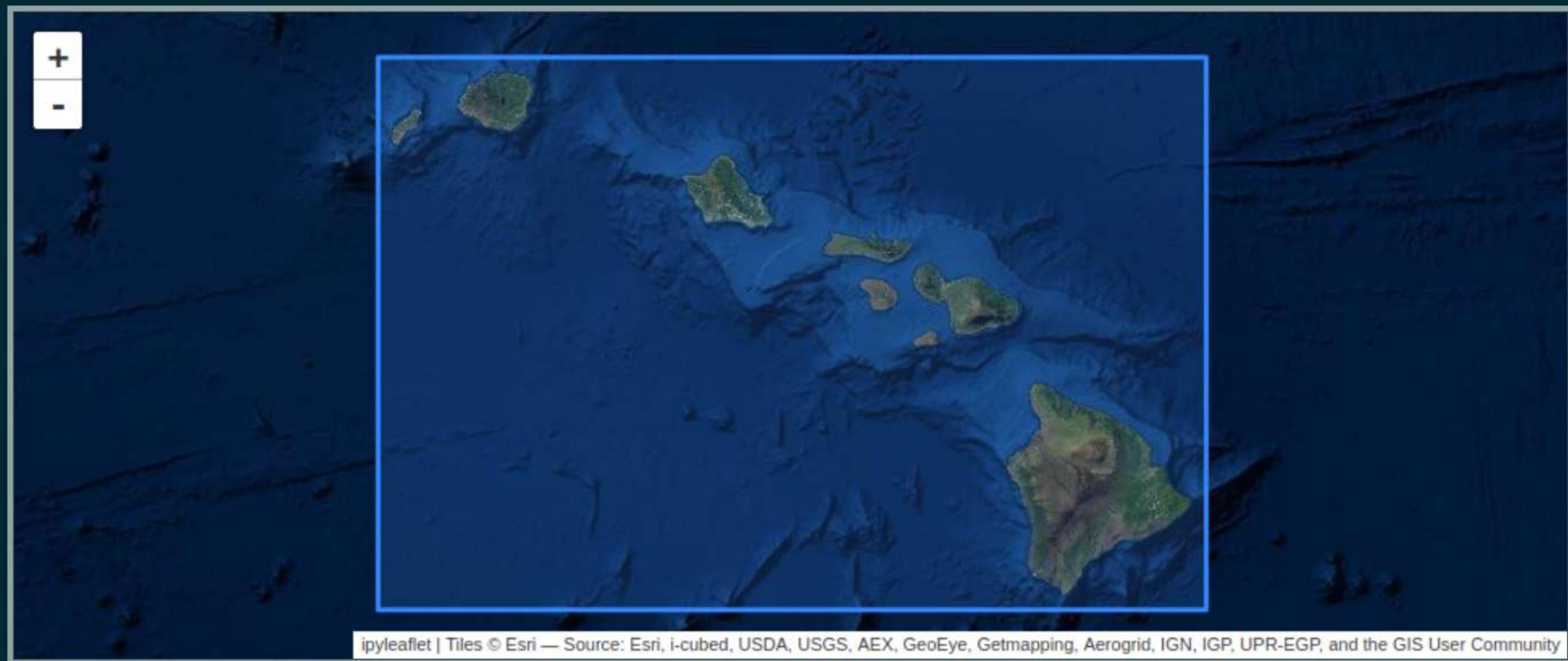


EXAMPLE 8 - LARGE AREA

- area of Hawaii
- single BBox too large
- split into smaller with splitter

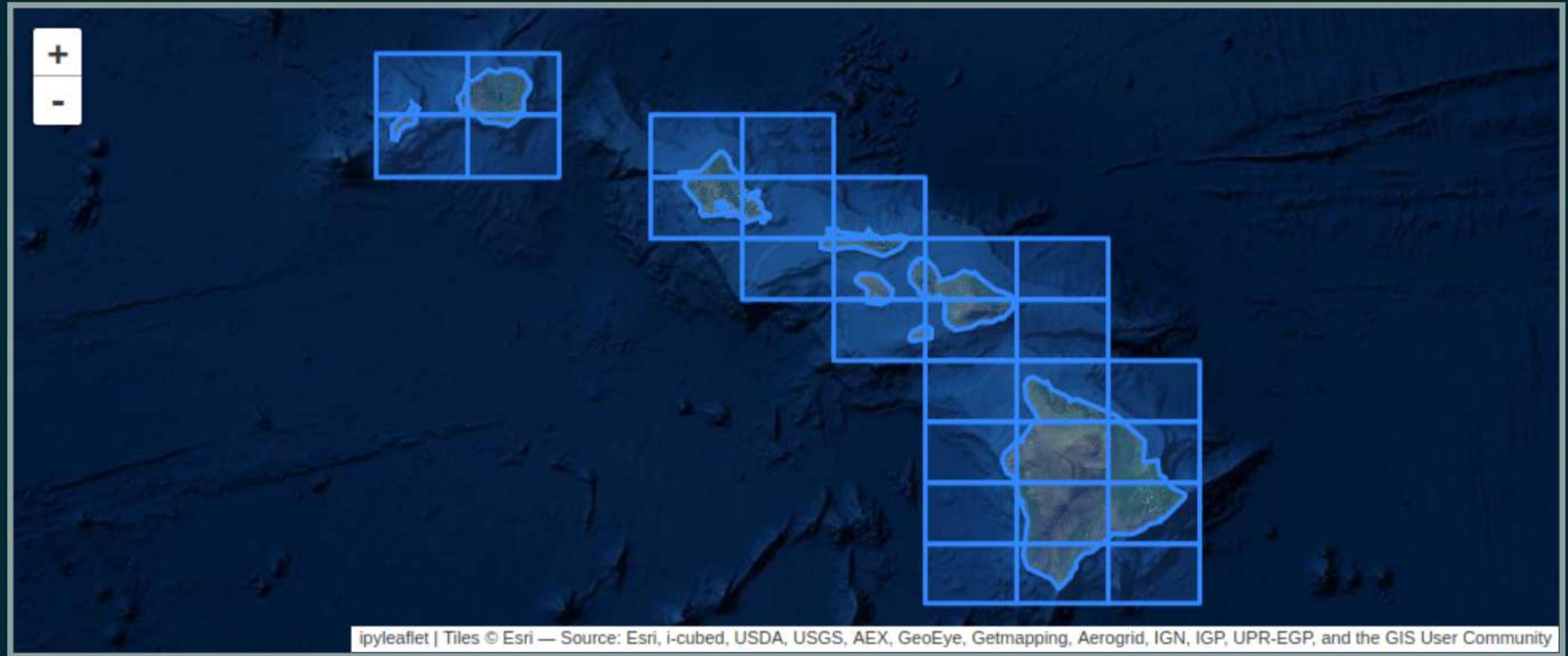


NEW AREA



Not practical and limited to 2500px!
Image size at resolution 240m: (2435, 1648)





Split to smaller boxes using the provided splitter
Each box at 240m res: (271, 179)



HOW TO SPLIT

```
from sentinelhub import BBoxSplitter
bbox_splitter = BBoxSplitter(
    [hawaii_area], CRS.WGS84,
    (9, 9)
)
bbox_list = bbox_splitter.get_bbox_list()

req_list = [req1, req2, ...]
req_list = [req.download_list[0] for r in req_list]
```



HOW TO SPLIT

```
from sentinelhub import BBoxSplitter
bbox_splitter = BBoxSplitter(
    [hawaii_area], CRS.WGS84,
    (9, 9)
)
bbox_list = bbox_splitter.get_bbox_list()

req_list = [req1, req2, ...]
req_list = [req.download_list[0] for r in req_list]
```

Load the splitter



HOW TO SPLIT

```
from sentinelhub import BBoxSplitter
bbox_splitter = BBoxSplitter(
    [hawaii_area], CRS.WGS84,
    (9, 9)
)
bbox_list = bbox_splitter.get_bbox_list()

req_list = [req1, req2, ...]
req_list = [req.download_list[0] for r in req_list]
```

Split in 9x9 pieces



HOW TO SPLIT

```
from sentinelhub import BBoxSplitter
bbox_splitter = BBoxSplitter(
    [hawaii_area], CRS.WGS84,
    (9, 9)
)
bbox_list = bbox_splitter.get_bbox_list()

req_list = [req1, req2, ...]
req_list = [req.download_list[0] for r in req_list]
```

Same as before for each bbox



DOWNLOAD TO DISK

```
return SentinelHubRequest(  
    data_folder='./hawaii',  
    evalscript=evalscript,  
    ...
```



DOWNLOAD TO DISK

```
return SentinelHubRequest(  
    data_folder='./hawaii',  
    evalscript=evalscript,  
    ...
```

Provide a data folder in order to save to disk



DOWNLOAD TO DISK

```
return SentinelHubRequest(  
    data_folder='./hawaii',  
    evalscript=evalscript,  
    ...
```

Merge images using tools from the geographic information systems (GIS) field



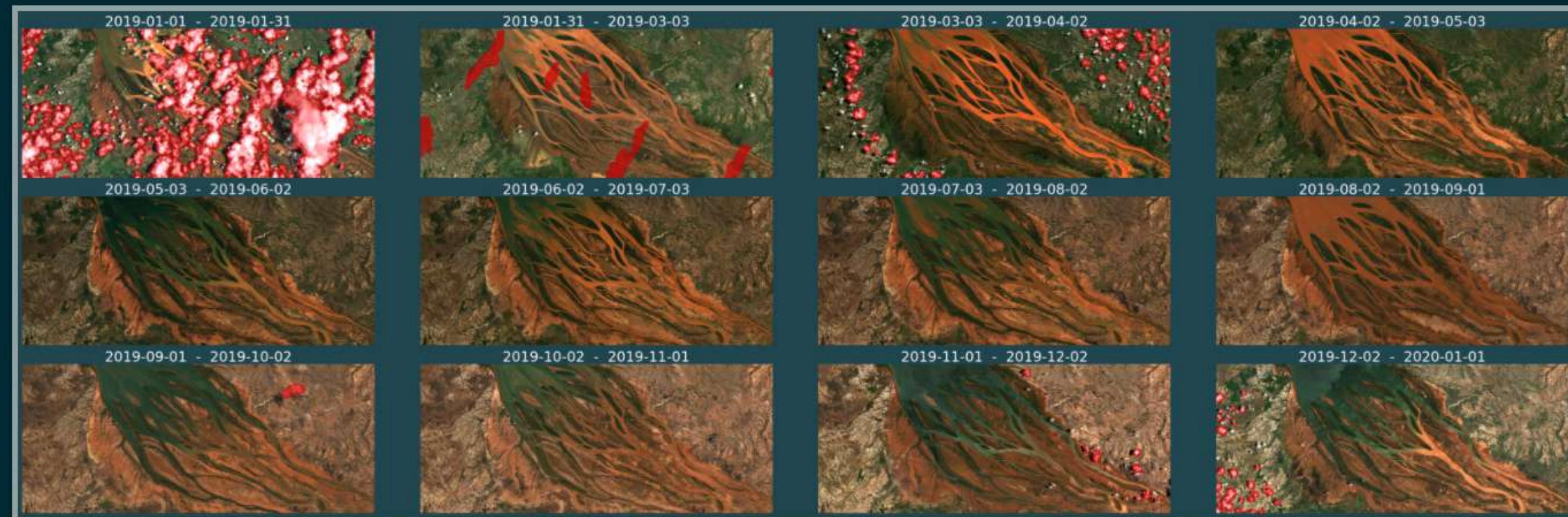


EXAMPLE 9 - STATISTICAL REQUESTS

- newest addition of StatsAPI
- used for obtaining spatial stats of an area
 - max, min, std, ...
- much quicker if you only want numbers



GET CLOUD COVERAGE INFO



EVALSCRIPT

```
// VERSION=3
function setup() {
  return {
    input: ["CLM", "dataMask"],
    output: [
      {id: "clm", bands: ["CLM"]},
      {id: "dataMask", bands: 1}
    ]
  }
}

function evaluatePixel(samples) {
  return {
    clm: [samples.CLM],
    dataMask: [samples.dataMask]
  };
}
```



EVALSCRIPT

```
// VERSION=3
function setup() {
  return {
    input: ["CLM", "dataMask"],
    output: [
      {id: "clm", bands: ["CLM"]},
      {id: "dataMask", bands: 1}
    ]
}
function evaluatePixel(samples) {
  return {
    clm: [samples.CLM],
    dataMask: [samples.dataMask]
  };
}
```

Set explicit names

```
stats_request = SentinelHubStatistical(  
    aggregation=SentinelHubStatistical.aggregation(  
        evalscript=stats_evalscript,  
        time_interval=('2019-01-01', '2020-01-01'),  
        aggregation_interval='P30D',  
        size=(631, 1047)  
),  
    input_data=[  
        SentinelHubStatistical.input_data(  
            DataCollection.SENTINEL2_L2A,  
            mosaicking_order='leastCC')  
    ],  
    bbox=betsiboka_bbox,  
    config=config  
)
```



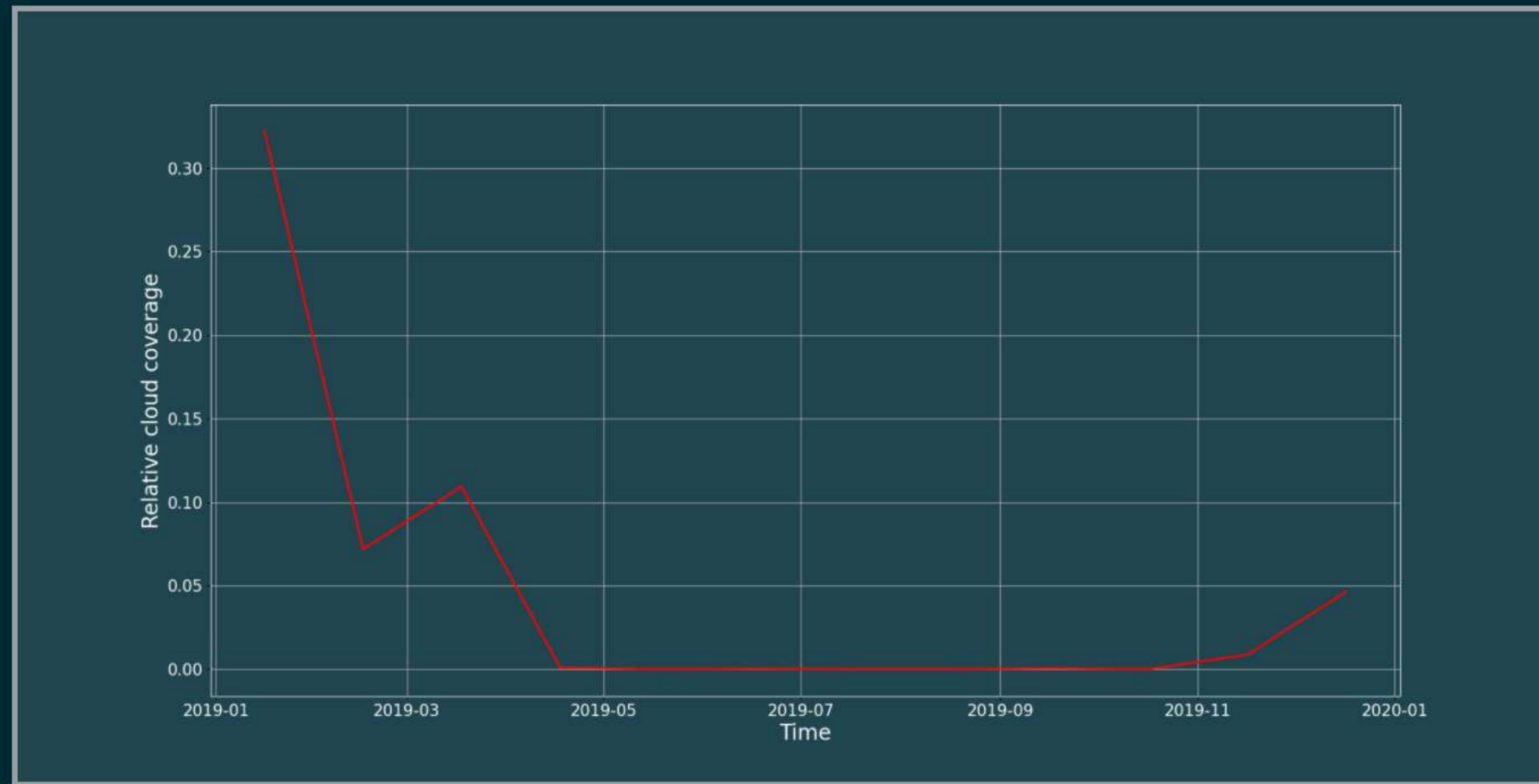
```
stats request = SentinelHubStatistical(  
    aggregation=SentinelHubStatistical.aggregation(  
        evalscript=stats_evalscript,  
        time_interval=('2019-01-01', '2020-01-01'),  
        aggregation_interval='P30D',  
        size=(631, 1047)  
),  
    input_data=[  
        SentinelHubStatistical.input_data(  
            DataCollection.SENTINEL2_L2A,  
            mosaicking_order='leastCC')  
    ],  
    bbox=betsiboka_bbox,  
    config=config  
)
```

Set the stats part

```
stats_request = SentinelHubStatistical(  
    aggregation=SentinelHubStatistical.aggregation(  
        evalscript=stats_evalscript,  
        time_interval=('2019-01-01', '2020-01-01'),  
        aggregation_interval='P30D',  
        size=(631, 1047)  
),  
    input_data=[  
        SentinelHubStatistical.input_data(  
            DataCollection.SENTINEL2_L2A,  
            mosaicking_order='leastCC')  
    ],  
    bbox=betsiboka_bbox,  
    config=config  
)
```

Define the range and intervals for aggregation

RESULT



WRAP UP

- this was just a taste!
- try and see what you can do with all this data
- we'll be waiting on Discord stand-by for technical help
- lots of info provided on our dedicated page

<https://www.sentinel-hub.com/dragonhack2021/>



WRAP UP

- this was just a taste!
- try and see what you can do with all this data
- we'll be waiting on Discord stand-by for technical help
- lots of info provided on our dedicated page

<https://www.sentinel-hub.com/dragonhack2021/>

GOOD LUCK!

