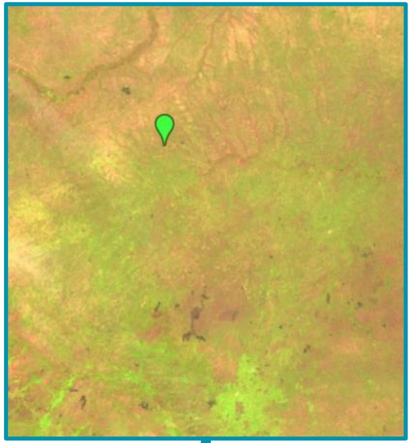
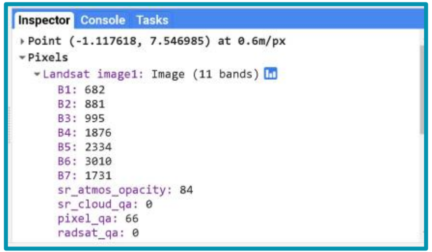
DATA EXPLORATION IN Google Earth Engine

Basic Concepts in GEE

Image

- **An Image** comprises of a two-dimensional array of individual picture elements called pixels arranged in columns and rows
- Each pixel represents an area on the Earth's surface, which has an intensity value, and a location address in the two dimensional image





```
▸ Point (-1.117618, 7.546985) at 0.6m/px
▾ Pixels
  ▾ Landsat image1: Image (11 bands)
      B1: 682
      B2: 881
      B3: 995
      B4: 1876
      B5: 2334
      B6: 3010
      B7: 1731
      sr_atmos_opacity: 84
      sr_cloud_qa: 0
      pixel_qa: 66
      radsat_qa: 0
```

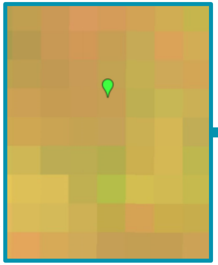
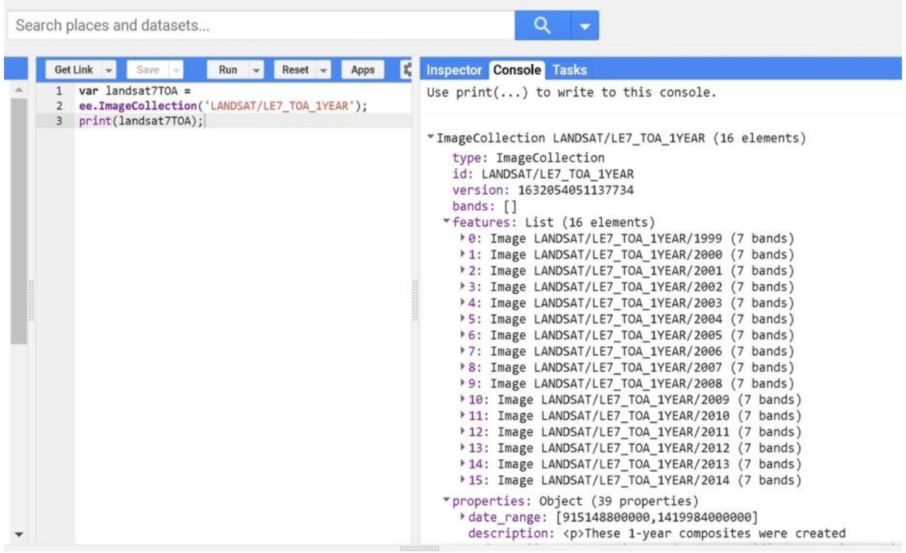


Image Collection

- **An Image Collection** is a stack of images.
- For instance, a collection of all Landsat 7 images in a given time period
- Each image collection has an ID



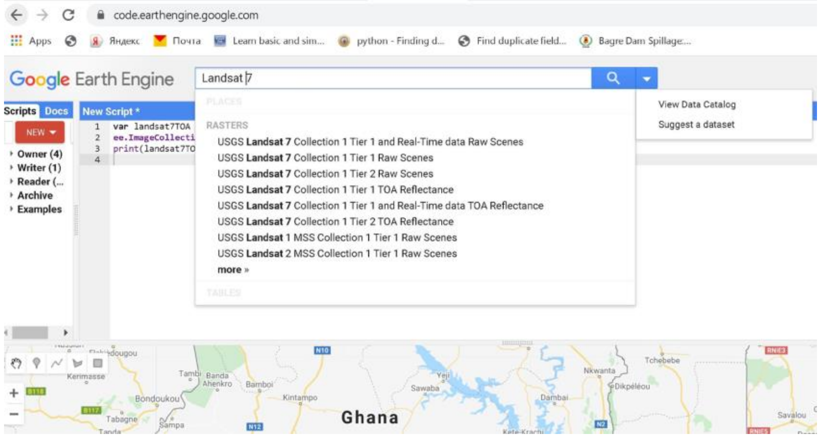
```
var landsat7TOA =
ee.ImageCollection('LANDSAT/LE7_TOA_1YEAR');
print(landsat7TOA);
```

```
Use print(...) to write to this console.

▾ImageCollection LANDSAT/LE7_TOA_1YEAR (16 elements)
   type: ImageCollection
   id: LANDSAT/LE7_TOA_1YEAR
   version: 1632054051137734
   bands: []
  ▾features: List (16 elements)
    ▸0: Image LANDSAT/LE7_TOA_1YEAR/1999 (7 bands)
    ▸1: Image LANDSAT/LE7_TOA_1YEAR/2000 (7 bands)
    ▸2: Image LANDSAT/LE7_TOA_1YEAR/2001 (7 bands)
    ▸3: Image LANDSAT/LE7_TOA_1YEAR/2002 (7 bands)
    ▸4: Image LANDSAT/LE7_TOA_1YEAR/2003 (7 bands)
    ▸5: Image LANDSAT/LE7_TOA_1YEAR/2004 (7 bands)
    ▸6: Image LANDSAT/LE7_TOA_1YEAR/2005 (7 bands)
    ▸7: Image LANDSAT/LE7_TOA_1YEAR/2006 (7 bands)
    ▸8: Image LANDSAT/LE7_TOA_1YEAR/2007 (7 bands)
    ▸9: Image LANDSAT/LE7_TOA_1YEAR/2008 (7 bands)
    ▸10: Image LANDSAT/LE7_TOA_1YEAR/2009 (7 bands)
    ▸11: Image LANDSAT/LE7_TOA_1YEAR/2010 (7 bands)
    ▸12: Image LANDSAT/LE7_TOA_1YEAR/2011 (7 bands)
    ▸13: Image LANDSAT/LE7_TOA_1YEAR/2012 (7 bands)
    ▸14: Image LANDSAT/LE7_TOA_1YEAR/2013 (7 bands)
    ▸15: Image LANDSAT/LE7_TOA_1YEAR/2014 (7 bands)
  ▾properties: Object (39 properties)
    ▸date_range: [915148800000,1419984000000]
     description: <p>These 1-year composites were created
```

Import an Image or Image Collection

- This can be done in two ways:
  - Use the search button and add the image to a defined workspace



```
var landsat7TOA
ee.ImageCollecti
print(landsat7TO
```

RASTERS

- USGS **Landsat 7** Collection 1 Tier 1 and Real-Time data Raw Scenes
- USGS **Landsat 7** Collection 1 Tier 1 Raw Scenes
- USGS **Landsat 7** Collection 1 Tier 2 Raw Scenes
- USGS **Landsat 7** Collection 1 Tier 1 TOA Reflectance
- USGS **Landsat 7** Collection 1 Tier 1 and Real-Time data TOA Reflectance
- USGS **Landsat 7** Collection 1 Tier 2 TOA Reflectance
- USGS **Landsat** 1 MSS Collection 1 Tier 1 Raw Scenes
- USGS **Landsat** 2 MSS Collection 1 Tier 1 Raw Scenes
- **more »**

  - Using the image ID

Var Landsat8=ee.ImageCollection('LANDSAT/LC8_L1T_TOA')

where 'LANDSAT/LC8_L1T_TOA' *is the ID*

Display or image visualization - 1

- Requires visualization parameters
- Uses the Function *Map.addLayer(image,{visualization parameters},'name')* where the curly brackets {} contain the visualization parameters
- Color palette
  - *Example* : palette: ['00FFFF', '0000FF'] or palette: [‘red', ‘green', 'blue'] (used for single-band images only)
- Bands
  - *Example :* 'bands': ['B4', 'B3', 'B2'] (normally a list of three band names to be mapped to RGB
- Image stretch parameters
  - *Example* : min: 0.0, max:0.3 (used to stretch the image for better visualization)

Display or image visualization - 2

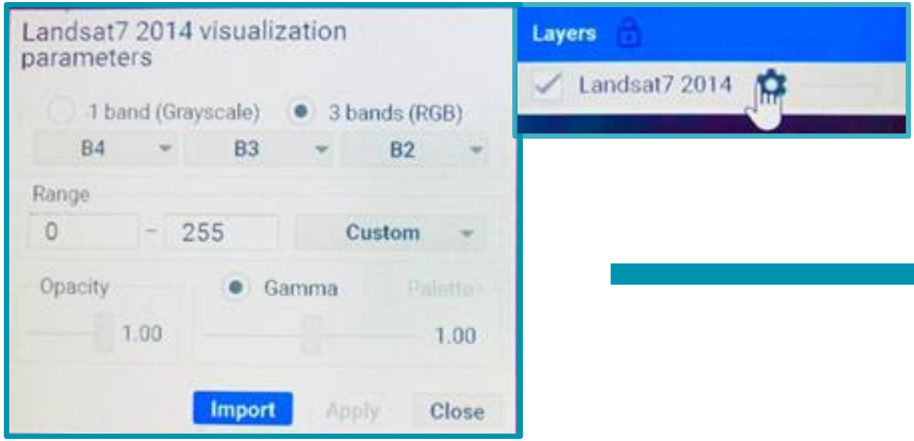
- The function Map.setCenter() is used to sets the viewport to a specific location and zoom level
- *Example* :
  - var landsat7TOA2014 = ee.Image('LANDSAT/LE7_TOA_1YEAR/2014');
  - Map.addLayer(ee.Image('LANDSAT/LE7_TOA_1YEAR/2014'),{'bands':['B4', 'B3','B2'],},'Landsat7 2014');
  - Map.setCenter(-0.411, 6.469,7), where **" -0.411, 6.469**" is location and **7** the zoom level

Image Visualization Parameters

PARAMETER	DESCRIPTION	TYPE
bands	Comma-delimited list of three band names to be mapped to RGB	list
min	Value(s) to map to 0	number or list of three numbers, one for each band
max	Value(s) to map to 255	number or list of three numbers, one for each band
gain	Value(s) by which to multiply each pixel value	number or list of three numbers, one for each band
bias	Value(s) to add to each DN	number or list of three numbers, one for each band
gamma	Gamma correction factor(s)	number or list of three numbers, one for each band
palette	List of CSS-style color strings (single-band images only)	comma-separated list of hex strings
opacity	The opacity of the layer (0.0 is fully transparent and 1.0 is fully opaque)	number
format	Either "jpg" or "png"	string

Display or image visualization - 3

- After displaying the image you can further manipulate the visualization parameters by using the settings of the image layer.



Landsat7 2014 visualization parameters

○ 1 band (Grayscale) ● 3 bands (RGB)

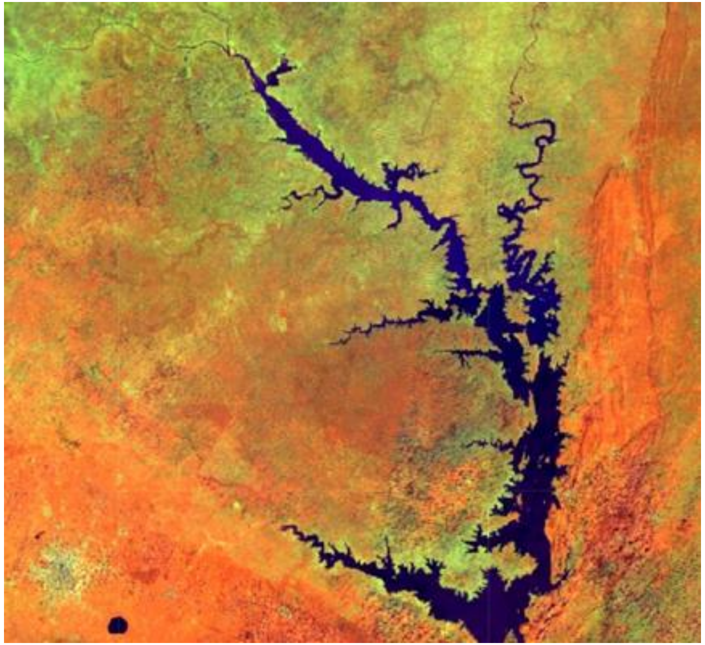
B4 B3 B2

Range

0 – 255 Custom

Opacity ● Gamma ○ Palette

1.00 1.00



Run Script

Visualization of Image/Image Collection

Image Processing - 1

- **Filter by metadata:** Query the image metadata using filters such as ee.Filter.eq(), ee.Filter.lt() etc. You can filter by path/row values, orbit number or cloud cover
  - *Example. var filtered1 = s2.filter(ee.Filter.lt('CLOUDY_PIXEL_PERCENTAGE', 30))*
- **Filter by date:** Select images in a particular date range using filters such as ee.Filter.date()
  - *Example. ee.Filter.date('2019-01-01', '2020-01-01'))*
- **Filter by location: S**ubset an image with a bounding box, or any user-defined geometry using the ee.Filter.bounds()
  - *Example. .filter(ee.Filter.bounds(geometry))*
- **Filter by cloud PERCENTAGE:** This function filters out images with less clouds using the function ee.Filter.lt()
  - *Example. .filter(ee.Filter.lt('CLOUDY_PIXEL_PERCENTAGE', 30))*

Image Processing - 2

- Concatenating filters with the dot (.) notation
  - *Example*

```
//importing geometry
var geometry= ee.Geometry.point([-3.02,6.67]);
// importing the image collection
var s2 = ee.ImageCollection("COPERNICUS/S2");
// applying the filtering functions
var filtered = s2.filter(ee.Filter.lt('CLOUDY_PIXEL_PERCENTAGE', 30))
   .filter(ee.Filter.date('2019-01-01', '2020-01-01'))
   .filter(ee.Filter.bounds(geometry))

print(filtered);
```

Image Processing - 3

- **Mosaicking:** the Function .mosaic() is used on a ImageCollection to create a image mosaics

- **Image Compositing:** application of reduce Functions

  - *Example:*

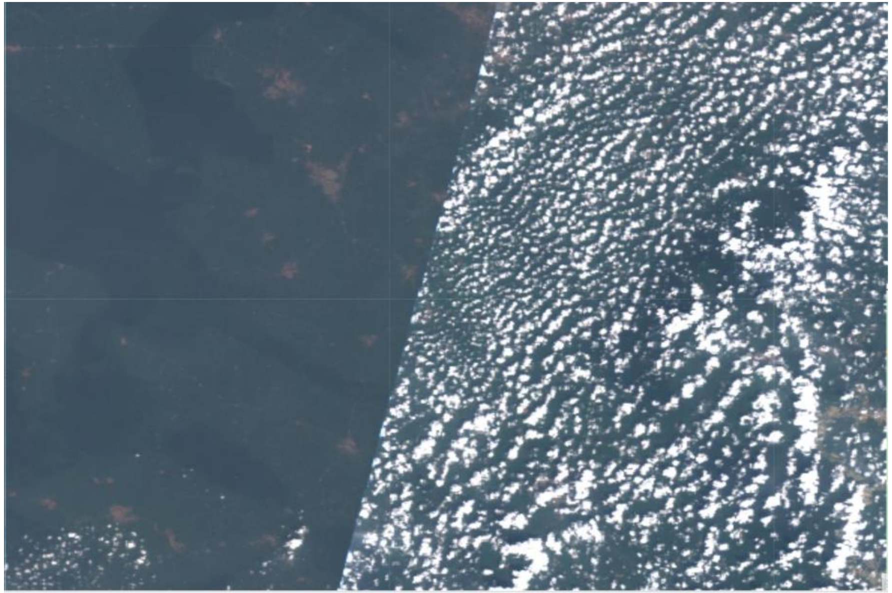
```
var filtered = s2.filter(ee.Filter.lt('CLOUDY_PIXEL_PERCENTAGE', 30))
.filter(ee.Filter.date('2019-01-01', '2020-01-01'))
 .filter(ee.Filter.bounds(geometry))
print(filtered);

// mosaicking the image collection
var mosaic = filtered.mosaic()
 // Compositing  using the median reducer
var medianComposite = filtered.median();
```

Compositing and Mosaicking

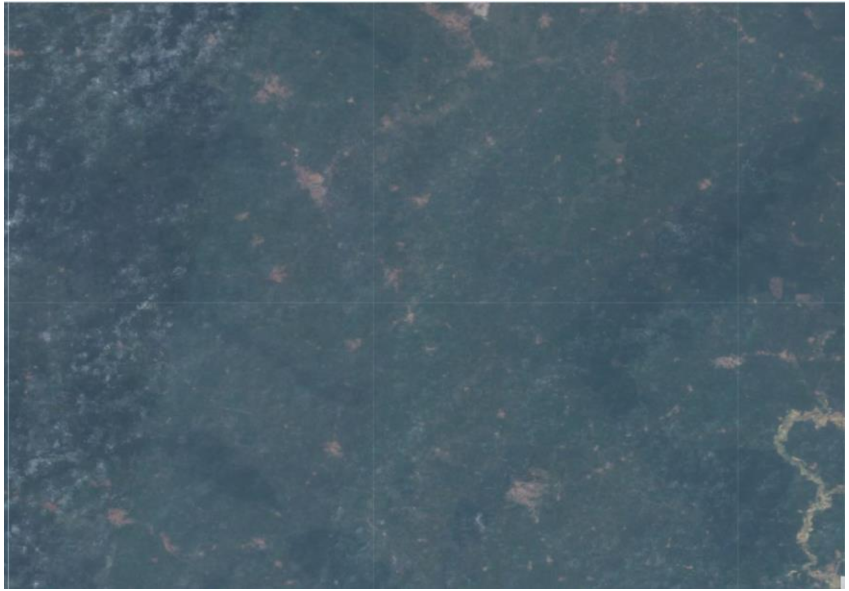
- In general, compositing refers to the process of combining spatially overlapping images into a single image based on an aggregation function.
- Mosaicking refers to the process of spatially assembling image datasets to produce a spatially continuous image.
- In Earth Engine, these terms are used interchangeably, though both compositing and mosaicking are supported.

- Visual Example of using the mosaic and the composite function on the same area and same date range



Mosaic

vrs



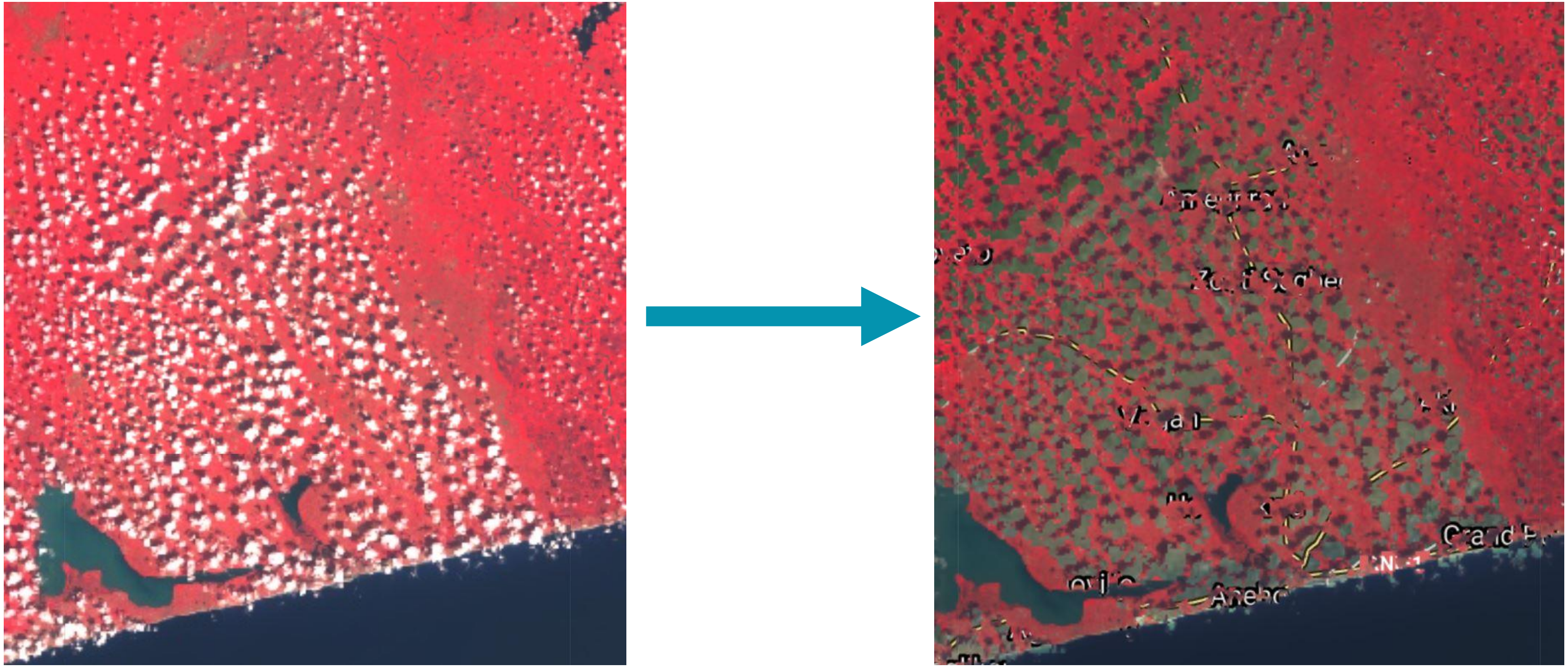
Composite

Run Script

Mosaicking and compositing Image Collection

Mosaicking Images

Cloud Masking



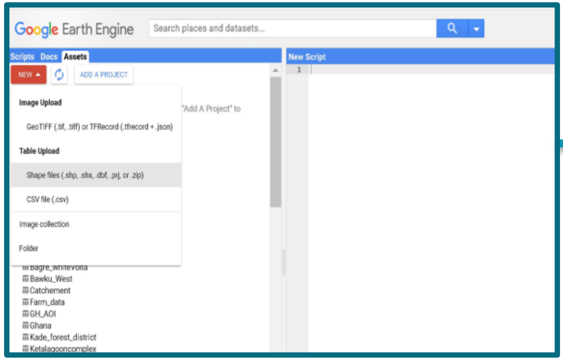
Cloud Masking

Feature and Feature Collection

- **A feature** is an object that stores its geographical representation as a line, points and polygon

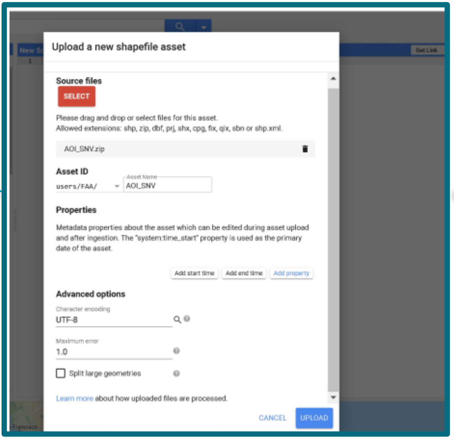
- **A FeatureCollection** is a group of related features which enables additional operations on the entire set such as filtering, merging, clipping

Importing a Feature/Feature Collection



Step1

click assets then click new



Please drag and drop or select files for this asset.
Allowed extensions: shp, zip, dbf, prj, shx, cpg, fix, qix, sbn or shp.xml.

AOI_SNV.zip

Asset ID: users/FAA/ Asset Name: AOI_SNV

Metadata properties about the asset which can be edited during asset upload and after ingestion. The "system:time_start" property is used as the primary date of the asset.

Character encoding: UTF-8

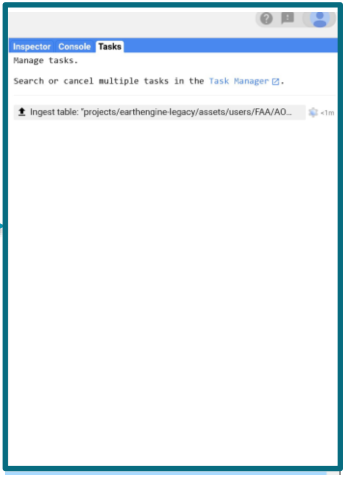
Maximum error: 1.0

□ Split large geometries

Learn more about how uploaded files are processed.

Step2

Upload the feature



Manage tasks.

Search or cancel multiple tasks in the Task Manager.

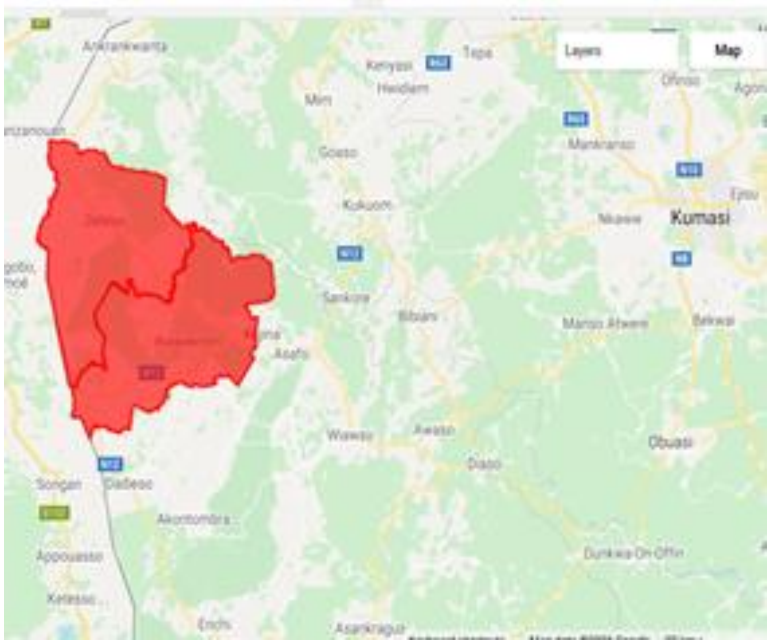
Ingest table: "projects/earthengine-legacy/assets/users/FAA/AO... <1m

Step3

Check uploading in the task bar

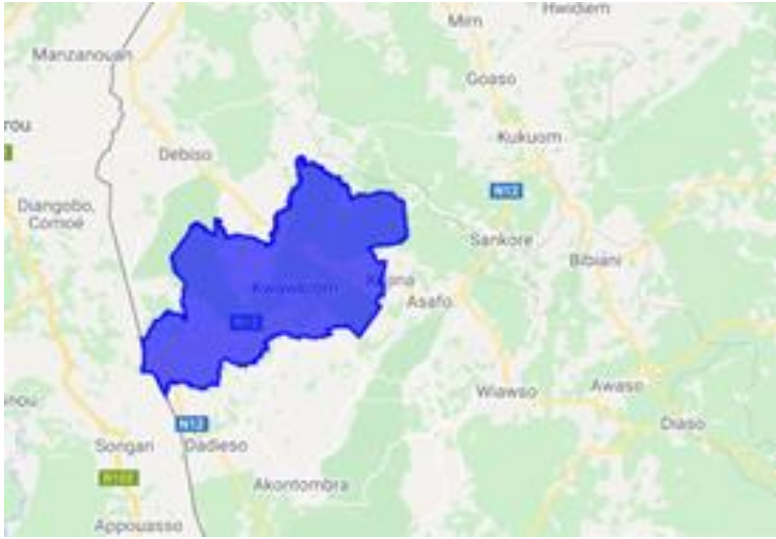
Display or visualizing Features

- Display polygon feature
  - *Example*
  - var aoi = ee.FeatureCollection(users/eopokukwarteng/District__Boundary/);
  - Map.addLayer(aoi, {'color': 'red'}, ' Districts of interest')
- {} contains the visualization parameters



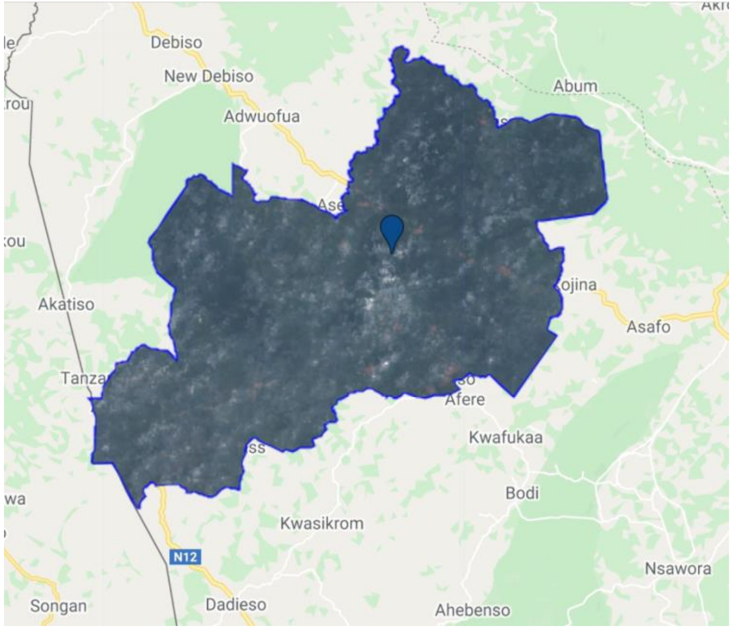
Query or filter Features

- **Filter specific district polygon**
  - *Example*
  - //Filter Juabeso district from the districts of Interest
  - var Juabeso = aoi.filter(ee.Filter.eq('DISTRICT', 'JUABESO')
  - Map.addLayer(Juabeso, {'color': 'blue'}, ' Juabeso')



Clip Image/ImageCollection

- Clip an image / image collection quried District of Interest
  - *Example*
  - // Clipping Function
  - var ClipImage =ImageCollection.clip(aoi);
  - var ClipImage=medianComposite.clip(Juabeso);



Run Script

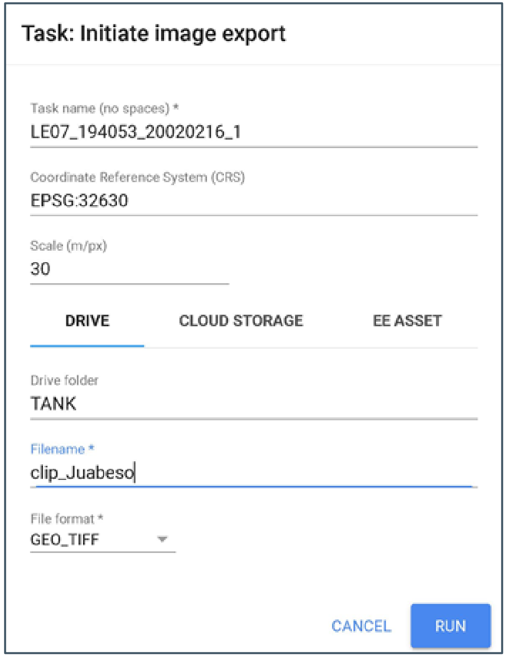
Analysis on Feature/Feature Collection

Export an Image Collection

- Images can be exported as GeoTiFF
  - *Example*
  - // Export Image Collection to GeoTIFF
  - Export.image.toDrive({

```
  image:clips.select('B2','B3','B4','B5','B6','B7'), // image bands to be exported
  description: 'LE08_2017', // name of Image being Exported
  region: table, // area of Interest
  folder: 'TANK',  //folder created in your google drive
  scale: 30,  //spatial resolution of output image
  crs:'EPSG:32630', // coordinate System
  maxPixels:1017604292451  //maximum allowed pixels to export
  });
```

- Run task to export image to local folder or cloud storage



Task: Initiate image export

Task name (no spaces) *
LE07_194053_20020216_1

Coordinate Reference System (CRS)
EPSG:32630

Scale (m/px)
30

Drive folder
TANK

Filename *
clip_Juabeso

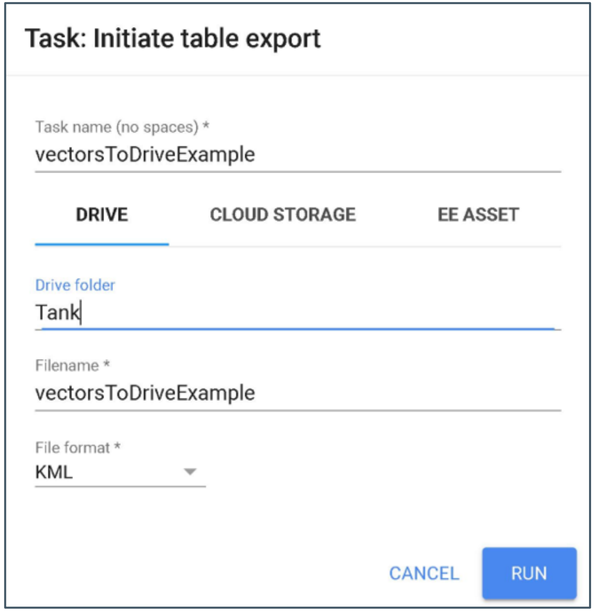
File format *
GEO_TIFF

Export a Feature Collection

- Features can be exported as KML or SHP
  - *Example*
  - // Export FeatureCollection to a KML file
  - Export.table.toDrive({

```
                    collection: features,
                    description:'vectorsToDriveExample',
                     fileFormat: 'KML'
      });
```

- Run task to export feature collection to local folder



Task: Initiate table export

Task name (no spaces) *
vectorsToDriveExample

Drive folder
Tank

Filename *
vectorsToDriveExample

File format *
KML

Run Script

Exporting Image and Feature Collection

Hands-on Exercises

Exercise 1

- Import Protected Areas into GEE
- Filter/ Query Protected Areas in Bono and Bono East Region
- Export the filtered Protected Areas as a shapefile

Exercise 2

- Import Regional Boundary into GEE
- Filter/ Query Bono Region
- Create an Image Collection for Sentinel 2 for 2020 covering Bono Region the area of interest
- Clip the Image Collection with the filtered Regional Boundary(Bono)
- Export the Clipped Image to GeoTIFF

Thank You