Lecture 01

GEE Introduction:

setup, datasets, image visualization

2025-10-03

Sébastien Valade

VNIVERSIDAD NACIONAL
AVFNOMA DE
MEXICO

- **Google Earth Engine** (GEE) is a cloud-based computing platform for processing satellite imagery and other geospatial datasets.

- Provides access to:
  - large database of satellite imagery (including NASA, USGS, ESA, and other satellite missions)
  - large computational power needed to analyze those images

- Provides API (Application Programming Interfaces) for making requests to the servers in:
  - JavaScript ⇒ Earth Engine Code Editor
  - Python ⇒ Google Colaboratory

- **Google Earth Engine** (GEE) is a cloud-based computing platform for processing satellite imagery and other geospatial datasets.

- Provides access to:
  - large database of satellite imagery (including NASA, USGS, ESA, and other satellite missions)
  - large computational power needed to analyze those images

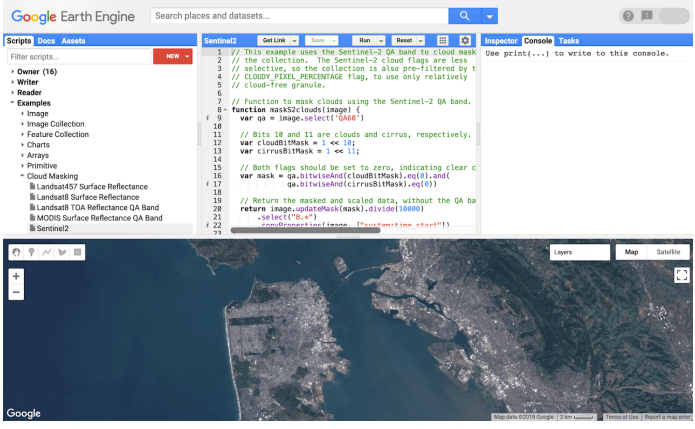
- Provides API (Application Programming Interfaces) for making requests to the servers in:
  - JavaScript ⇒ Earth Engine Code Editor
  - Python ⇒ Google Colaboratory

- **Google Earth Engine** (GEE) is a cloud-based computing platform for processing satellite imagery and other geospatial datasets.

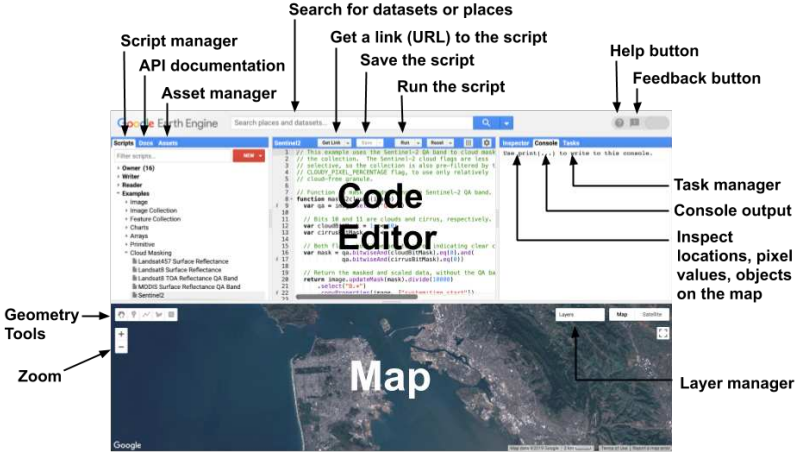
- Provides access to:
  - large database of satellite imagery (including NASA, USGS, ESA, and other satellite missions)
  - large computational power needed to analyze those images

- Provides API (Application Programming Interfaces) for making requests to the servers in:
  - JavaScript ⇒ Earth Engine Code Editor
  - Python ⇒ Google Colaboratory

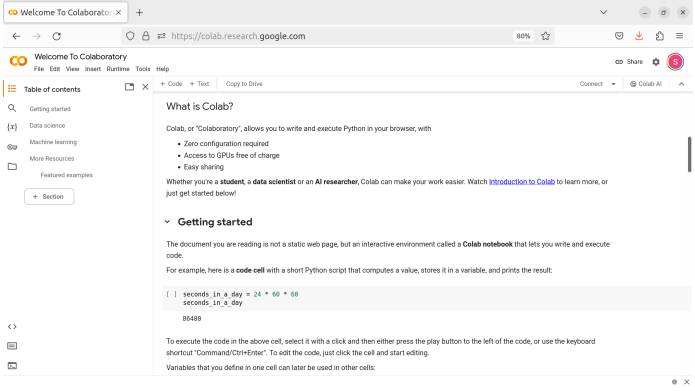
1. Earth Engine Code Editor (JavaScript API)
   ⇒ free web-based IDE *(Integrated Development Environment)* using the JavaScript API



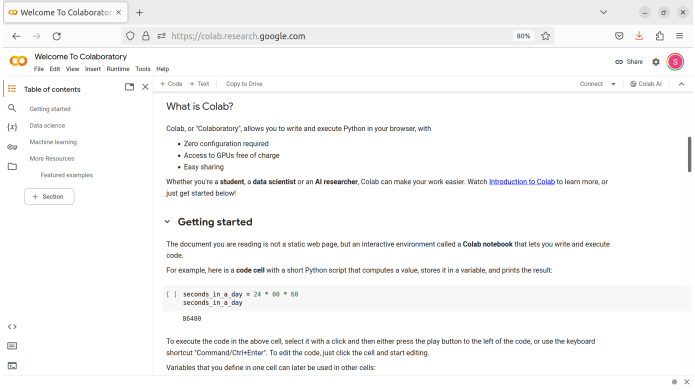
1. Earth Engine Code Editor (JavaScript API)
   ⇒ free web-based IDE *(Integrated Development Environment)* using the JavaScript API



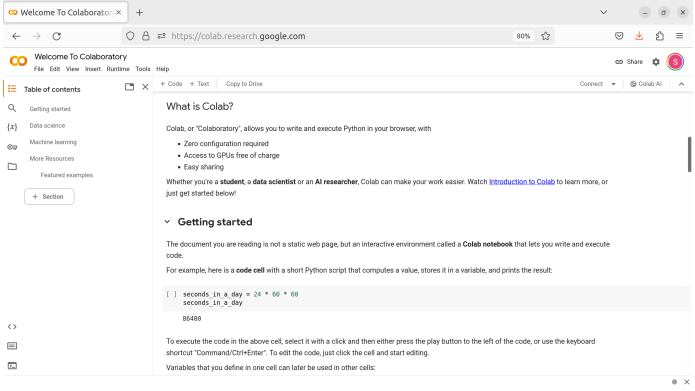
2. Google Colaboratory (Python API)
   ⇒ free cloud-based Jupyter notebook environment for writing and executing Python code
   ⇒ avoids the need to set up a local development environment, i.e. software (libraries) & hardware (GPU)
   ⇒ provides access to GEE Python API, free GPU and TPU resources, enabling users to perform computationally intensive tasks



2. Google Colaboratory (Python API)
   ⇒ free cloud-based Jupyter notebook environment for writing and executing Python code
   ⇒ avoids the need to set up a local development environment, i.e. software (libraries) & hardware (GPU)
   ⇒ provides access to GEE Python API, free GPU and TPU resources, enabling users to perform computationally intensive tasks



2. Google Colaboratory (Python API)
   ⇒ free cloud-based Jupyter notebook environment for writing and executing Python code
   ⇒ avoids the need to set up a local development environment, i.e. software (libraries) & hardware (GPU)
   ⇒ provides access to GEE Python API, free GPU and TPU resources, enabling users to perform computationally intensive tasks



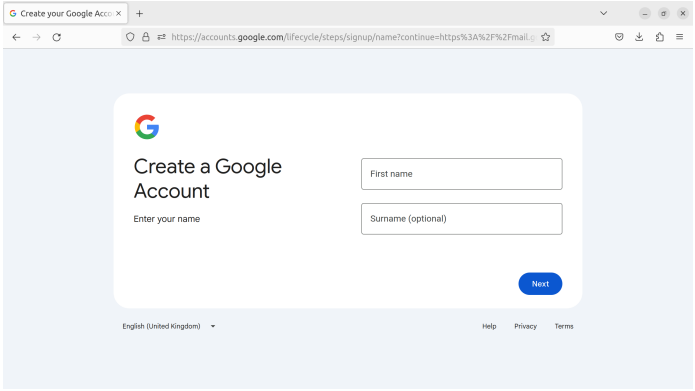
Nota Bene

The steps required by Google to access and use GEE APIs are regularly evolving.

$\Rightarrow$ the steps described are those required as of September-2025

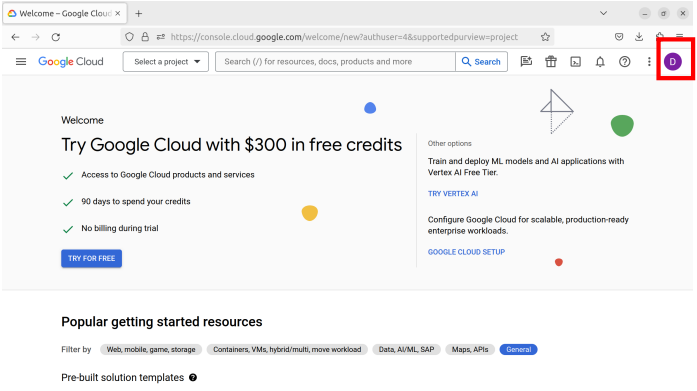
$\Rightarrow$ visit the Earth Engine access guidelines for the most up-to-date information

1. Create a Google account (if you have one, skip this step)



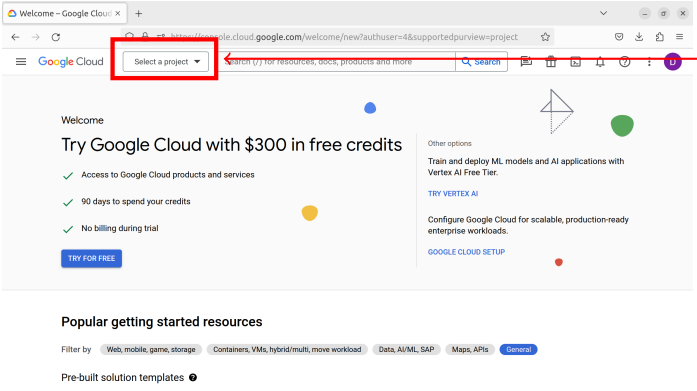
2. Create a Google Cloud project & enable GEE API

2.1 Access your account's Google Cloud Console



2. Create a Google Cloud project & enable GEE API

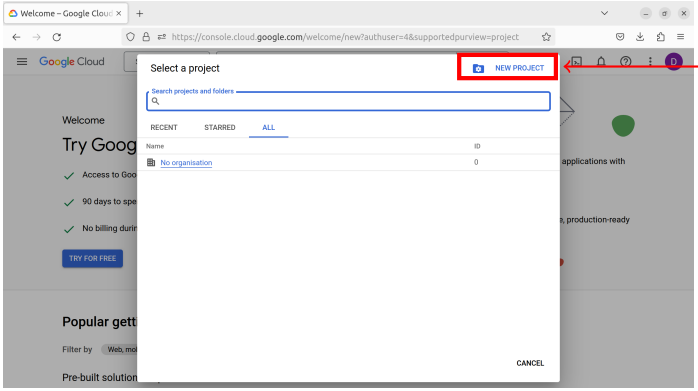
2.2 Create a new project in your Google Cloud Console



2. select a project

2. Create a Google Cloud project & enable GEE API

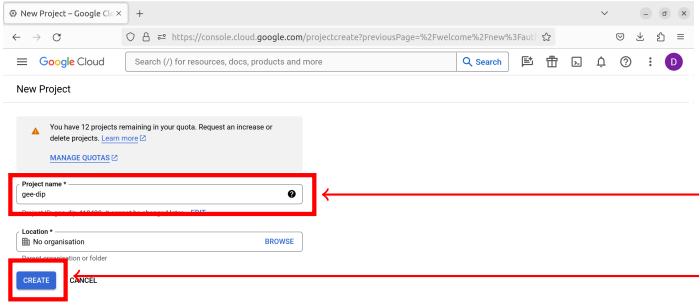
2.2 Create a new project in your Google Cloud Console



3. create new project

2. Create a Google Cloud project & enable GEE API

2.2 Create a new project in your Google Cloud Console

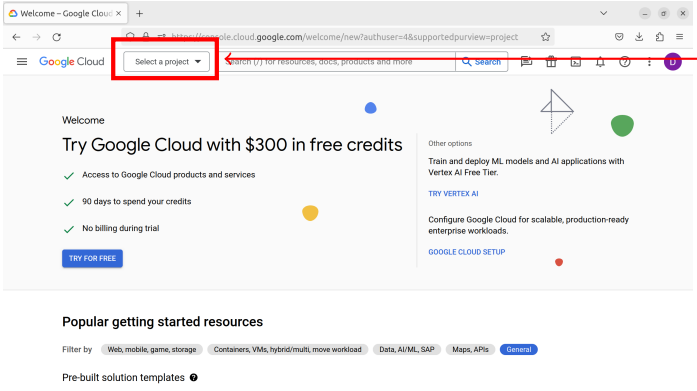


4. select project name

5. create project

2. Create a Google Cloud project & enable GEE API

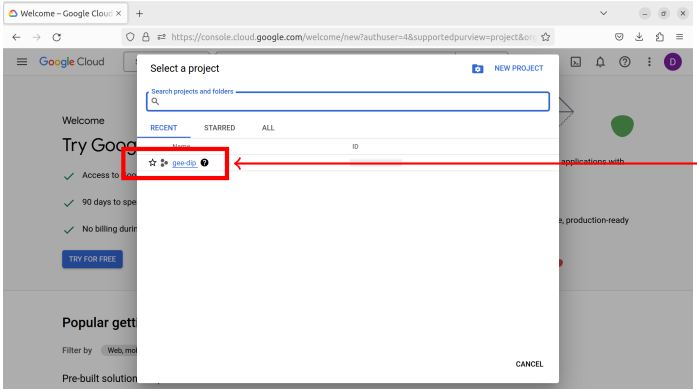
2.3 Enable GEE API in the newly created project



6. select the project

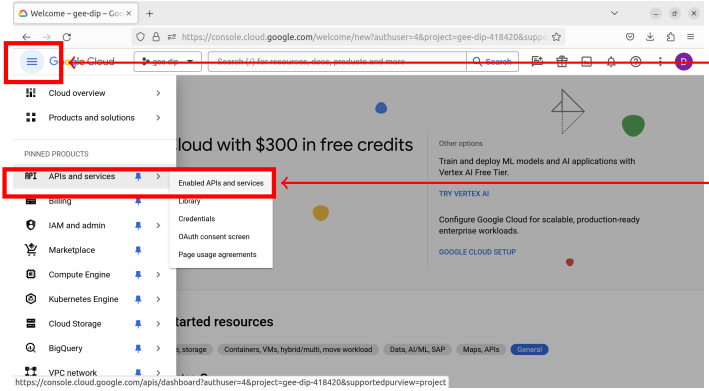
2. Create a Google Cloud project & enable GEE API

2.3 Enable GEE API in the newly created project



2. Create a Google Cloud project & enable GEE API

2.3 Enable GEE API in the newly created project

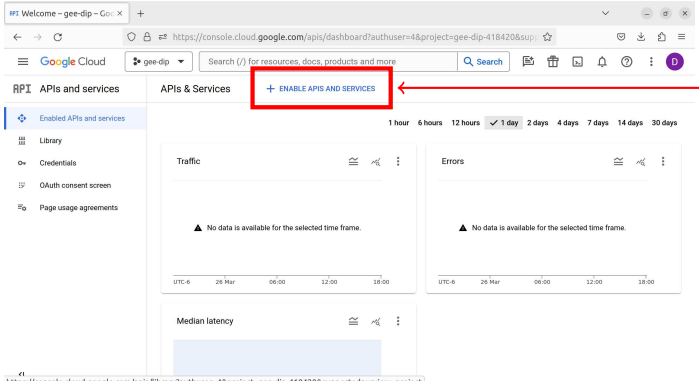


8. select navigation menu

9. “Enable API and services”

2. Create a Google Cloud project & enable GEE API

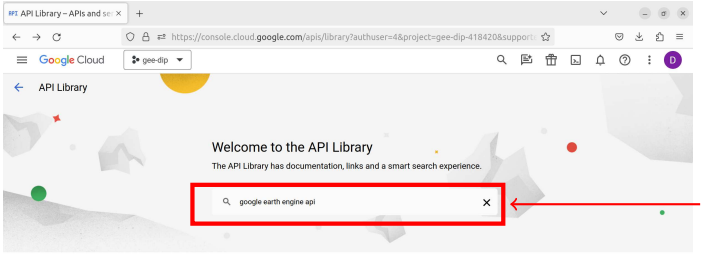
2.3 Enable GEE API in the newly created project



10. click here

2. Create a Google Cloud project & enable GEE API

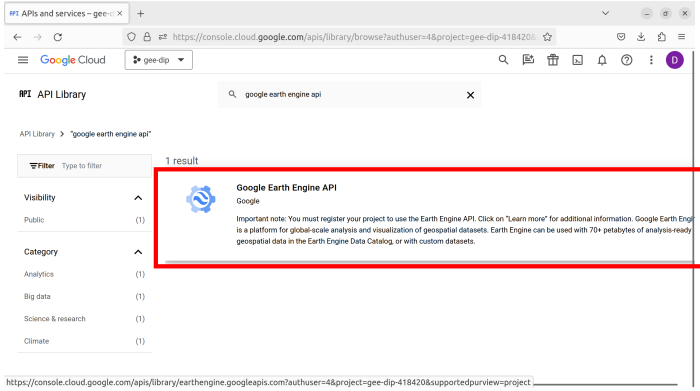
2.3 Enable GEE API in the newly created project



11. search/select “Google Earth Engine API”

2. Create a Google Cloud project & enable GEE API

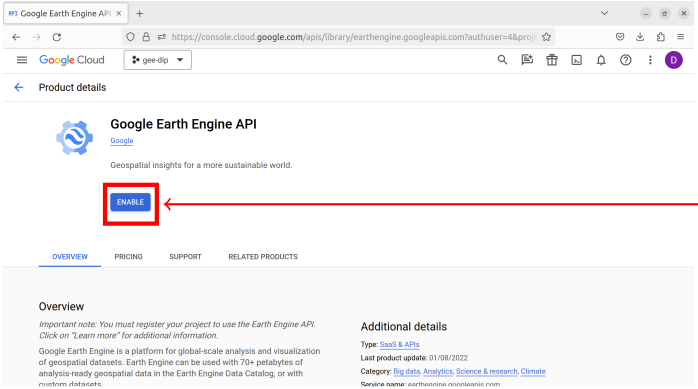
2.3 Configure the newly created project



12. click here

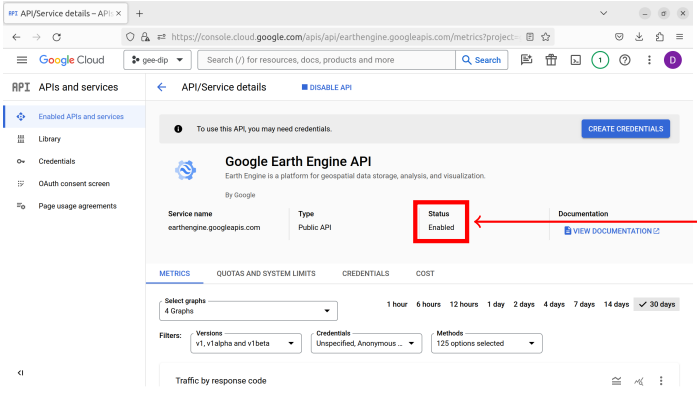
2. Create a Google Cloud project & enable GEE API

2.3 Configure the newly created project



2. Create a Google Cloud project & enable GEE API

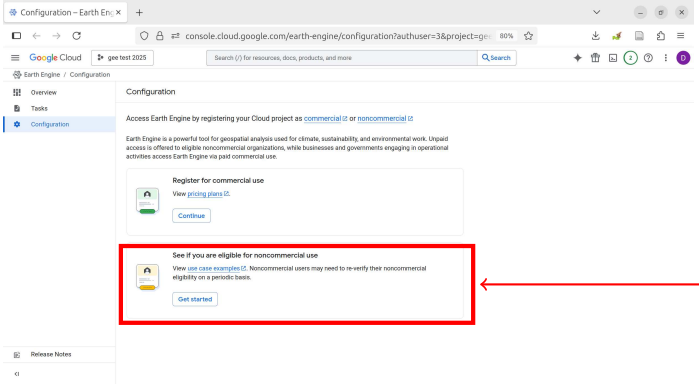
2.3 Enable GEE API in the newly created project



GEE API is now enabled

3. Register Google Cloud project project for use with GEE

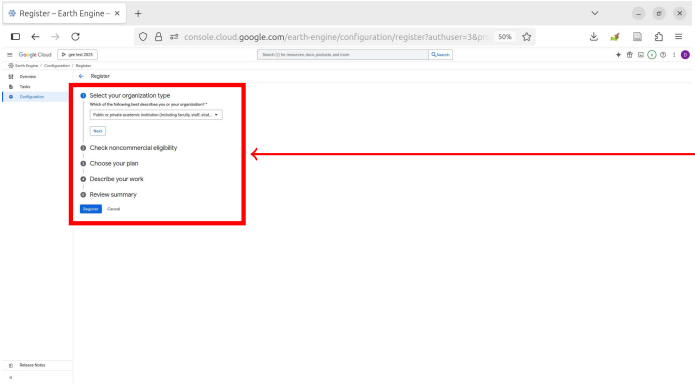
3.1 Access register page at https://code.earthengine.google.com/register



1. clic to register a noncommercial project

3. Register Google Cloud project project for use with GEE

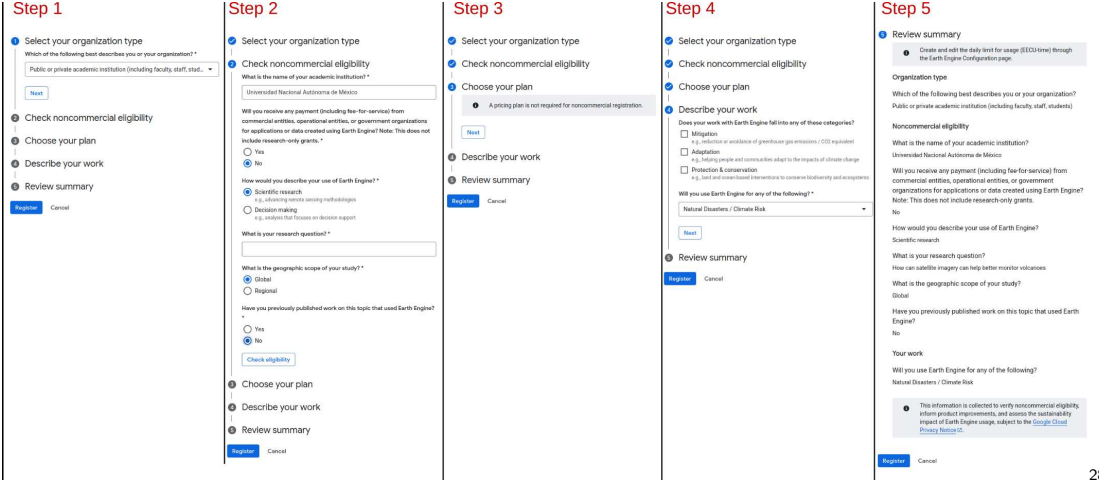
3.1 Register project



2. Fill in questionaire steps 1-5

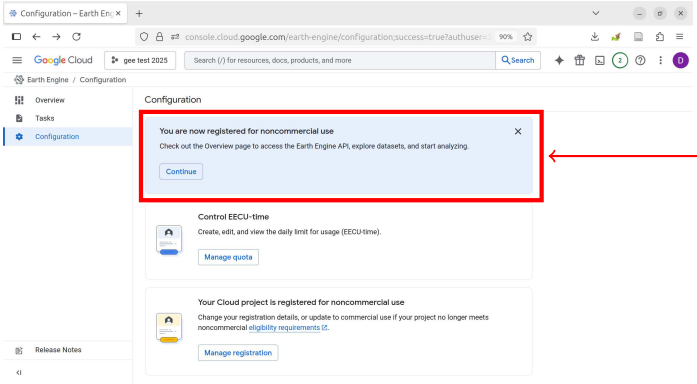
3. Register Google Cloud project project for use with GEE

3.1 Register project



3. Register Google Cloud project project for use with GEE

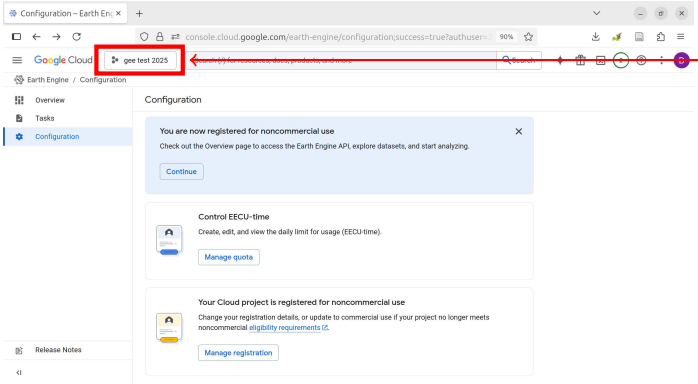
3.2 Register project



3. register successful

3. Register Google Cloud project project for use with GEE

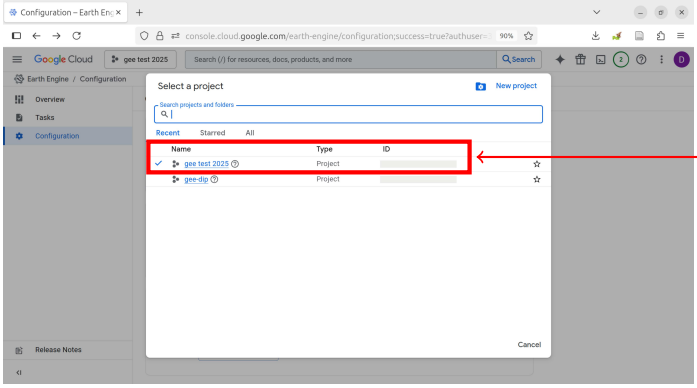
3.2 Register project



4. check project id

3. Register Google Cloud project project for use with GEE

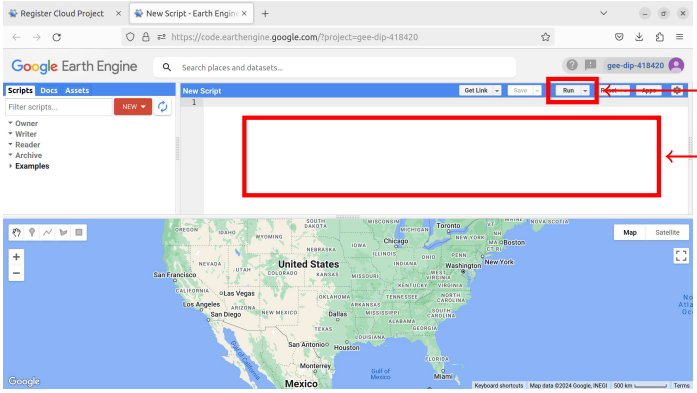
3.1 Register project



Project info:
- **project ID**
- project Name

4. (Run GEE in JavaScript IDE) → *in case you want to try*

   4.1 Access GEE JavaScript IDE at https://code.earthengine.google.com/

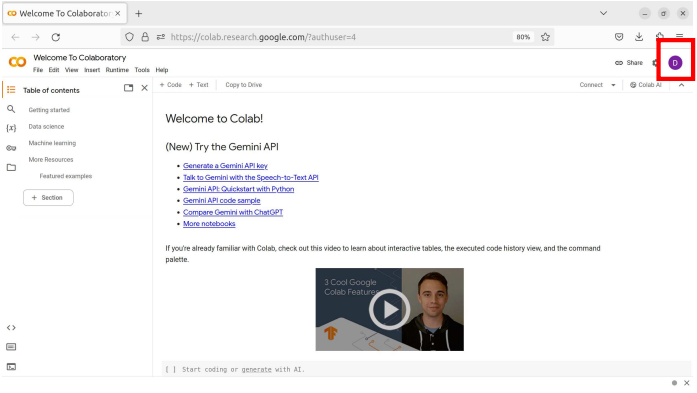


2. clic “Run” to execute code

1. type code in JavaScript
⇒ ex: official tutorials

5. Run GEE in Colab

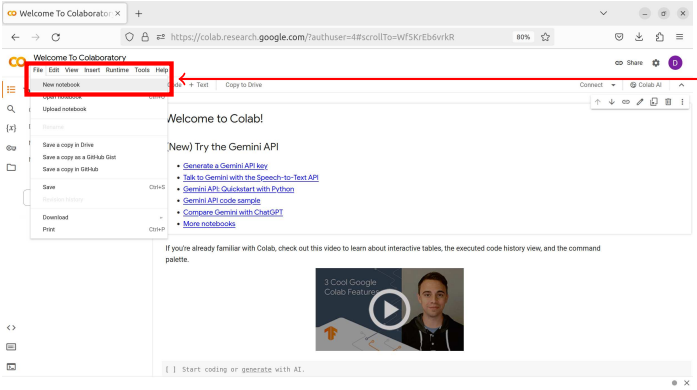
5.1 Access Google Colaboratory at https://colab.research.google.com/



1. select your account here

4. Access GEE in Colab

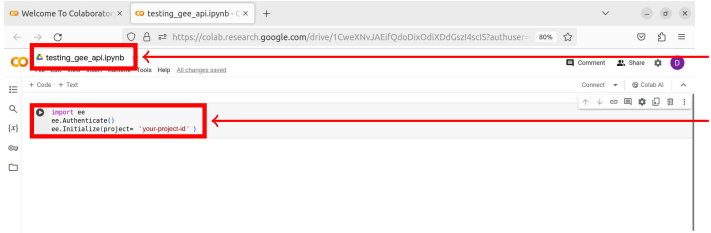
4.2 Create new notebook



2. File - New notebook

4. Access GEE in Colab

4.3 Import ee library & initialize with **project-id** (in which GEE API was enabled)



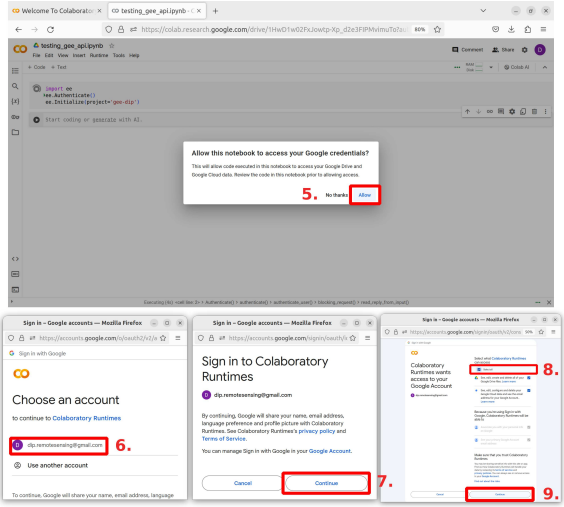
3. rename notebook (optional)

4. import Earth Engine lib (ee), initialize using **project-id** (NOT project-name!), and execute cell

```
import ee
ee.Authenticate()
ee.Initialize(project='your-project-id')  # <--- use project-id (NOT project-name!)
```

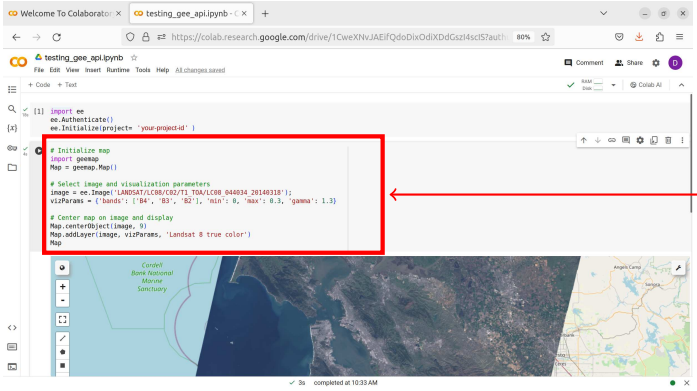
4. Access GEE in Colab

4.3 Execute cell & give authorizations in pop-up windows



4. Access GEE in Colab

4.4 Start coding with GEE in Colab !



10. start coding!
(ex: Tutorial intro-to-python-api)

GEE's public data archive includes >40 years of **satellite imagery** expanded daily:

1. **Landsat** collections
   ⇒ NASA/USGS Program, since 1972
   ⇒ 9 generation of satellites (polar-orbiting):
   - **Landsat-1** (1972) - **Landsat-3** (1978): optical & infrared imaging (VIS/NIR)
   - **Landsat-4** (1982) - **Landsat-9** (2021): optical & infrared imaging (VIS/NIR/SWIR/TIR)

   ⇒ GEE archive includes:

   - Landsat 1-5 (1972–1999) Sensor: MSS (Multispectral Scanner)
   - Landsat 4 (1982–1993) Sensor: TM (Thematic Mapper)
   - Landsat 5 (1984–2012) Sensor: TM (Thematic Mapper)
   - Landsat 7 (1999–2021) Sensor: ETM+ (Enhanced Thematic Mapper Plus)
   - Landsat 8 (2013–Present) Sensor: OLI/TIRS (Op. Land Imager / Therm. Infrared Sensor)
   - Landsat 9 (2021–Present) Sensor: OLI/TIRS (Op. Land Imager / Therm. Infrared Sensor)

GEE's public data archive includes >40 years of **satellite imagery** expanded daily:

1. **Landsat** collections
   ⇒ NASA/USGS Program, since 1972
   ⇒ 9 generation of satellites (polar-orbiting):
   - **Landsat-1** (1972) - **Landsat-3** (1978): optical & infrared imaging (VIS/NIR)
   - **Landsat-4** (1982) - **Landsat-9** (2021): optical & infrared imaging (VIS/NIR/SWIR/TIR)

   ⇒ GEE archive includes:

- Landsat 1-5	(1972–1999)	Sensor: MSS (Multispectral Scanner)
- Landsat 4	(1982–1993)	Sensor: TM (Thematic Mapper)
- Landsat 5	(1984–2012)	Sensor: TM (Thematic Mapper)
- Landsat 7	(1999–2021)	Sensor: ETM+ (Enhanced Thematic Mapper Plus)
- Landsat 8	(2013–Present)	Sensor: OLI/TIRS (Op. Land Imager / Therm. Infrared Sensor)
- Landsat 9	(2021–Present)	Sensor: OLI/TIRS (Op. Land Imager / Therm. Infrared Sensor)

GEE's public data archive includes >40 years of **satellite imagery** expanded daily:

2. **Sentinel** collections
   ⇒ ESA/Copernicus Program, since 2014
   ⇒ constellation of satellites consisting comprising various sensors:
   - **Sentinel-1**: radar imaging (C-band SAR)
   - **Sentinel-2**: optical & infrared imaging (VIS/SWIR)
   - **Sentinel-3**: optical & infrared imaging (VIS/SWIR/TIR)
   - **Sentinel-5P**: ultra-violet, optical, infrared imaging (UV/VIS/NIR/SWIR)

   ⇒ GEE archive includes:

   - Sentinel 1 (2014–Present) Sensor: SAR (C-band), GRD scenes (Ground Range Detected)
   - Sentinel 2 (2015–Present) Sensor: MSI (Multispectral Instrument)
   - Sentinel 3 (2016–Present) Sensor: OLCI (Ocean and Land Color Instrument)
   - Sentinel 5P (2018–Present) Sensor: TROPOMI (TROPOspheric Monitoring Instrument)

GEE's public data archive includes >40 years of **satellite imagery** expanded daily:

3. **Sentinel** collections
   ⇒ ESA/Copernicus Program, since 2014
   ⇒ constellation of satellites consisting comprising various sensors:
   - **Sentinel-1**: radar imaging (C-band SAR)
   - **Sentinel-2**: optical & infrared imaging (VIS/SWIR)
   - **Sentinel-3**: optical & infrared imaging (VIS/SWIR/TIR)
   - **Sentinel-5P**: ultra-violet, optical, infrared imaging (UV/VIS/NIR/SWIR)

   ⇒ GEE archive includes:

   - Sentinel 1 (2014–Present) Sensor: SAR (C-band), GRD scenes (Ground Range Detected)
   - Sentinel 2 (2015–Present) Sensor: MSI (Multispectral Instrument)
   - Sentinel 3 (2016–Present) Sensor: OLCI (Ocean and Land Color Instrument)
   - Sentinel 5P (2018–Present) Sensor: TROPOMI (TROPOspheric Monitoring Instrument)

GEE's public data archive includes >40 years of **satellite imagery** expanded daily:

4. **MODIS** collections
   ⇒ NASA's "Moderate Resolution Imaging Spectroradiometer"
   ⇒ sensor on board 2 satellites: Terra (since 1999) & Acqua (since 2002)
   ⇒ GEE archive includes: daily surface spectral reflectances from MODIS, as well as several derived products (e.g., vegetation indices, snow cover, etc)
5. High-Resolution Imagery
   ⇒ GEE archive currently includes:
   - Planet dataset: imagery acquired by the Planet Labs constellation of small satellites (Planetscope 4.77 m resolution; SkySat 0.8 m resolution)
   - NAIP (*National Agriculture Imagery Program*) dataset: aerial imagery acquired during the agricultural growing seasons in the continental U.S. (0.6 m resolution)

GEE's public data archive includes >40 years of **satellite imagery** expanded daily:

4. **MODIS** collections
   ⇒ NASA's "Moderate Resolution Imaging Spectroradiometer"
   ⇒ sensor on board 2 satellites: Terra (since 1999) & Acqua (since 2002)
   ⇒ GEE archive includes: daily surface spectral reflectances from MODIS, as well as several derived products (e.g., vegetation indices, snow cover, etc)

5. High-Resolution Imagery
   ⇒ GEE archive currently includes:
   - Planet dataset: imagery acquired by the Planet Labs constellation of small satellites (Planetscope 4.77 m resolution; SkySat 0.8 m resolution)
   - NAIP (*National Agriculture Imagery Program*) dataset: aerial imagery acquired during the agricultural growing seasons in the continental U.S. (0.6 m resolution)

In addition to satellite imagery, GEE also includes **other scientific datasets**:

1. Digital Elevation Models (DEMs) collections
   ⇒ DEMs describe Earth's topography
   ⇒ GEE archive includes:
   - global DEMs:
     - SRTM DEM (NASA's Shuttle Radar Topography Mission) data at 30-meter resolution
     - NASA DEM (NASA reprocessing of SRTM data with improved accuracy) 30-meter resolution
     - Copernicus DEM (ESA) data at 30-meter resolution
     - ALOS (JAXA) data at 30-meter resolution
   - regional DEMs at higher resolutions

In addition to satellite imagery, GEE also includes **other scientific datasets**:

1. Digital Elevation Models (DEMs) collections
   ⇒ DEMs describe Earth's topography
   ⇒ GEE archive includes:
   - global DEMs:
     - SRTM DEM (NASA's Shuttle Radar Topography Mission) data at 30-meter resolution
     - NASA DEM (NASA reprocessing of SRTM data with improved accuracy) 30-meter resolution
     - Copernicus DEM (ESA) data at 30-meter resolution
     - ALOS (JAXA) data at 30-meter resolution
   - regional DEMs at higher resolutions

In addition to satellite imagery, GEE also includes **other scientific datasets**:

2. Thematic datasets:
   - Surface Temperature: includes land and sea surface temperature products derived from several spacecraft sensors, including MODIS, ASTER, and AVHRR, in addition to raw Landsat thermal data
   - Climate: includes climate models generate both long-term climate predictions and historical interpolations of surface variables
   - Atmospheric: includes ozone data from NASA's TOMS and OMI instruments and the MODIS Monthly Gridded Atmospheric Product
   - Weather: includes forecasted and measured conditions over short periods of time, including precipitation, temperature, humidity, and wind, and other variables. Includes in particular NOAA's Global Forecast System (GFS) and the NCEP Climate Forecast System (CFSv2)
   - Land Cover: includes the physical landscape in terms of land cover classes such as forest, grassland, and water
   - Urban: includes a number of demographic data products and settlement characteristics
   - Other Geophysical Data: includes data from other satellite image sensors

The GEE data model revolves around the following components:

- **Image objects**
  - ⇒ `ee.Image`
  - ⇒ `Image` objects represent raster data (i.e., satellite imagery, climate data, or any gridded data)
  - ⇒ `Image` objects consist of one or more bands, where each band represents a different type of information (e.g., red, green, blue bands for RGB imagery)
- **Geometry objects**
  - ⇒ `ee.Geometry`
  - ⇒ `Geometry` objects represent vector data (i.e., points, lines, or polygons)
  - ⇒ `Geometry` objects support different geometries:
    - Point: point defined by a list of two coordinates [lon,lat] in the given projection
    - LineString: a list of at least 2 `Points`
    - LinearRing: a closed `LineString`
    - Polygon: a list of rings defining the boundaries of the polygon
    - as well as: `MultiPoint`, `MultiLineString`, `MultiPolygon`

- **Feature objects**
  - ⇒ `ee.Feature`
  - ⇒ `Feature` objects are `Geometry` objects with attributes
  - ⇒ `Feature` objects store a `Geometry` object (or null) and a *properties* property storing a dictionary of other properties

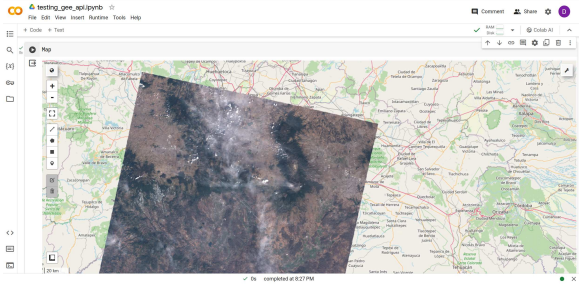
- **Collection objects**
  - ⇒ Collections are groups of `Image` or `Feature` objects
  - ⇒ `ee.ImageCollection` : group of `Image` objects, which can be organized and filtered based on various criteria such as date, metadata, or spatial location
  - ⇒ `ee.FeatureCollection` : group of `Feature` objects

3.3. Jumpstart into image visualization

```
# Initialize
import geemap
import ee
ee.Authenticate()
ee.Initialize(project='your-project-id')  # Initialize using project-id with enabled GEE API
Map = geemap.Map()  # Initialize map

# Select image and visualization parameters
image = ee.Image('LANDSAT/LC08/C02/T1_TOA/LC08_026047_20200116');  # Landsat 8 Top of Atmosphere (TOA) image over Popocatépetl
vis_param = {'bands': ['B4', 'B3', 'B2'], 'min': 0, 'max': 0.3, 'gamma': 1.3}  # Select bands for true color RGB

# Center map on image and display
Map.centerObject(image, 9)
Map.addLayer(image, vizParams, 'Landsat 8 true color')
Map
```



3.4. Image search & filtering

Image collections can be searched and filtered based on various criteria:

- date: filterDate
- location: filterBounds

```
# Create geometry Point object
lon, lat = -98.622, 19.023  # Popocatépetl
roi = ee.Geometry.Point(lon, lat)

# Filter collection by date & location
collection = (
    ee.ImageCollection('LANDSAT/LC08/C02/T1_TOA')
    .filterDate('2020-01-01', '2020-12-31')  # Filter by date
    .filterBounds(roi)  # Filter by location
    )
print(collection.size().getInfo())

# Get first image in collection
image = collection.first()
```