WGDisasters 24

CEOS Analytics Lab
(CAL)
(Modified and presented at WGISS-60)

Committee on Earth Observation Satellites

Jonathan Hodge
Adapted by Alex Leith
CEOS SEO

CAL Topics

- Overview & Capabilities
- CEOS SARCalNet Support
- Case Study: ALOS-2 Forested Wetlands Inundation Mapping
- Relevance to WGDisasters & ongoing opportunities

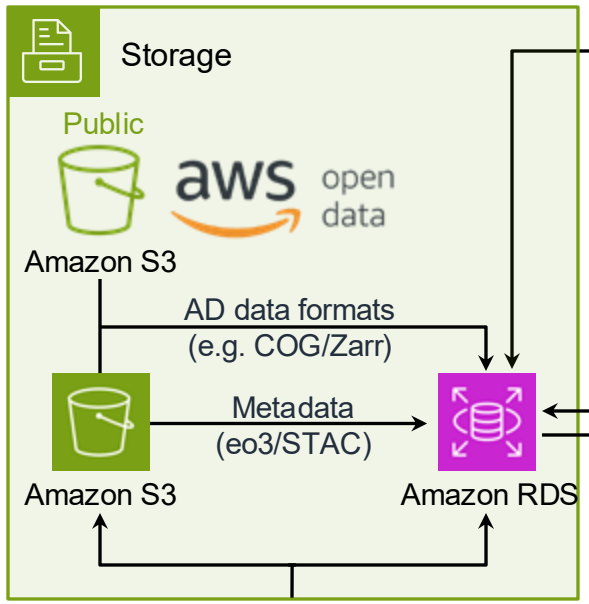
CEOS Analytics Laboratory (CAL)

(Formerly known as EAIL)

cal.ceos.org

CEOS Analytics Laboratory

https://cal.ceos.org



Data for visualization

Metadata queries

Metadata and data response

Data updates and new product generation
(ARCO data formats and eo3 metadata)

Elastic Kubernetes Service

OGC Web Services

OWS

hpa

- *WMS, WMTS and WCS*
- *Horizontal Pod Autoscaler (HPA) adjusts to user demand*
- *Customisable, on-the-fly functions and calculations*

Data visualisation

terria

- *Handles OGC, ESRI, other map services and geospatial files*
- *2D, 3D and 4D visualisations*
- *Above and below ground*
- *Charting and time-series*

Analytics environment

Amazon EC2
Auto scaling

jupyter

- *32GB, 8CPU default capacity*
- *>500 pre-installed python libraries*
- *Optional extra CPU/RAM*
- *Optional GPU*

Flexible, scalable workers

Amazon EC2
Auto scaling

dask

- *Powerful parallel processing*
- *Optional GPU, extra CPU or extra RAM (per worker)*

Argo Workflows

argo

- *Powerful workflow engine*
- *Hugely scalable*
- *Automated data and product updates*
- *Customised, on-demand data workflows*

Powered by:

OPEN DATA CUBE

CSIRO Earth Analytics Science and Innovation

Automation and orchestration with:

Terraform HELM flux

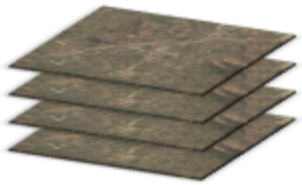
In addition to:

Amazon Cognito Amazon EFS Amazon DynamoDB Amazon CloudFront

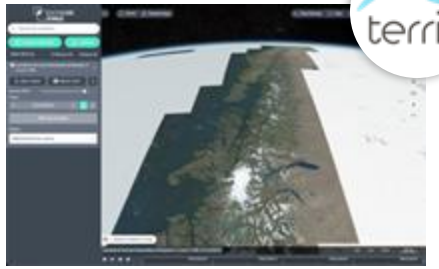
CEOS Analytics Laboratory

CEOS Analytics Laboratory

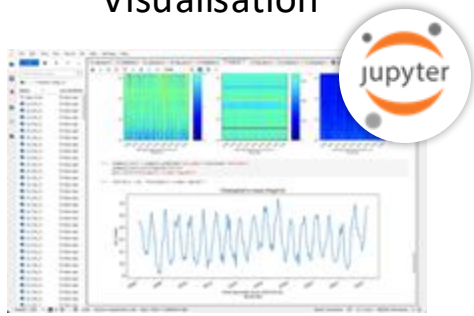
ARD & ARCO



Analysis Ready:
- Landsat 5/7/8/9
- Sentinel 1/2/3
- MODIS
- VIIRS
- ALOS
- Elevation
- Small sats
- Commercial
- …



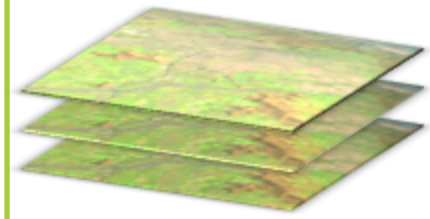
Visualisation



Exploratory analysis

Scalable workflows

from pilot or project → **to national or continental**



1 computer or 200
30GB RAM or 6TB
8 CPU or 1600

ARCO = Analysis Ready, Cloud Optimized

Data analytics - JupyterLab

jupyter AWRA Yearly Total E0 Averages Integrated Widgets Last Checkpoint: 07/25/2017 (autosaved)

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

Code

Cropping a NetCDF with GeoJSON and Knowledge Network

This notebook provides and example of interacting with the knowledge network to cookied cut /crop a netcdf file based on geojson retrieved interactively. The notebook provides code and widgets to explore and retrieve geojson interactively from a the knowledge network apis. Selecting a geojson region loads the geojson into a variable ready for cropping. A netcdf file is queried from thredds using the netcdf4 library currently the code is specific to a particularly netcdf file and variable but can readily be modified to a netcdf file of your choice. Finally the geojson object is used to crop a region of the netcdf file.

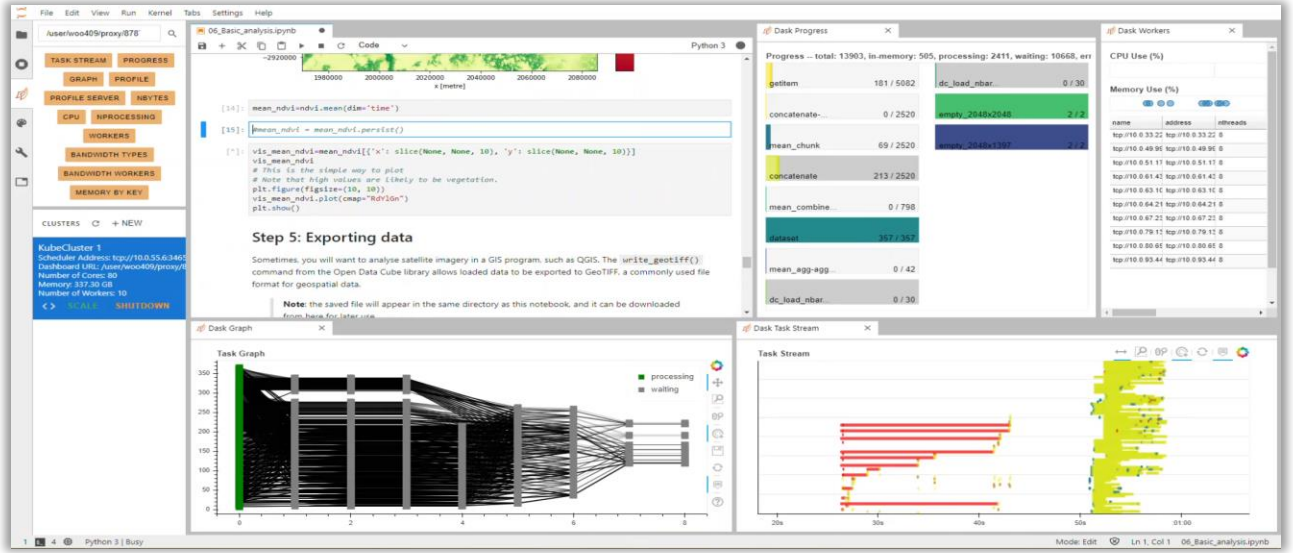
Feedback and improvements welcome. Please create a modified notebook using File > Make a Copy.

Thanks to Tony Zhao and David Robertson's team for sharing there code and pointing me in the right direction with cropping netcdf.

In [1]:
```
%matplotlib inline

import knownet_tools
# json
import json
# for geojsonpip
import geojson
# for geoprocessing
import owslib

# widgets
# mapping widget
import ipyleaflet as ll
# interactive widgets
from ipywidgets import interact, interactive, fixed, interact_manual
import ipywidgets as widgets
from ipywidgets import HBox

import sys, os, time, requests
from IPython.display import clear_output
import IPython.display as idisplay

from netCDF4 import Dataset
import netCDF4
import numpy as np
import pydap
import requests

from matplotlib import pyplot

import rasterio.features
import rasterio.mask as mask
import rasterio

from shapely.geometry import asShape, Polygon, mapping
import shapely

from rasterio.features import rasterize
from rasterio.transform import Affine

#graphing
```

HoloViews Datashader

plotly bokeh matplotlib

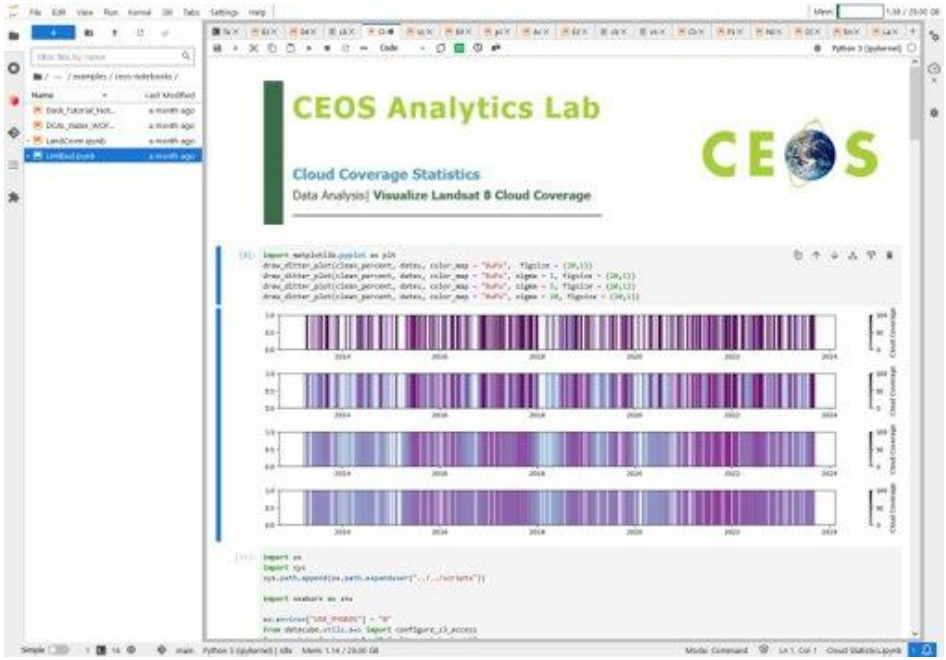


dask

"Embarrassingly parallel workloads"

Example Notebooks

- Cloud Statistics & filtering
- Land Change
- Water Threshold
- Vegetation Phenology & change
- Spectral Products
- SAR examples
- Water Observations from Space (WOFS)



Notebooks available at: https://github.com/AMA-Labs/cal-notebooks

Advanced analytics capabilities

- AWS Graphics Processing Unit (GPU) nodes
- Additional scientific programming options with R
- Numerous machine learning and code scaling options
- Jupyter images updated quarterly to ensure latest versions

Alex Leith @alexgleith · Jan 5

Promote

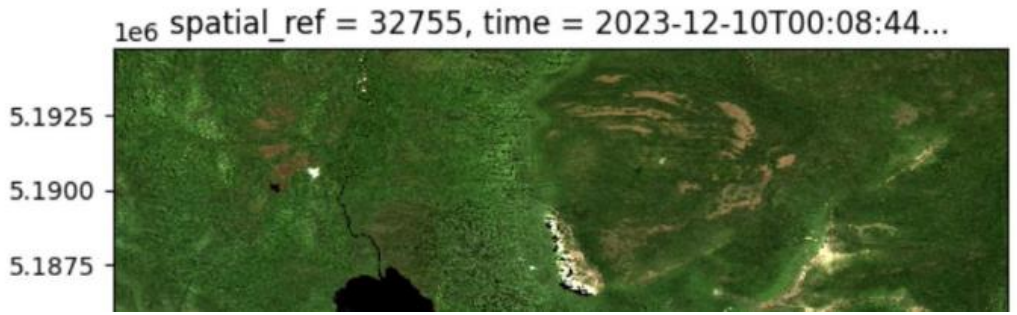
Did you know that with 2 Python libraries, 6 lines of code and around 15 seconds, you can load satellite data from anywhere in the world?

This is so much easier than it used to be!

```python
from pystac_client import Client
from odc.stac import load

client = Client.open("https://earth-search.aws.element84.com/v1")
collection = "sentinel-2-l2a"
tas_bbox = [146.5, -43.6, 146.7, -43.4]
search = client.search(collections=[collection], bbox=tas_bbox, datetime="2023-12")

data = load(search.items(), bbox=tas_bbox, groupby="solar_day", chunks={})
data[["red", "green", "blue"]].isel(time=2).to_array().plot.imshow(robust=True)
```

✓ 16.6s

```
<matplotlib.figure.AxesImage at 0x2bfd7dd50>
```



Combining data

```python
def mask_elevation(
    ds: Dataset,
    ds_to_mask: Dataset | None = None,
    threshold: float = 10,
    return_mask: bool = False,
) -> Dataset:
    """
    Mask elevation. Returns 1 for high areas, 0 for low
    """
    e84_catalog = "https://earth-search.aws.element84.com/v1/"
    e84_client = Client.open(e84_catalog)
    collection = "cop-dem-glo-30"

    items = search_across_180(
        region=ds.odc.geobox, client=e84_client, collections=[collection]
    )

    # Using geobox means it will load the elevation data the same shape as the other data
    elevation = load(items, measurements=["data"], like=ds.odc.geobox).squeeze()

    # True where data is above elevation
    mask = elevation.data < (threshold * 1.0)

    return apply_mask(ds, mask, ds_to_mask, return_mask)
```

Links to resources

- Alex's simple S-2 access example: https://gist.github.com/alexgleith/dc49156aab4b9270b0a0f145bd7fa0ce
- Cloud Native Geo for EO Workshop Notebooks: https://github.com/auspatious/cloud-native-geospatial-eo-workshop
- Example of loading elevation and combining with other data: https://github.com/digitalearthpacific/dep-mangroves-ammi/blob/main/src/util.py#L102
- ODC STAC Library: https://odc-stac.readthedocs.io/en/latest/