SATPY

an open-source Python library for Earth Observation satellite data

Pierre de Buyl, PyTroll developers

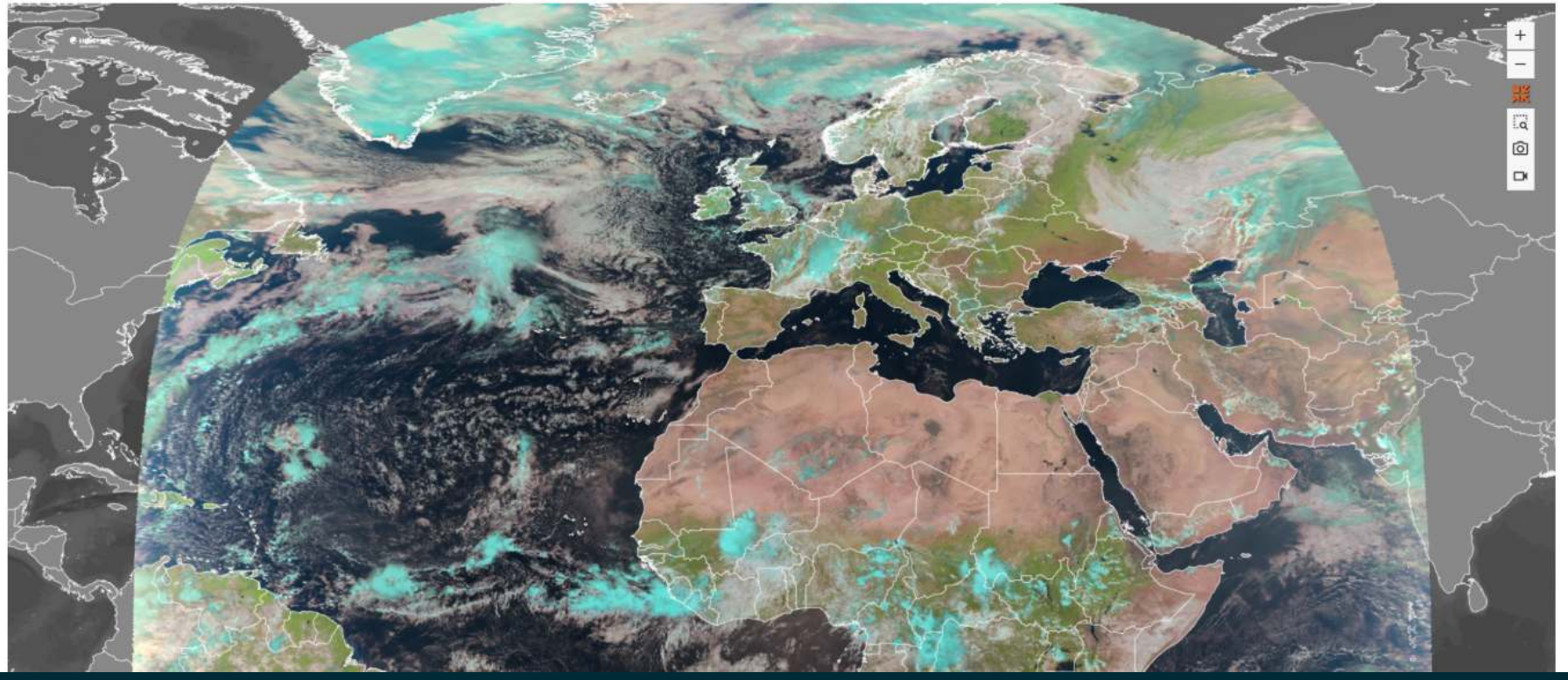
EUMETSAT 2025 - Lyon

OVERVIEW

SATPY @ VIEW.EUMETSAT.INT

PyTroll/satpy-based image generation for MSG 0 degree service

As **announced in June**, the gradual transition of the EUMETView image generation of MSG layers to a new, PyTroll/satpy-based system is ongoing. After the transition of MSG IODC layers, MSG 0 degree layers are also migrated to the new system on 26 August, with the MSG RSS Service migration to follow soon, which will complete the transition.



WHAT IS SATPY?

- Started in 2015.
- Satpy is an open-source software library for reading and processing data from Earth observing satellite instruments.
- Satpy is part of the PyTroll project, a collection of open-source software.
- Satpy can probably read "your" satellite (or you can add the reader)

WHO IS SATPY

- PyTroll is a community of developers started in 2009.
- PyTroll is supported by a Memorandum of Understanding, a formal collaboration agreement between SMHI, FMI, KNMI, DWD, MeteoSwiss, and MET Norway.
- More than 100 persons have contributed to the satpy repository.

Pytroll

Code of Conduct Community Gallery Coding Guidelines History of Pytroll Pytroll Project Logos

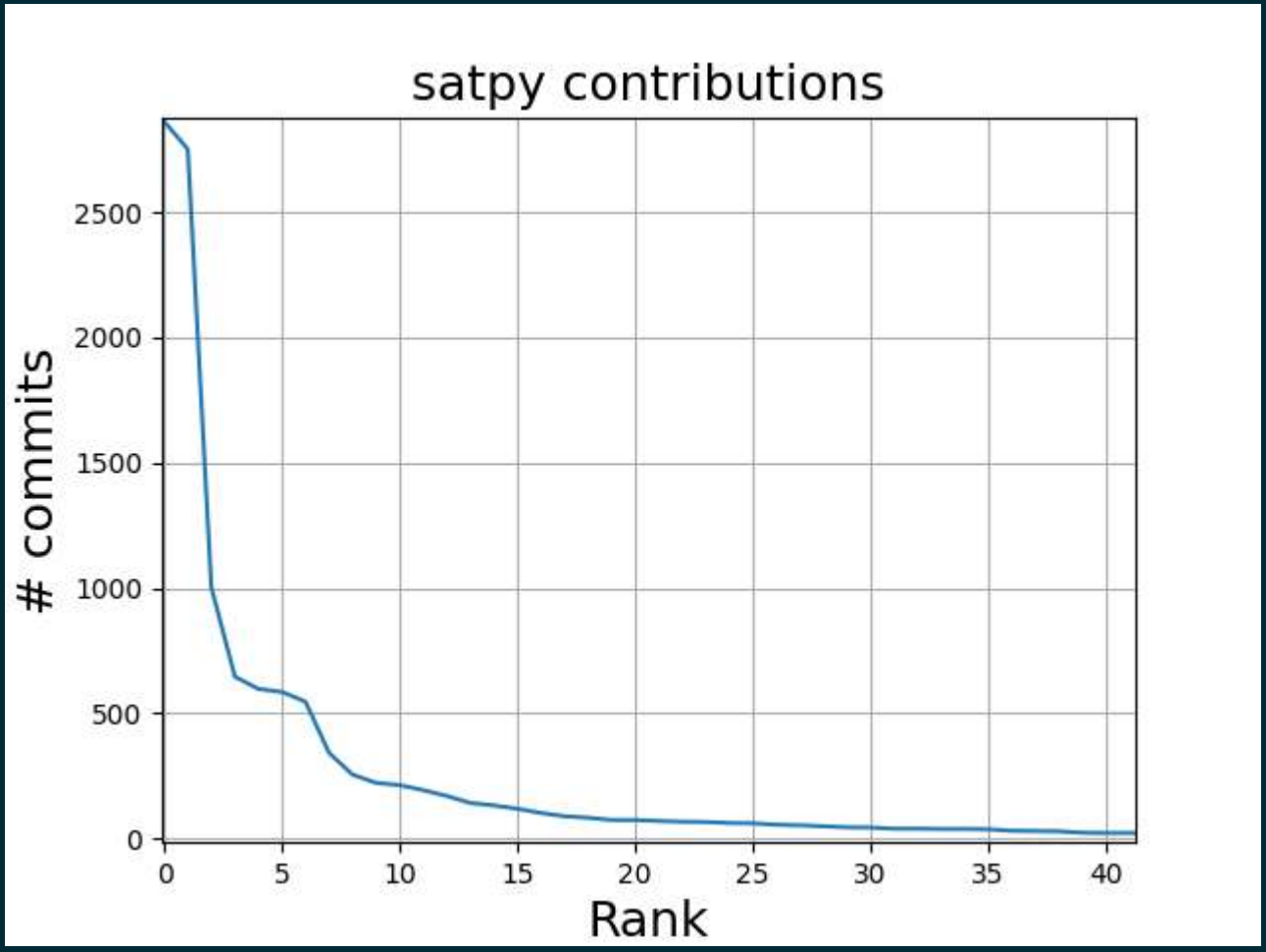
Contributor Weeks Pytroll Manifesto Packages Summary Software Pytroll User Days

Pytroll is an easy to use, modular, free and open source python framework for the processing of earth observation satellite data. The provided python packages are designed to be used both in R&D environments and in 24/7 operational production.

The focus is on atmospheric applications and imaging sensors, but as seen from the list of supported satellite sensors below the data that can be handled by Pytroll allows the usage in a wide range of earth sciences.

PYTROLL WORKSHOPS

SATPY CONTRIBUTOR GRAPH



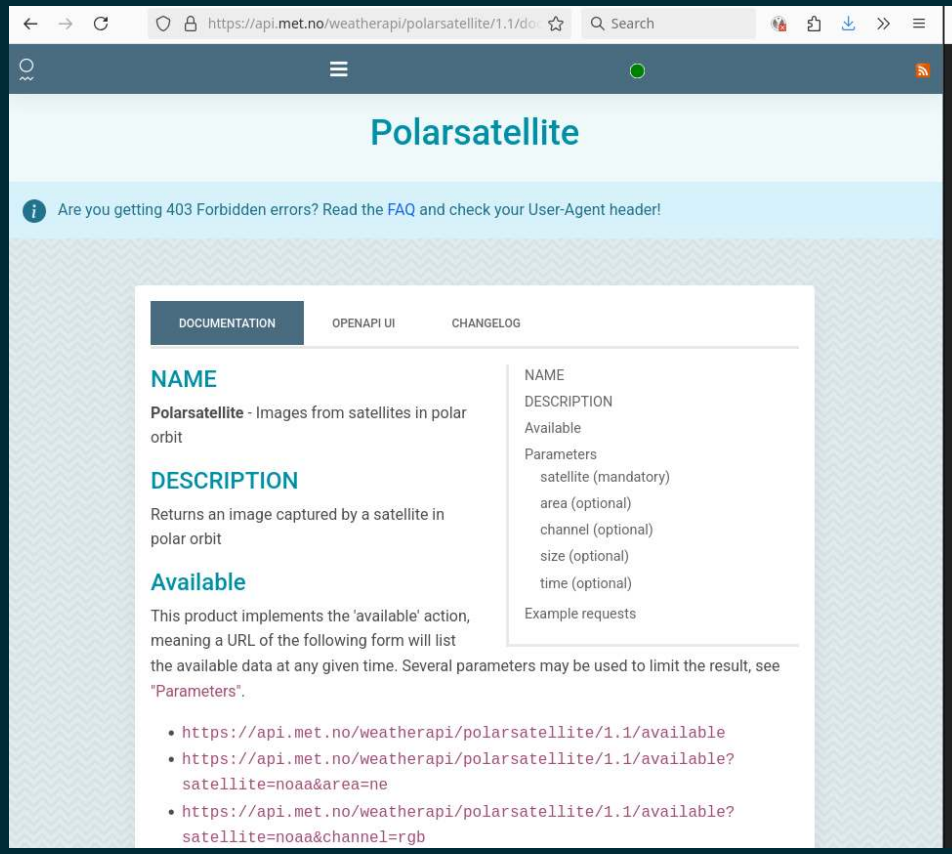
SATPY DISCUSSIONS

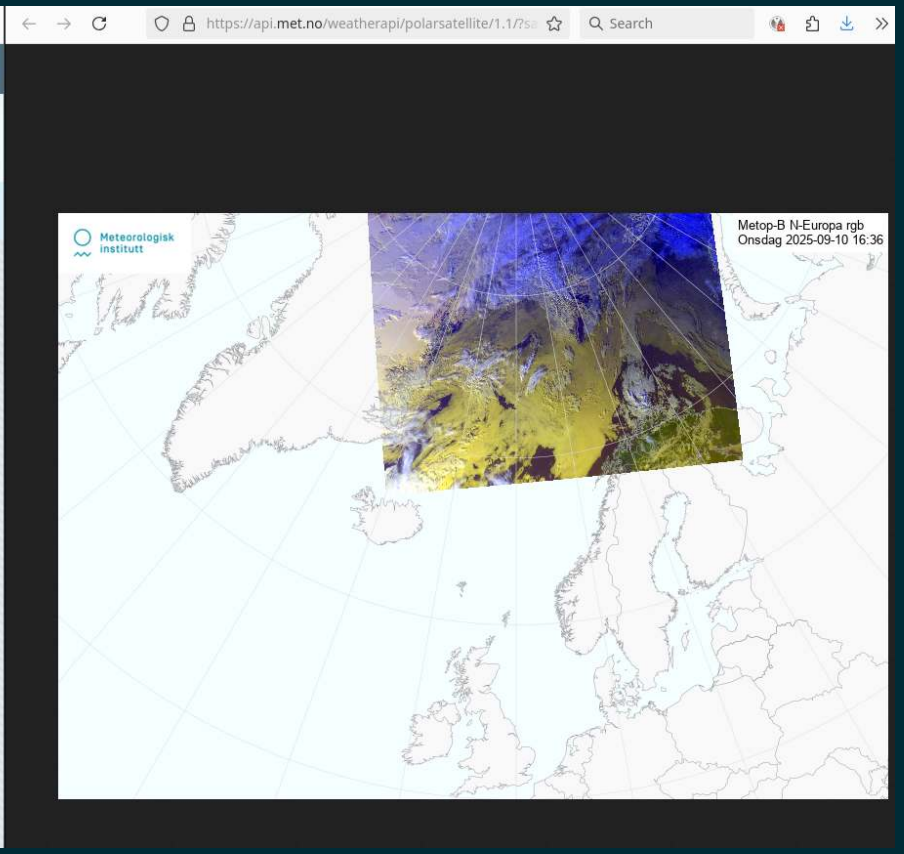
- Via GitHub issues or slack (`#satpy`, `#pyresample`, `#composites`, etc.)
- Tips for reading and displaying data
- Bug fixing
- New instruments

WHO USES SATPY?

CSPP GEOSPHERE (SSEC - UNIVERSITY OF WISCONSIN – MADISON)

MET.NO API SERVER






EUROPEAN MET OFFICES

EUMETSAT - FCI PRODUCT GUIDE

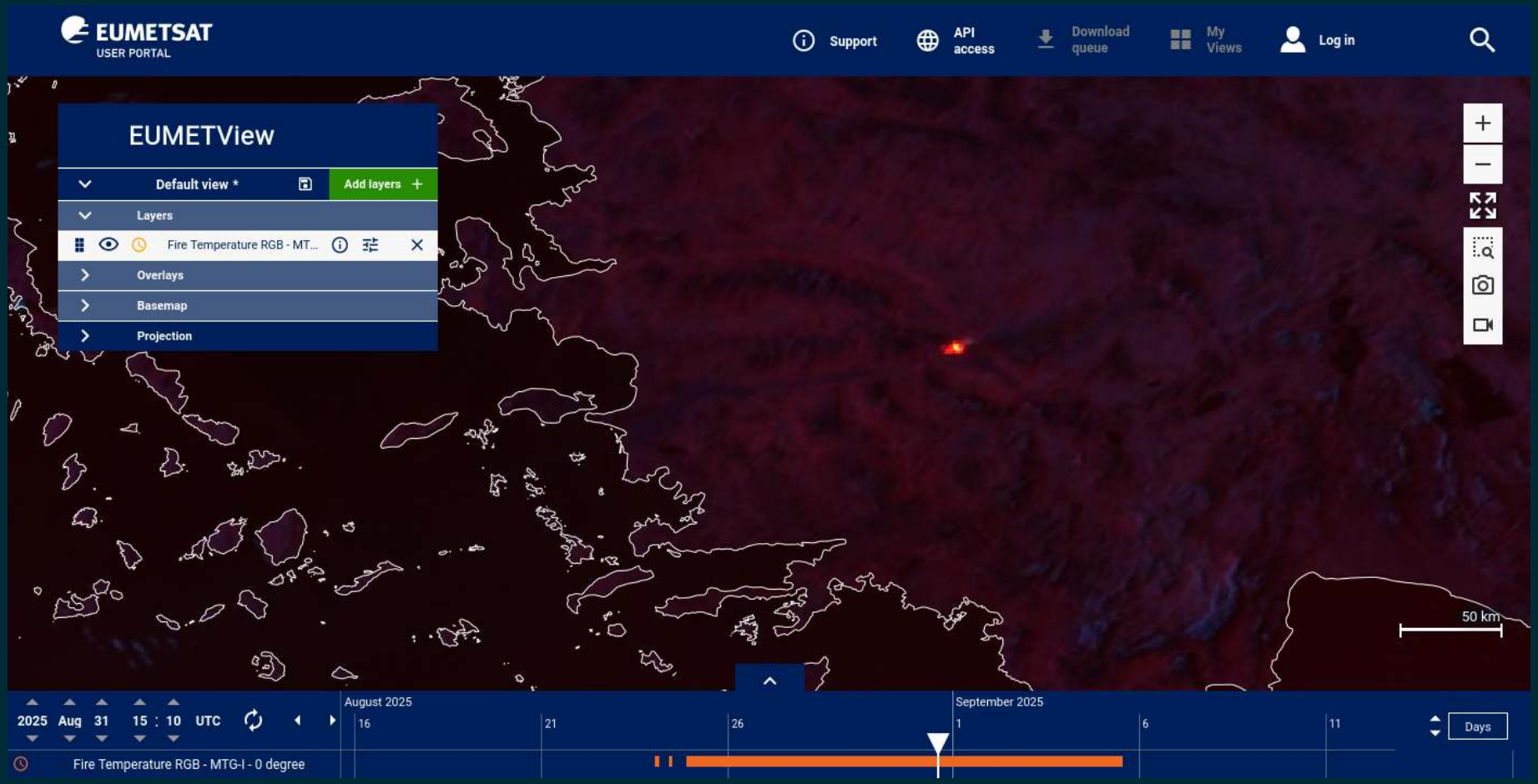
geophysical parameters from various file formats to the common *Xarray* DataArray and Dataset classes for easier interoperability with other scientific python libraries. *Satpy* also provides interfaces for creating RGB (Red/Green/Blue) images and other composite types by combining data from multiple instrument bands or products. Various atmospheric corrections and visual enhancements are provided for improving the usefulness and quality of output images. The *Pyresample* package is used to resample data to different uniform areas or grids. The documentation is available at http://satpy.readthedocs.org/.

Satpy also includes a reader of the FCI Level 1c data. The following Python code snippet shows an example on how to use *Satpy* to generate a `natural_color` RGB composite over the European area. A more detailed tutorial is available as part of the Satpy documentation at https://satpy.readthedocs.io/en/latest/examples/fci_l1c_natural_color.html .

```
from satpy.scene import Scene
from satpy import find_files_and_readers
```

EUMETSAT - FCI TRUE COLOR

EUMETSAT - FCI FIRE PRODUCT



TECHNICAL POINT OF VIEW

WHAT'S IN THE BOX (EXCERPT)

- Base dependencies: dask, **NumPy**, pillow, pykdtree, **pyproj**, **pyresample**, trollimage, trollsift, **xarray**, zarr
- Readers dependencies: pyhdf, netCDF4, rasterio, rioxarray, pyorbital, h5netcdf, pyPublicDecompWT, pygrib, fsspec, numba

CAPABILITIES

```
>>> print(satpy.available_readers())
['abi_l1b', 'abi_l1b_scmi', 'abi_l2_nc', 'acspo',
'agri_fy4a_l1', 'agri_fy4b_l1', 'ahi_hrit', 'ahi_hsd',
'ahi_l1b_gridded_bin', 'ahi_l2_nc', 'ami_l1b', 'amsr2_l1b',
'amsr2_l2', 'amsr2_l2_gaasp', 'amsub_l1c_aapp',
'ascat_l2_soilmoisture_bufr', 'atms_l1b_nc', 'atms_sdr_hdf5',
'avhrr_l1b_aapp', 'avhrr_l1b_eps', 'avhrr_l1b_hrpt',
'avhrr_l1c_eum_gac_fdr_nc', 'aws1_mwr_l1b_nc',
'aws1_mwr_l1c_nc', 'camel_l3_nc', 'clavrx', 'cmsaf-
claas2_l2_nc', 'electrol_hrit', 'epic_l1b_h5',
'eps_sterna_mwr_l1b_nc', 'fci_l1c_nc', 'fci_l2_bufr',
'fci_l2_grib', 'fci_l2_nc', 'fy3a_mersi1_l1b',
'fy3b_mersi1_l1b', 'fy3c_mersi1_l1b', 'generic_image', 'geocat',
'gerb_l2_hr_h5', 'ghi_l1', 'ghrsst_l2', 'gld360_ualf2',
'glm_l2', 'gms5-vissr_l1b', 'goci2_l2_nc', 'goes-imager_hrit',
```

- Calibration, Resampling, Composite generation

SATPY SCENE OBJECT

```
fn = glob.glob('/mnt/data/MTI1_202503101400/
        W*RAD*FDHSI*BODY*.nc')
scn = satpy.Scene(filenames=fn, reader='fci_l1c_nc')
```

SATPY SCENE - DATASET

```
scn.load(['ir_133'])
scn['ir_133']
```

```
<xarray.DataArray (y: 5568, x: 5568)> Size: 124MB
dask.array<concatenate, shape=(5568, 5568), dtype=float32,
chunksize=(150, 5568), chunktype=numpy.ndarray>
Coordinates:
    crs      object 8B
PROJCRS["unknown",BASEGEOGCRS["unknown",DATUM["unknown...
  * y        (y) float64 45kB -5.567e+06 -5.565e+06 ...
5.565e+06 5.567e+06
  * x        (x) float64 45kB -5.567e+06 -5.565e+06 ...
5.565e+06 5.567e+06
Attributes: (12/21)
    orbital_parameters:   {'satellite_actual_longitude':
-0.2617217555642128,...
    sensor:               fci
    standard_name:        toa_brightness_temperature
```

SATPY SCENE - COMPOSITES

```
scn.available_composite_names()
```

```
['24h_microphysics', 'airmass', 'ash', 'cimss_cloud_type',
'cimss_cloud_type_raw', 'cloud_phase',
'cloud_phase_distinction', 'cloud_phase_distinction_raw',
'cloud_phase_raw', 'cloud_phase_with_night_ir105', 'cloud_type',
'cloud_type_with_night_ir105', 'cloudtop',
'colorized_ir_clouds', 'convection',
'day_essl_colorized_low_level_moisture',
'day_essl_low_level_moisture', 'day_microphysics',
'day_severe_storms', 'day_severe_storms_tropical', 'dust',
'essl_colorized_low_level_moisture', 'essl_low_level_moisture',
'fci_fire_channels_sum', 'fire_temperature',
'fire_temperature_38refl', 'fire_temperature_rad',
'flames_masked', 'fog', 'geo_color',
'geo_color_background_with_low_clouds', 'geo_color_high_clouds',
'geo_color_low_clouds', 'geo_color_night', 'green_snow',
```

INSTALLATION USING MINIFORGE

```
# Step 1: Install and activate miniforge https://github.com/
        conda-forge/miniforge
# Step 2: Install satpy
mamba install satpy
# Step 3: Install reader dependencies
mamba install matplotlib ipympl rioxarray bottleneck cartopy
        libgdal-jp2openjpeg s3fs
```

INSTALLATION ON THE EWC

- Use miniforge or pip
- All examples in this presentation generated using jupyterhub on the EWC
- Access to data using eumdac or EUMETCast Terrestrial
- Used by Pytroll members with SEVIRI and FCI workflows

EXAMPLES

SENTINEL 2 (I)

Lyon, 2025-08-17

```
import satpy
filenames = satpy.find_files_and_readers(base_dir='/mnt/data/
        S2', reader='msi_safe')
scn = satpy.Scene(filenames=filenames)
scn.load(['false_color', 'true_color'])
scn.save_datasets()
```

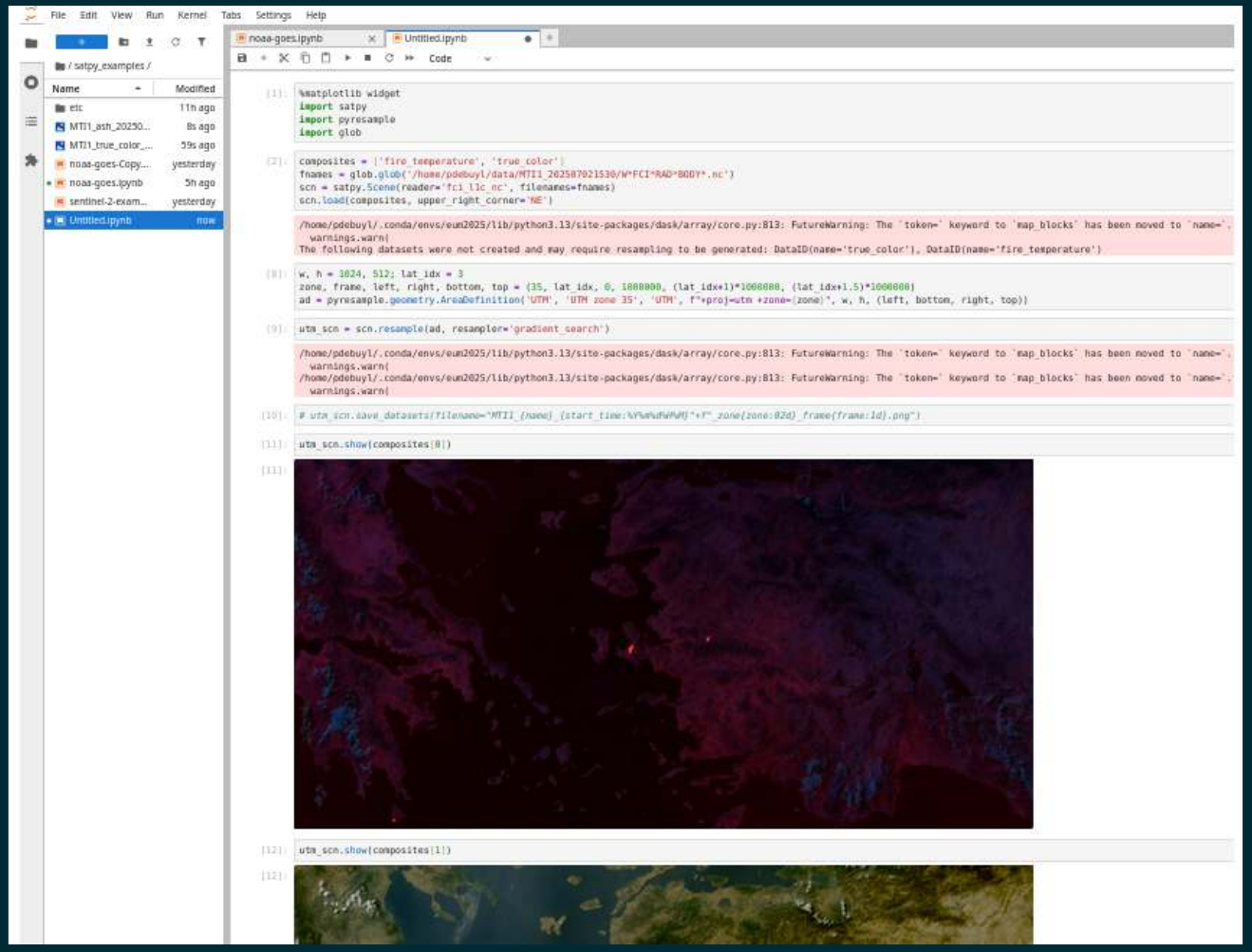

SENTINEL 2 (II)


SENTINEL 2 (III)

MTG FCI

- FCI reader in satpy used for EUMETSAT public imagery
- Access to the FCI decompressor via `hdf5plugin`
- Local area resampling
- Color composition
- Merging with Lightning Imager L2 data

MTG FCI NOTEBOOK



2025/07/02 IMAGE OF THE WEEK

[1]:
```
%matplotlib widget
import satpy
import pyresample
import glob
```

[ ]:
```
composites = ['fire_temperature', 'true_color']
fnames = glob.glob('/home/pdebuyl/data/MTI1_202507021530/W*FCI*RAD*BODY*.nc')
scn = satpy.Scene(reader='fci_l1c_nc', filenames=fnames)
scn.load(composites, upper_right_corner='NE')
```

[8]:
```
w, h = 1024, 512; lat_idx = 3
zone, frame, left, right, bottom, top = (35, lat_idx, 0, 1000000, (lat_idx+1)*1000000, (lat_idx+1.5)*1000000)
ad = pyresample.geometry.AreaDefinition('UTM', 'UTM zone 35', 'UTM', f"+proj=utm +zone={zone}", w, h, (left, bottom, right, top))
```

[ ]:
```
utm_scn = scn.resample(ad, resampler='gradient_search')
```

13]:
```
utm_scn.save_datasets(filename="MTI1_{name}_{start_time:%Y%m%d%H%M}"+f"_zone{zone:02d}_frame{frame:1d}.png")
```

11]:
```
utm_scn.show(composites[0])
```

11]:

LIGHTNING IMAGER

- Complex case: merging instruments and type of data
- The "LI Accumulated Flash Area" product stores the da
  per 30s chunks
- Reader `li_l2_nc` in satpy
- "MultiScene" with `true_color_with_night_ir10`
  `acc_flash_area`, and
  `true_color_with_night_ir105_acc_flash_a`


LIGHTNING IMAGER

GOES 19: READING REMOTE DATA USING FSSPEC

```
import os ; os.environ['SATPY_CONFIG_PATH'] = 'etc'
import satpy
reader_kwargs = {'storage_options': {'s3': {'anon': True},
        'simplecache': {'cache_storage': '/tmp/s3_cache'} } }
yyyy = '2025' ; jday = 228 ; hour = '13' # 16 august 2025 at
        13:00
filenames =[f'simplecache::s3://noaa-goes19/ABI-L1b-RadF/{yyyy}/
        {jday:03d}/{hour}/OR_ABI-L1b-RadF-M6C*_s{yyyy}{jday:03d}
        {hour}0*.nc']
composite = 'night_ir_with_color'
scn = satpy.Scene(reader='abi_l1b', filenames=filenames,
        reader_kwargs=reader_kwargs)
scn.load([composite])
```

File Edit View Run Kernel Tabs Settings Help

Launcher | noaa-goes.ipynb

Code

Notebook

[1]:
```
import os, itertools
os.environ['SATPY_CONFIG_PATH'] = 'etc'
import satpy
import s3fs
import pyresample
reader_kwargs = {'storage_options': {'s3': {'anon': True}, 'simplecache': {'cache_storage': '/tmp/s3_cache'} } }
```

[2]:
```
yyyy = '2025' ; jday = 228 # 16 august
hour = '13'
filenames =[f'simplecache::s3://noaa-goes19/ABI-L1b-RadF/{yyyy}/{jday:03d}/{hour}/OR_ABI-L1b-RadF-M6C*_s{yyyy}{jday:03d}{hour}0*.nc']
noaa_bucket = s3fs.S3FileSystem(anon=True)
composite = 'night_ir_with_color'
if len(noaa_bucket.glob(filenames[0].split('::')[1])) == 0:
    print("No file found")
scn = satpy.Scene(reader='abi_l1b', filenames=filenames, reader_kwargs=reader_kwargs)
scn.load([composite])
```

```
The following datasets were not created and may require resampling to be generated: DataID(name='night_ir_with_color')
```

[4]:
```
# Caribbean area coordinates N 31 W -84 S 11 E -54
projection = {'proj': 'latlong', 'datum': 'WGS84'}
ad = pyresample.AreaDefinition("eqc", "lat/lon", "eqc", width=900, height=600,
                               projection=projection, area_extent=(-84, 11, -54, 31))

sub_scn = scn.resample(ad, resampler='gradient_search')
sub_scn.show(composite)
```

```
/home/pdebuyl/.conda/envs/eum2025/lib/python3.13/site-packages/dask/_task_spec.py:759: RuntimeWarning: invalid value encountered in log
  return self.func(*new_argspec)
```

[4]:

GOES 19

- Files are saved to disk and animated with ffmpeg

```
ffmpeg -f image2 -pattern_type glob -framerate 6\
   -i 'night_ir_with_color_202508*.png' -pix_fmt yuv420p\
   -strict -2 G19_CARIBBEAN_20250816.mp4
```

GOES 19

REFERENCES & ACKNOWLEDGMENTS

- Many thanks to the PyTroll community for their help

- Satpy https://satpy.readthedocs.io/
- Satpy https://github.com/pytroll/satpy
- PyTroll https://pytroll.github.io/
- https://www.eumetsat.int/image-week-wildfires-eastern-mediterranean
- https://en.wikipedia.org/wiki/Hurricane_Erin_(2025)