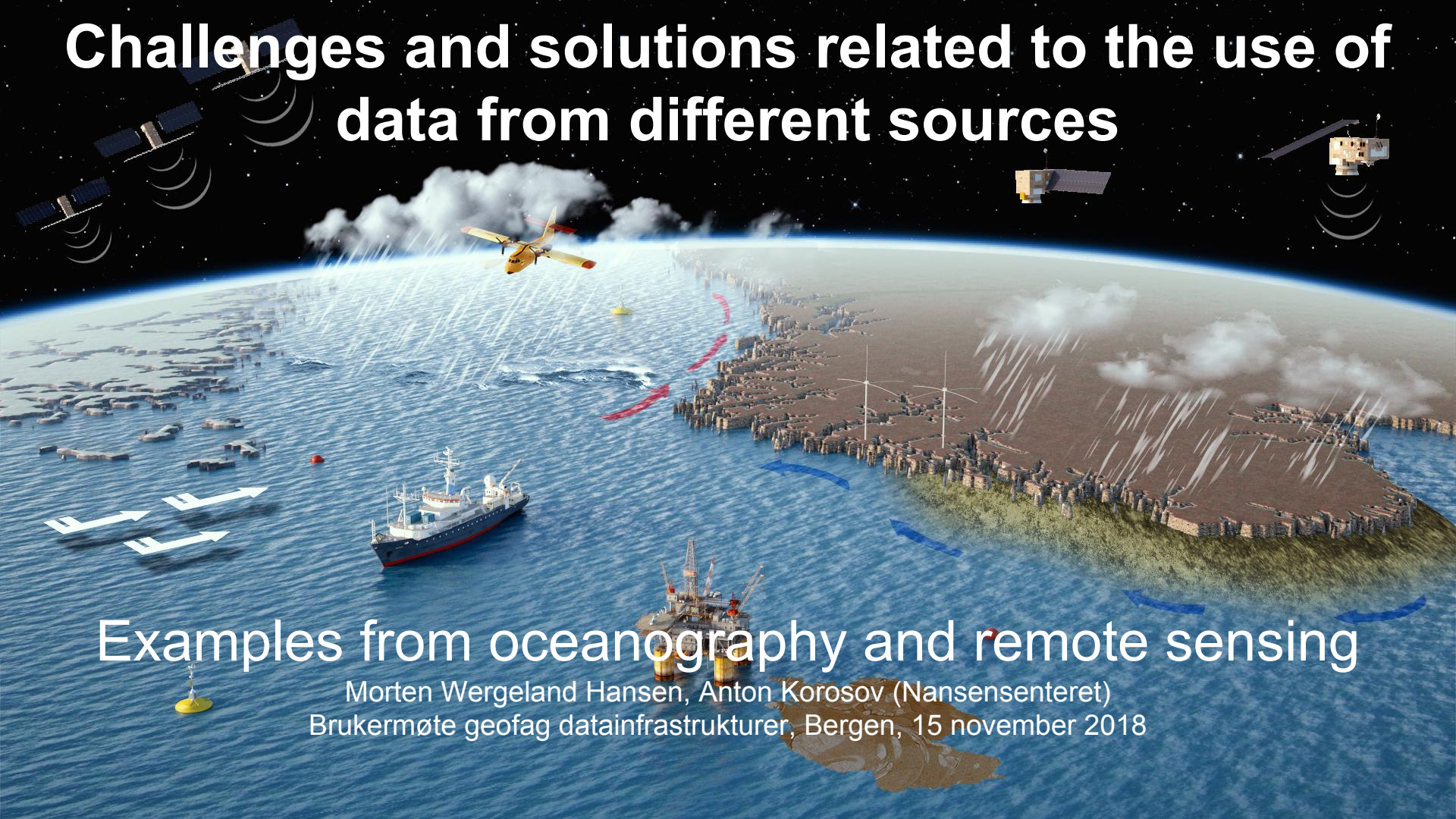
# Challenges and solutions related to the use of data from different sources



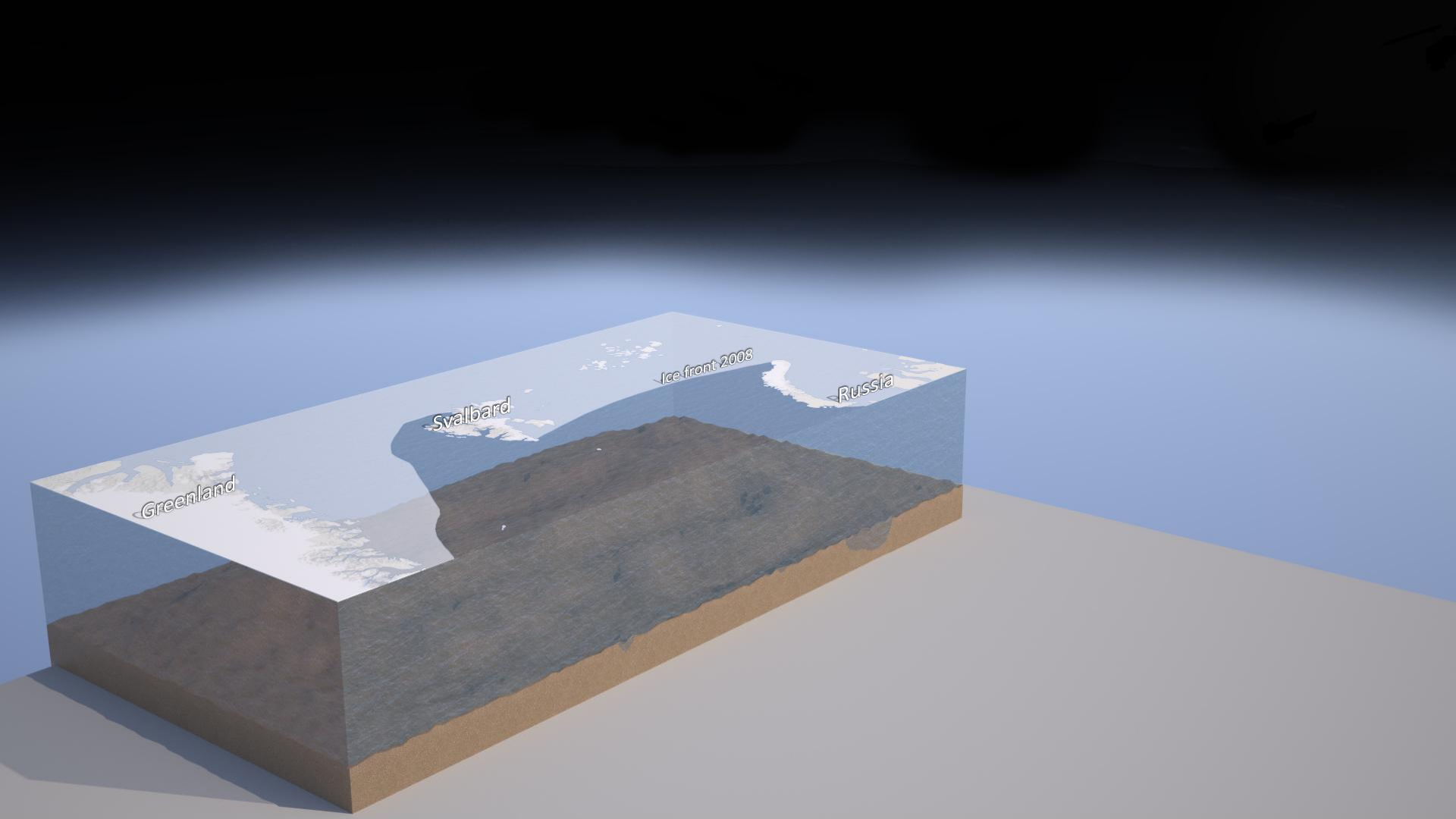
## Examples from oceanography and remote sensing

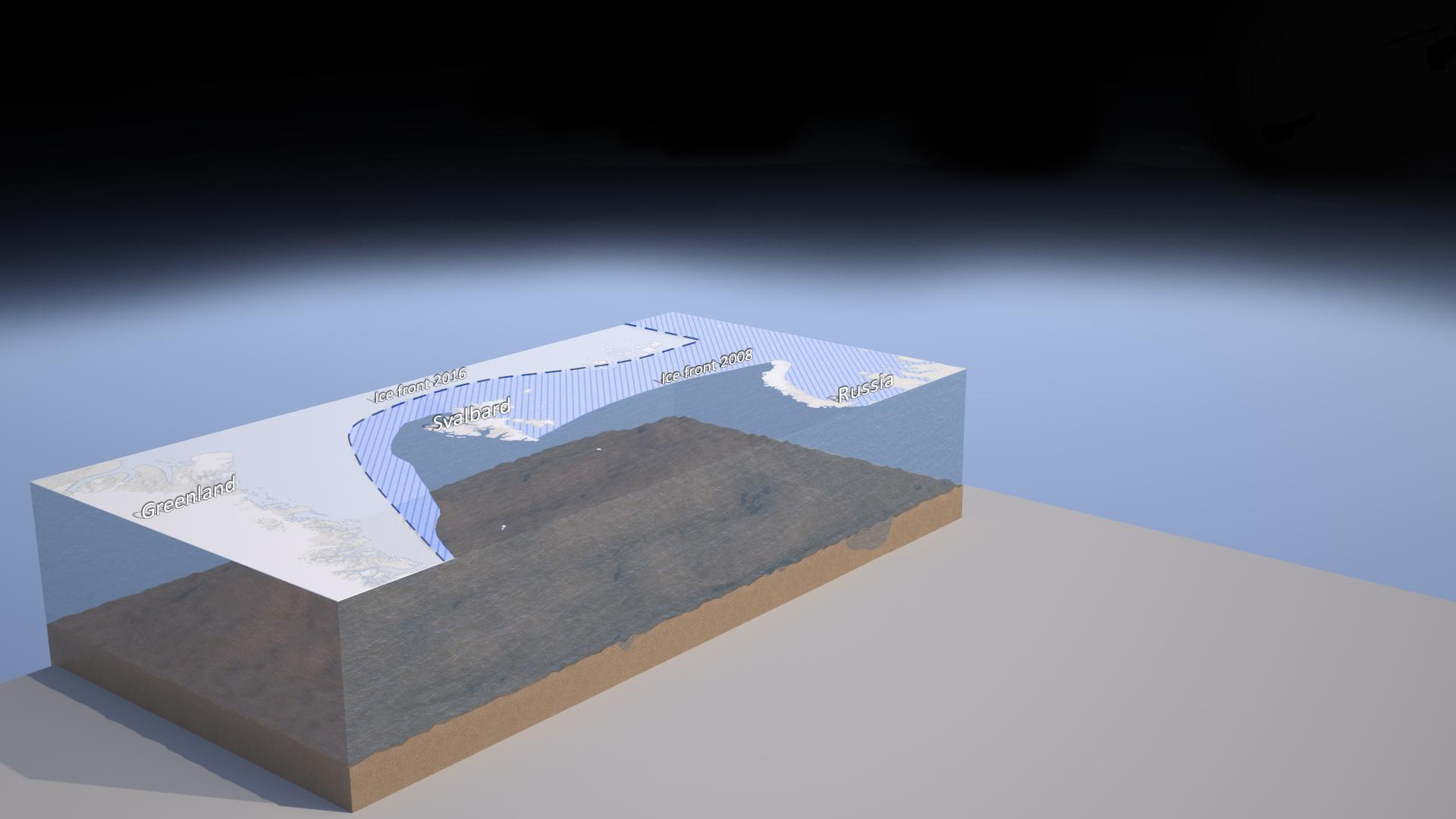
Morten Wergeland Hansen, Anton Korosov (Nansensenteret)

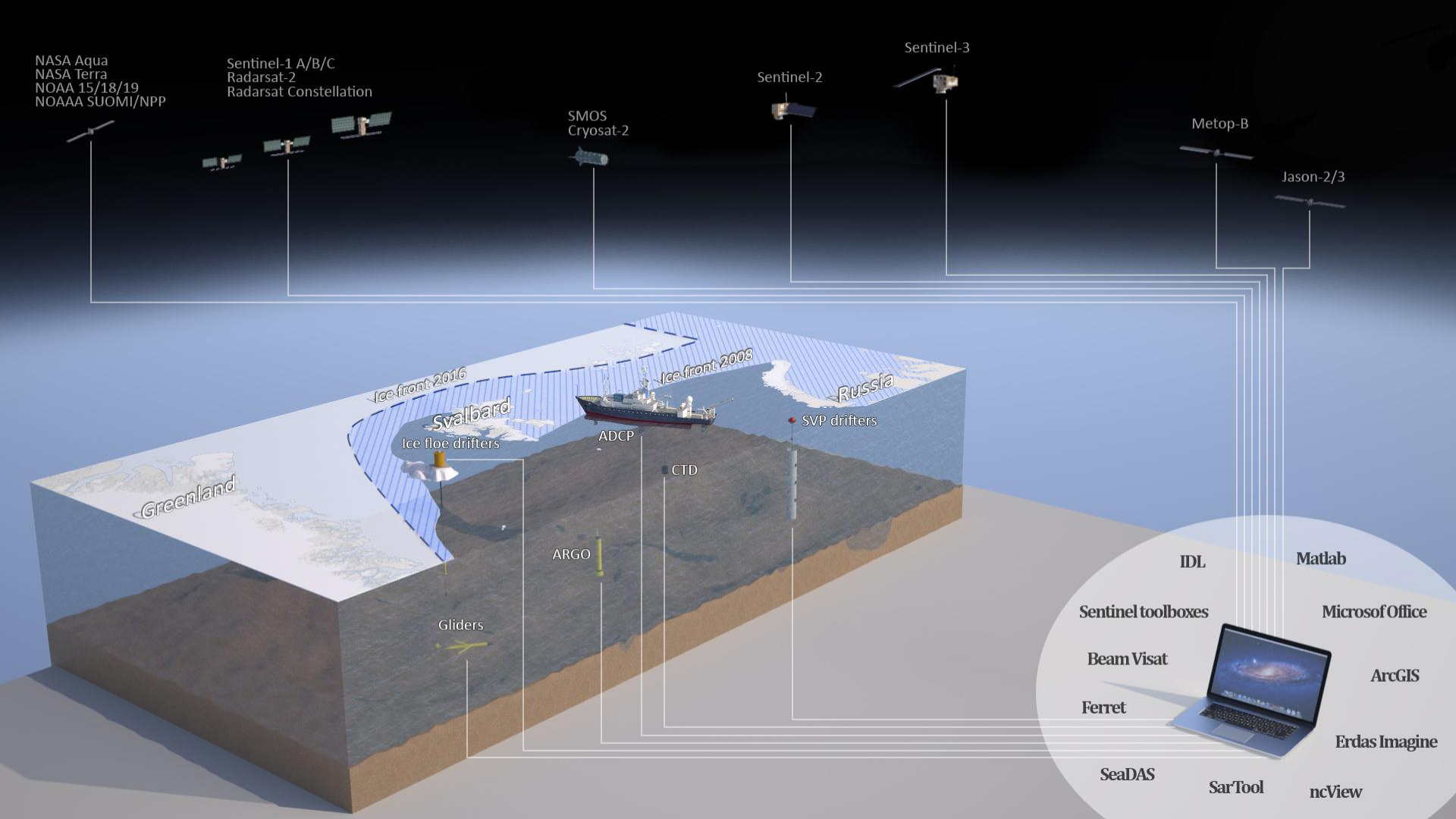
Brukermøte geofag datainfrastrukturer, Bergen, 15 november 2018

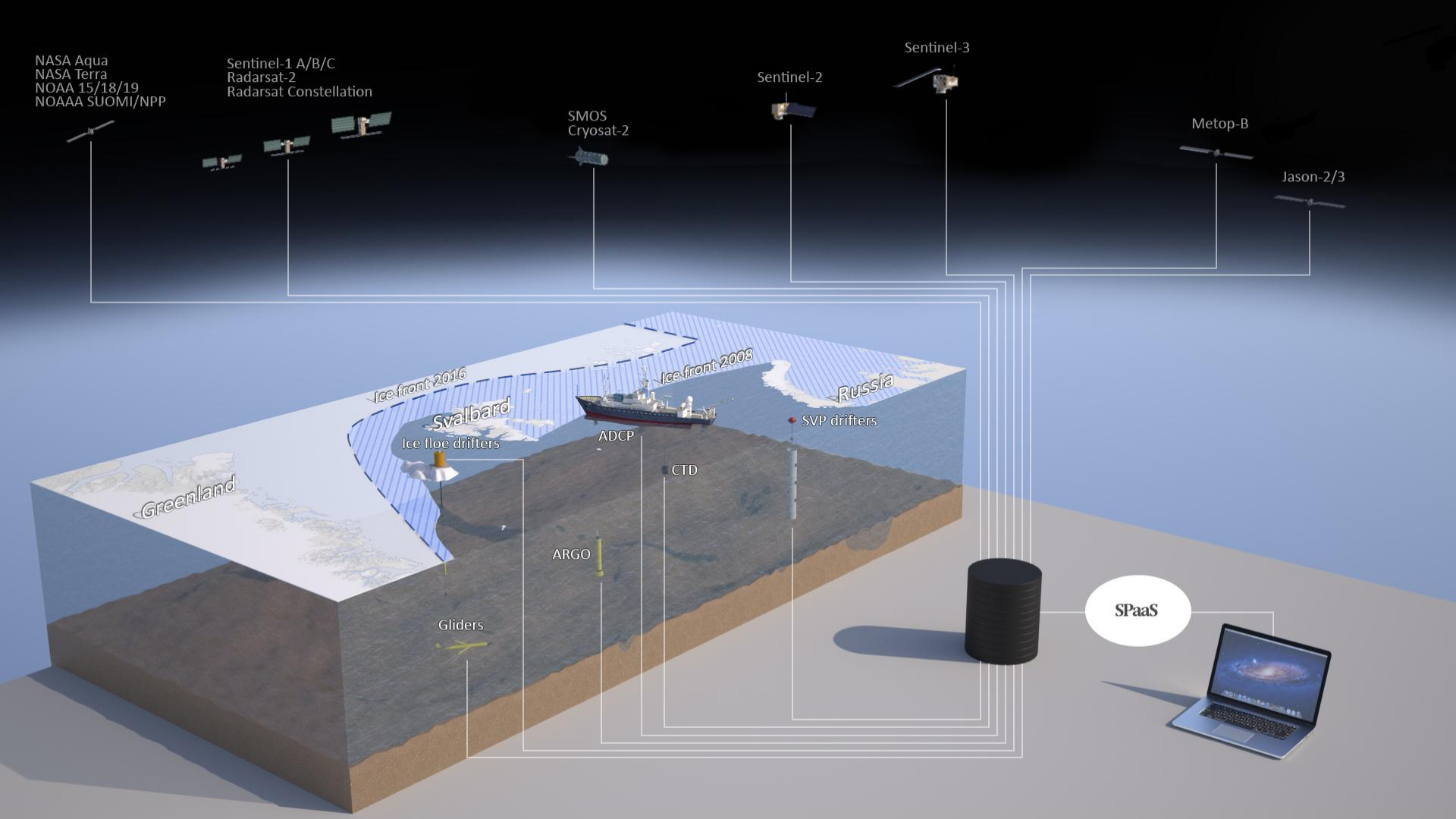
# Content

- Background
- New opportunities with open solutions
- Nansat
- Geo-SPaaS
- Use cases
  - Sea surface current from the ESA GlobCurrent project
  - Sea ice research at the Nansen Center


Ice front 2008
Russia
Svalbard
Greenland


Ice front 2016
Ice front 2008
Svalbard
Russia
Greenland


NASA Aqua
NASA Terra
NOAA 15/18/19
NOAAA SUOMI/NPP
Sentinel-1 A/B/C
Radarsat-2
Radarsat Constellation
SMOS
Cryosat-2
Sentinel-2
Sentinel-3
Metop-B
Jason-2/3
Ice front 2016
Ice front 2008
Svalbard
Russia
Greenland
Ice floe drifters
ADCP
SVP drifters
CTD
ARGO
Gliders
IDL
Matlab
Sentinel toolboxes
Microsof Office
Beam Visat
ArcGIS
Ferret
Erdas Imagine
SeaDAS
SarTool
ncView


NASA Aqua
NASA Terra
NOAA 15/18/19
NOAAA SUOMI/NPP
Sentinel-1 A/B/C
Radarsat-2
Radarsat Constellation
SMOS
Cryosat-2
Sentinel-2
Sentinel-3
Metop-B
Jason-2/3
Ice front 2016
Ice front 2008
Svalbard
Russia
Greenland
Ice floe drifters
ADCP
SVP drifters
CTD
ARGO
Gliders
SPaaS

# New opportunities with open access

- Political changes
  - Publicly funded data should be open and accessible
- New standards
  - NetCDF-CF
  - OpenDAP - users can define and select slice of data set before download
  - OGC, INSPIRE, NASA DIF, NASA GCMD keywords, etc.

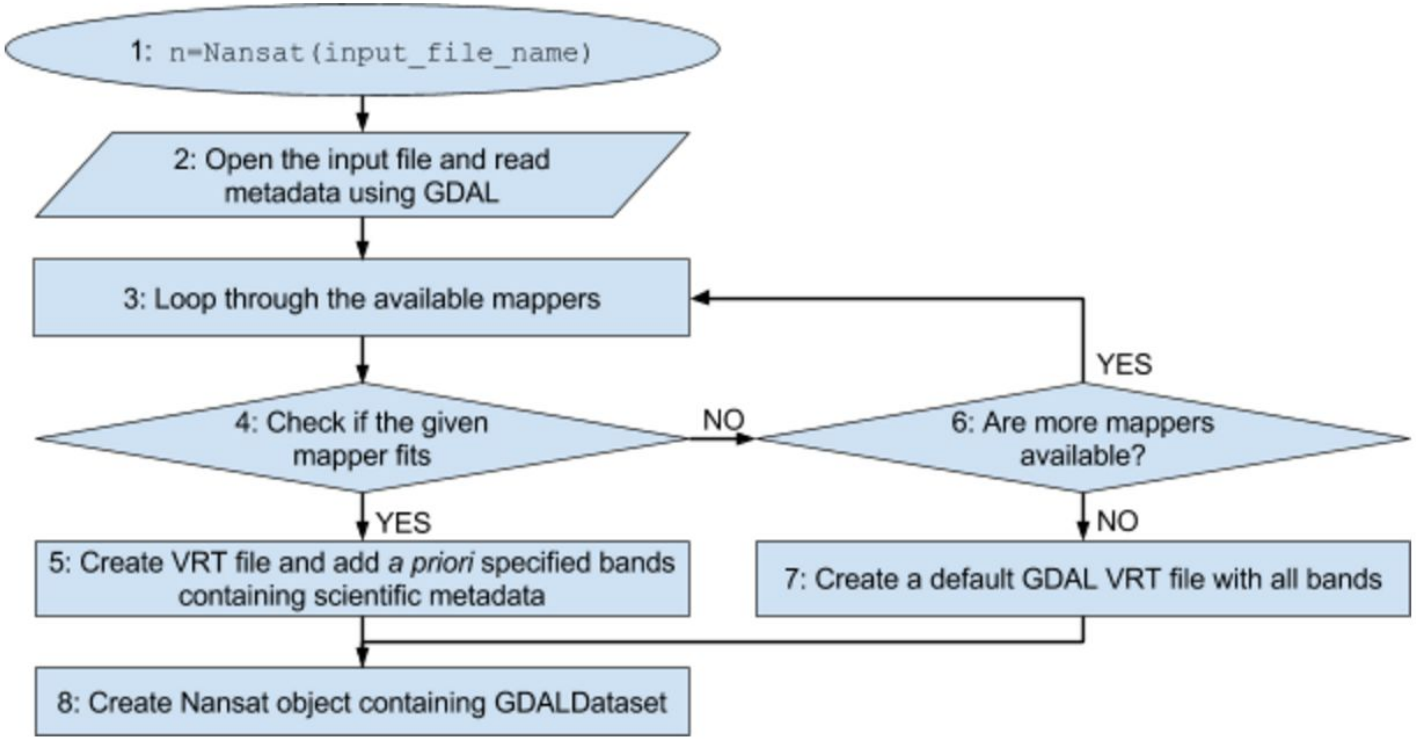
Data can be found and streamed over the internet -
Allows access to pixels rather than images

NMDC norwegian marine data centre
NorDataNet Norwegian Scientific Data Network
Copernicus Europe's eyes on Earth
European Commission
esa
NORMAP
satellittdata.no
GLOBCURRENT
NOAA

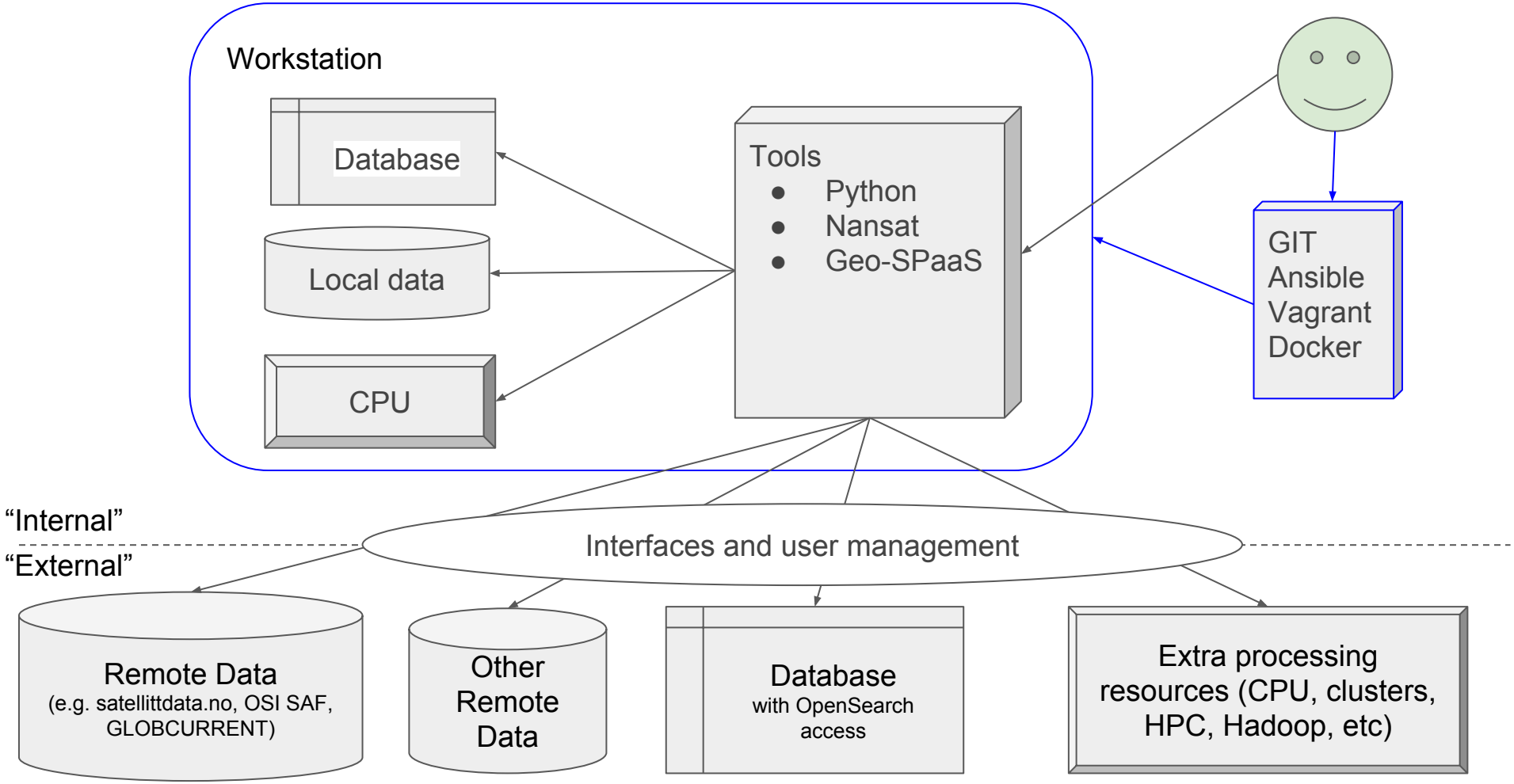
# Nansat - a scientist orientated Python package for geospatial data processing

- https://github.com/nansencenter/nansat
- Bygger på GDAL, men legger til beskrivelser av datasettene i form av metadata
  - Kombinerer allsidigheten og styrken til GDAL med kunnskap om og beskrivelse av datainnholdet
  - Støtter over 40 ulike formater fra satellitt og numeriske modeller
  - Støtter ethvert griddet datasett som følger NetCDF-CF-standarden
- “Lazy operations” - operasjoner beskrives i xml og utføres først når dataene skal leses ut (f.eks. som numpy.array)
  - Reprojisering
  - Subsampling
  - Cropping
- Tillater strømming av data fra OpenDAP

# Workflow


1: n=Nansat(input_file_name)
2: Open the input file and read metadata using GDAL
3: Loop through the available mappers
4: Check if the given mapper fits
NO
6: Are more mappers available?
YES
YES
NO
5: Create VRT file and add *a priori* specified bands containing scientific metadata
7: Create a default GDAL VRT file with all bands
8: Create Nansat object containing GDALDataset

# Geo-scientific Platform as a Service (Geo-SPaaS)



# Adding metadata to Geo-SPaaS catalog

1: SSH to virtual machine (VM):

```
$> vagrant ssh <vm-name>
```

2: Add metadata to database:

```
$> cd ~/geospaas_project/
$> ./manage.py ingest <path to dataset>
```

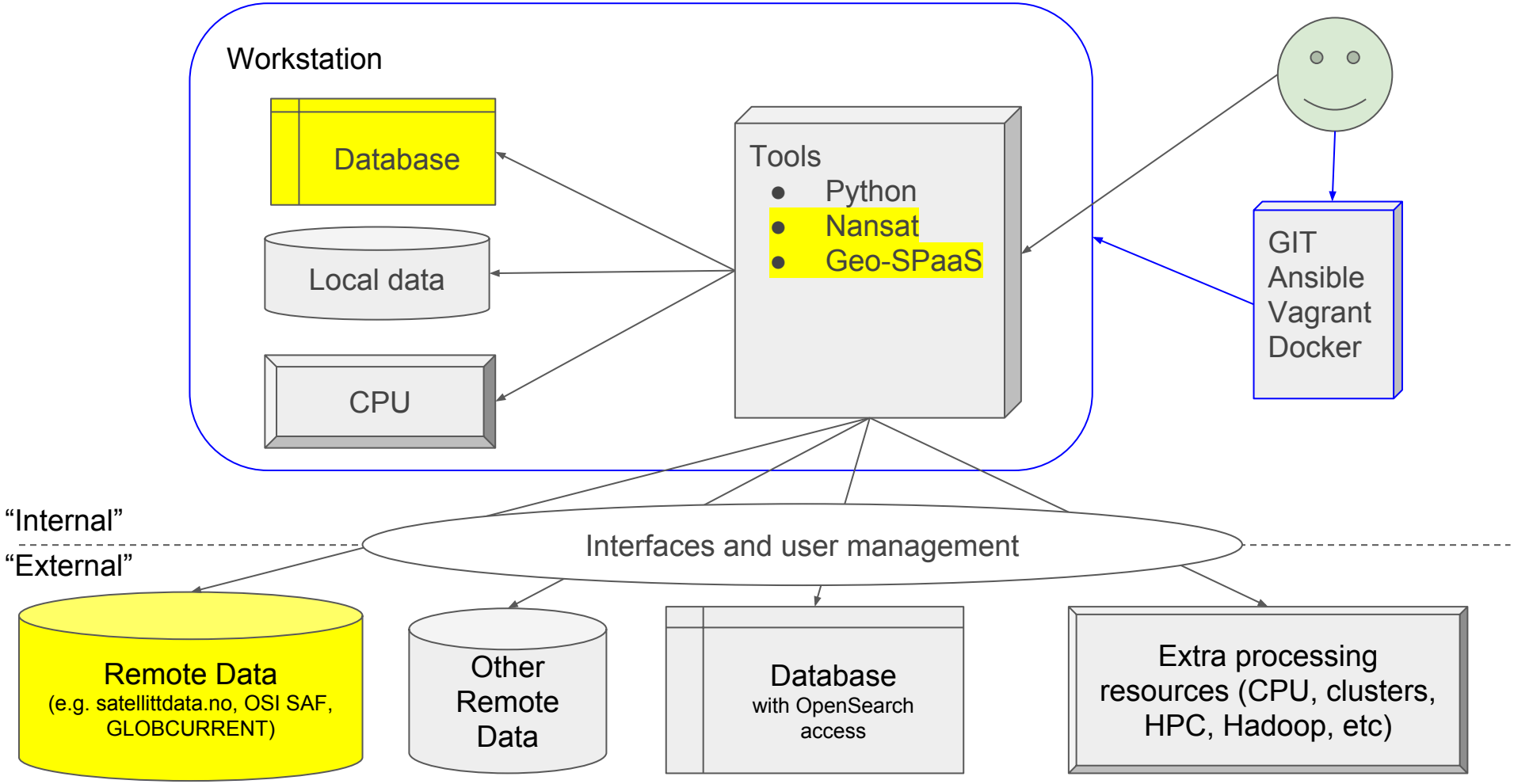
Works on all data that can be read with Nansat

4: Start IPython shell to search, access, and analyse data:

```
$> ./manage.py shell
```

5: Or go to http://192.168.33.15:8888/ to try the Jupyter Notebooks (psw: vagrant)

# ESA GlobCurrent - overflatestrøm fra satellitt vs drivbøyemålinger



# Search, connect and access metadata at Ifremer

Search GlobCurrent repository

```
In [3]: # Find GlobCurrent data after 1st January 2014
        gc_datasets = Dataset.objects.filter(entry_title__contains='globcurrent',
                                             time_coverage_start__gt=timezone.datetime(2014,1,1,tzinfo=timezone.utc))

In [4]: gc_datasets

Out[4]: [<Dataset: JASON-1/POSEIDON-3/2014-01-02T00:00:00+00:00>]

In [5]: # Take the first GlobCurrent dataset, define region of interest, and reproject
        #
        # NOTE:
        #       - only metadata is transferred from the source (in this case the Ifremer Hyrax server)
        #       - operations are mapped in a vrt file (xml)
        n = Nansat(gc_datasets[0].dataseturi_set.all()[0].uri)
        d = Domain(NSR().wkt, '-te 10 -44 40 -30 -tr 0.125 0.125')
        n.reproject(d, addmask=False)

        /home/vagrant/miniconda/lib/python2.7/site-packages/nansat/mappers/opendap.py:89: UserWarning: Time consuming loading
        time from OpenDAP...
          warnings.warn('Time consuming loading time from OpenDAP...')

        miniconda/lib/python2.7/site-packages/nansat/mappers/opendap.py:91: UserWarning: Loading time - OK!
        n('Loading time - OK!')
```

GLOBCURRENT

# Search, connect and access metadata at Ifremer

Search GlobCurrent repository

```
In [3]: # Find GlobCurrent data after 1st January 2014
        gc_datasets = Dataset.objects.filter(entry_title__contains='globcurrent',
                                             time_coverage_start__gt=timezone.datetime(2014,1,1,tzinfo=timezone.utc))
```

```
In [4]: gc_datasets

Out[4]: [<Dataset: JASON-1/POSEIDON-3/2014-01-02T00:00:00+00:00>]
```

```
In [5]: # Take the first GlobCurrent dataset, define region of interest, and reproject
        #
        # NOTE:
        #       - only metadata is transferred from the source (in this case the Ifremer Hyrax server)
        #       - operations are mapped in a vrt file (xml)
        n = Nansat(gc_datasets[0].dataseturi_set.all()[0].uri)
        d = Domain(NSR().wkt, '-te 10 -44 40 -30 -tr 0.125 0.125')
        n.reproject(d, addmask=False)
```

Read metadata

```
/home/vagrant/miniconda/lib/python2.7/site-packages/nansat/mappers/opendap.py:89: UserWarning: Time consuming loading
time from OpenDAP...
  warnings.warn('Time consuming loading time from OpenDAP...')

miniconda/lib/python2.7/site-packages/nansat/mappers/opendap.py:91: UserWarning: Loading time - OK!
n('Loading time - OK!')
```

GLOBCURRENT

# Search, connect and access metadata at Ifremer

Search GlobCurrent repository

```
In [3]: # Find GlobCurrent data after 1st January 2014
        gc_datasets = Dataset.objects.filter(entry_title__contains='globcurrent',
                                             time_coverage_start__gt=timezone.datetime(2014,1,1,tzinfo=timezone.utc))
```

```
In [4]: gc_datasets

Out[4]: [<Dataset: JASON-1/POSEIDON-3/2014-01-02T00:00:00+00:00>]
```

```
In [5]: # Take the first GlobCurrent dataset, define region of interest, and reproject
        #
        # NOTE:
        #       - only metadata is transferred from the source (in this case the Ifremer Hyrax server)
        #       - operations are mapped in a vrt file (xml)
        n = Nansat(gc_datasets[0].dataseturi_set.all()[0].uri)
        d = Domain(NSR().wkt, '-te 10 -44 40 -30 -tr 0.125 0.125')
        n.reproject(d, addmask=False)
```

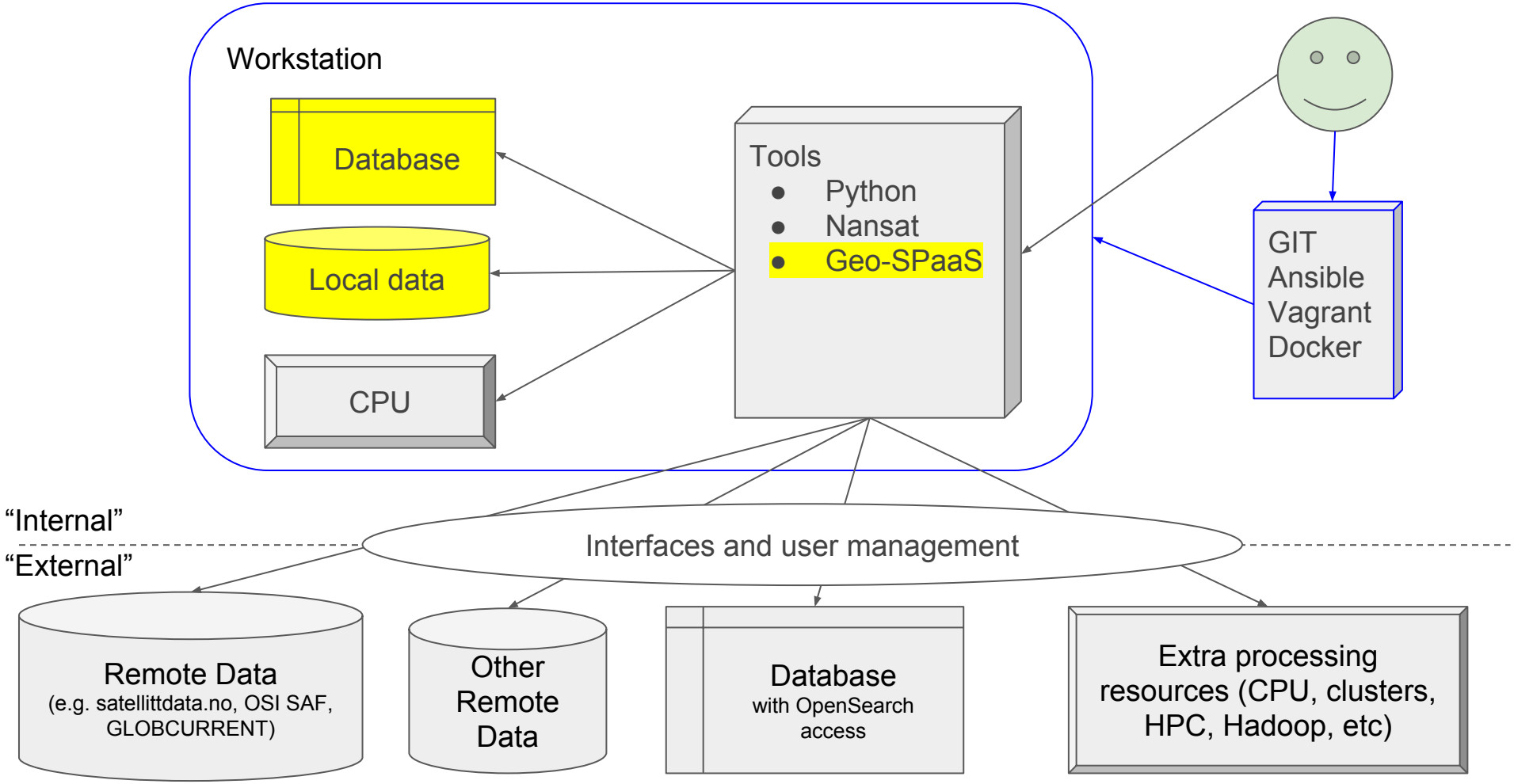
Read metadata

Create xml mapping to region of interest

```
/home/vagrant/miniconda/lib/python2.7/site-packages/nansat/mappers/opendap.py:89: UserWarning: Time consuming loading
time from OpenDAP...
  warnings.warn('Time consuming loading time from OpenDAP...')

miniconda/lib/python2.7/site-packages/nansat/mappers/opendap.py:91: UserWarning: Loading time - OK!
n('Loading time - OK!')
```

GLOBCURRENT

# GlobCurrent Use Case - collocate with surface drifters



# Collocate GlobCurrent with surface drifters

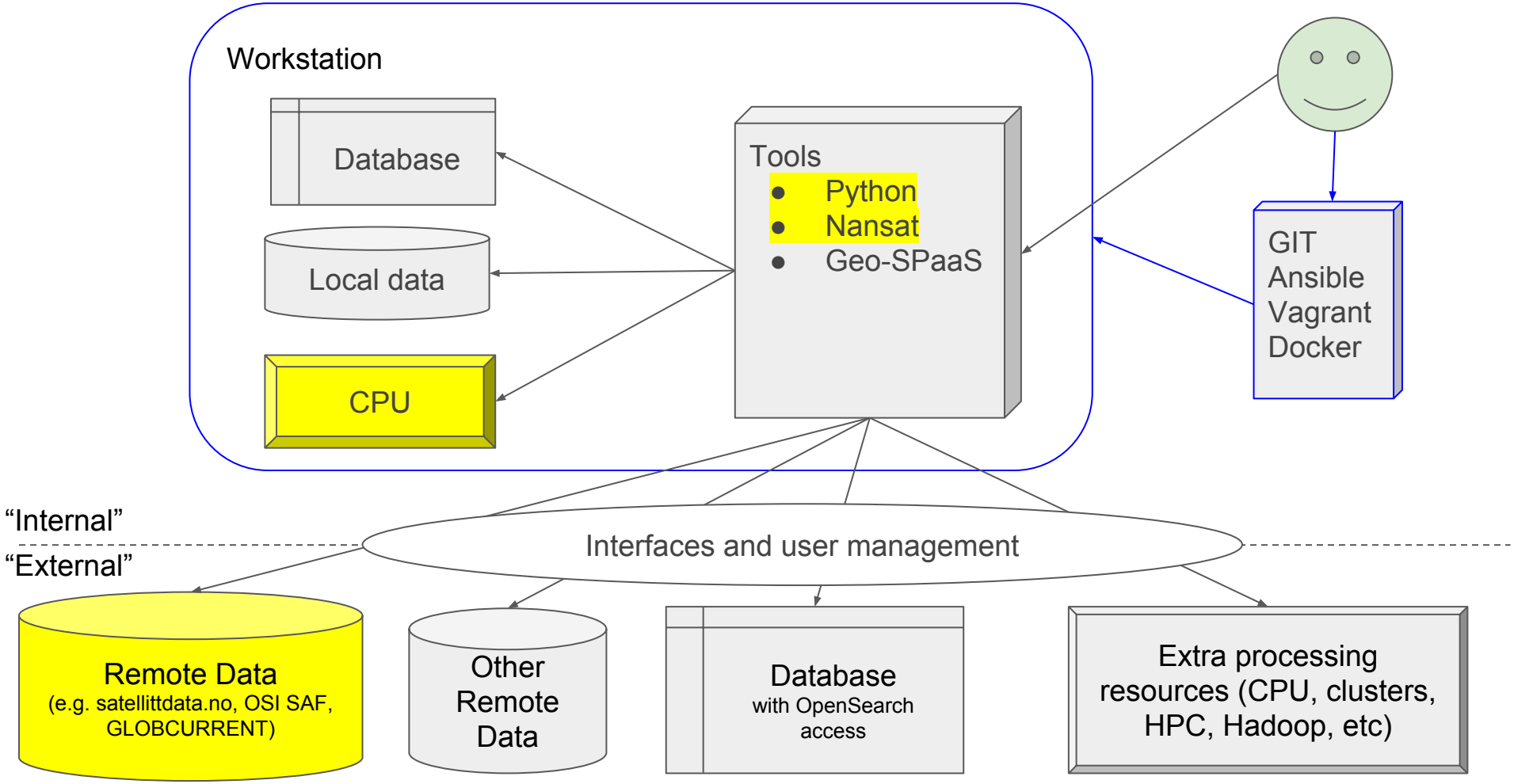
In [6]:
```
# Search for drifters in the same region (defined at lon={0,360}) and time
drifterdomain = Domain(NSR().wkt, '-te 10 -44 40 -30 -tr 0.125 0.125')
geometry = WKTReader().read(drifterdomain.get_border_wkt(nPoints=1000))
drifters = Dataset.objects.filter(entry_title__contains='drifter',
                                  time_coverage_start__range=[n.time_coverage_start-timezone.timedelta(days=15),
                                      n.time_coverage_end+timezone.timedelta(days=15)],
                                  geographic_location__geometry__intersects=geometry)

print len(drifters)
```

Drifters at same time and location as the GlobCurrent dataset

```
/home/vagrant/miniconda/lib/python2.7/site-packages/nansat/domain.py:681: UserWarning: > 180 deg correction to longit
udes - disabled..
  warnings.warn("> 180 deg correction to longitudes - disabled..")

/home/vagrant/miniconda/lib/python2.7/site-packages/django/db/models/fields/__init__.py:1453: RuntimeWarning: DateTim
eField Dataset.time_coverage_start received a naive datetime (2013-12-18 00:00:00) while time zone support is active.
  RuntimeWarning)

/home/vagrant/miniconda/lib/python2.7/site-packages/django/db/models/fields/__init__.py:1453: RuntimeWarning: DateTim
eField Dataset.time_coverage_start received a naive datetime (2014-01-18 00:00:00) while time zone support is active.
  RuntimeWarning)

16
```

GLOBCURRENT

# GlobCurrent Use Case - access remote data and plot



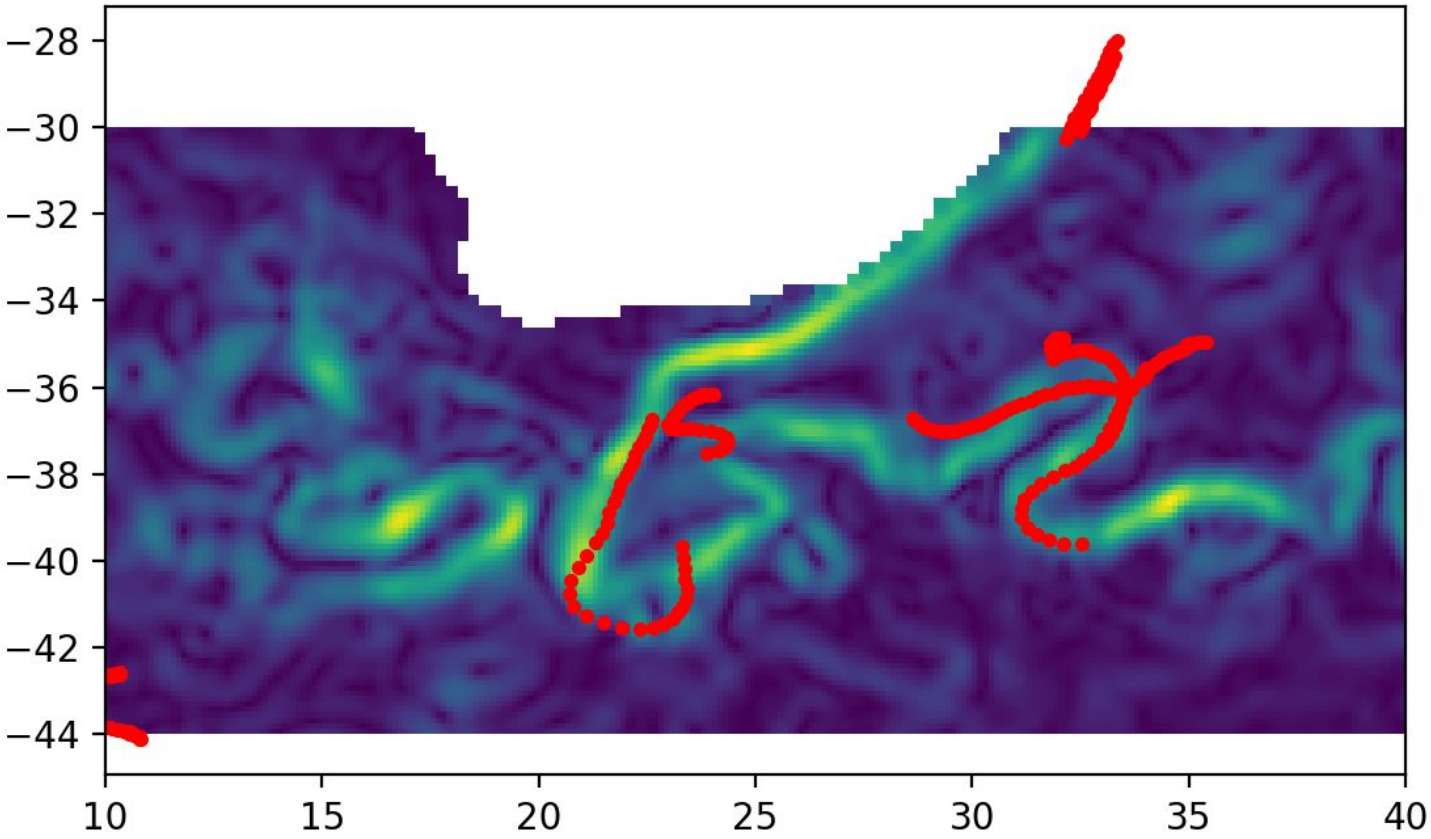
# Access remote GlobCurrent data and plot

In [7]:
```
# Get zonal and meridional geostrophic current
#
# NOTE:
#       - the xml-mapped operations are now performed
#       - actual data is streamed and assigned to variables as numpy arrays
u = n['eastward_geostrophic_current_velocity']
v = n['northward_geostrophic_current_velocity']
```

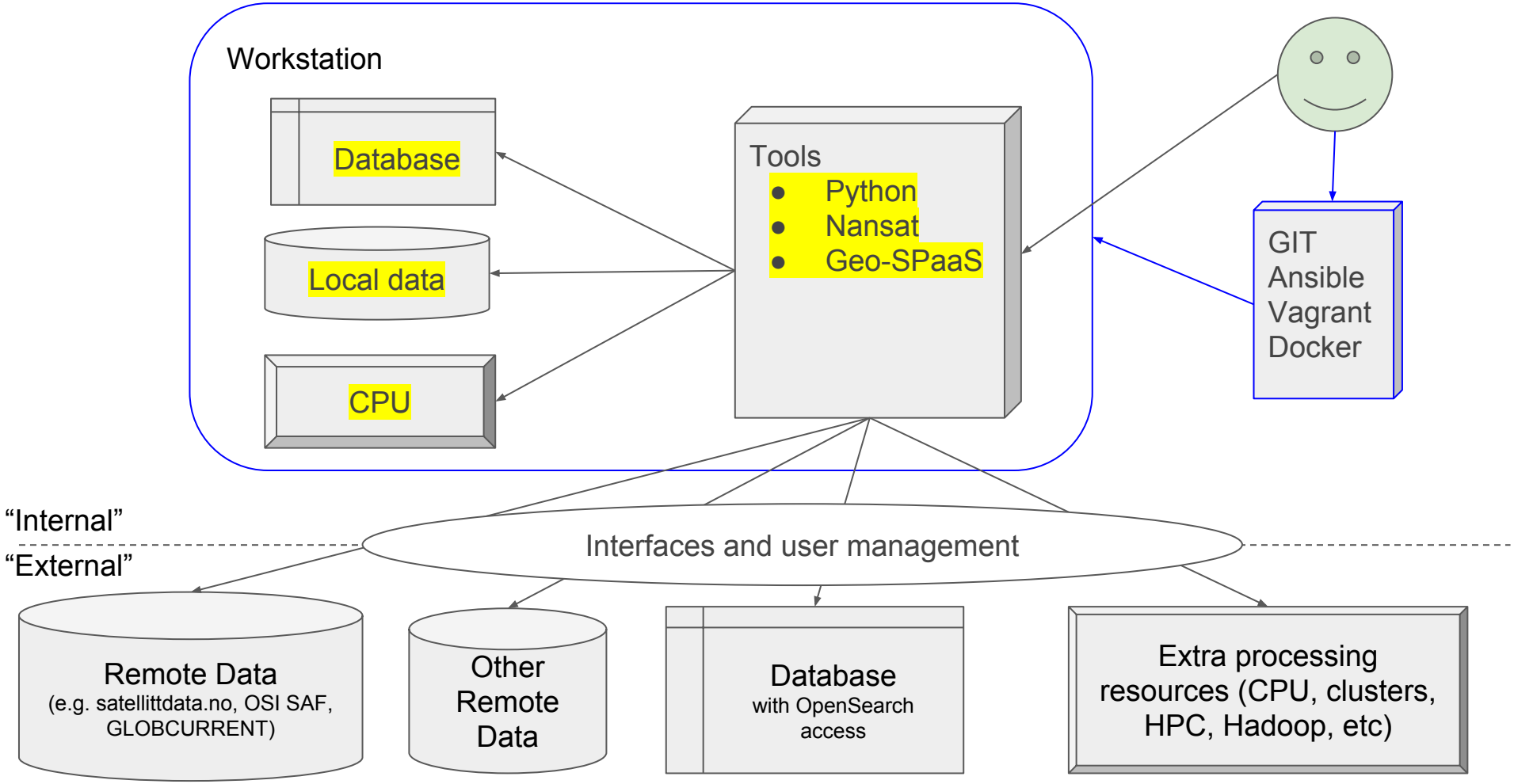
Actual operations are performed here

In [8]:
```
# Plot surface geostrophic current and collocated drifters
plt.figure()
plt.imshow(np.hypot(u, v), extent=(10,40,-44,-30))
for d in drifters:
    try:
        coords = np.array(d.geographic_location.geometry.coords).T
        plt.plot(coords[0], coords[1],'r.')
    except:
        continue
```

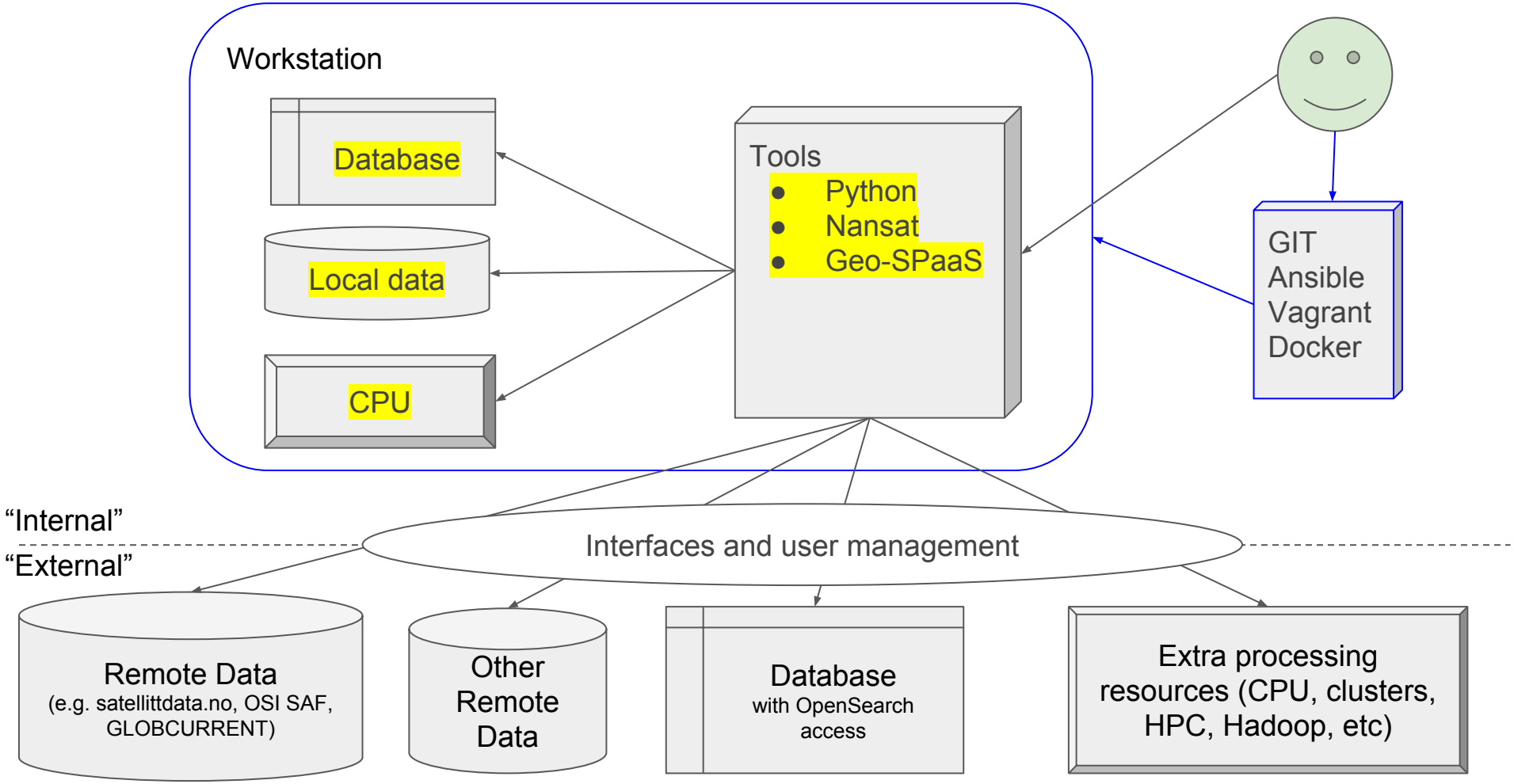
# The Agulhas current



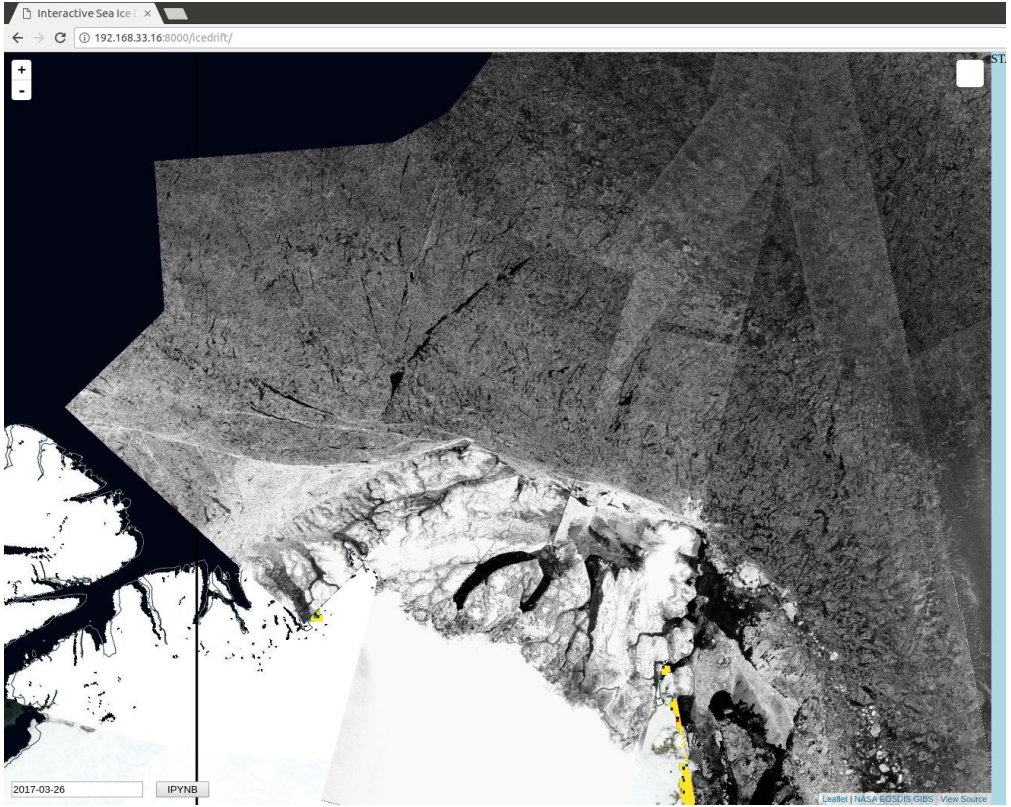
# Sea ice drift from SAR



# Sea ice drift from SAR



# Geo-SPaaS Virtual Machines for sea ice drift



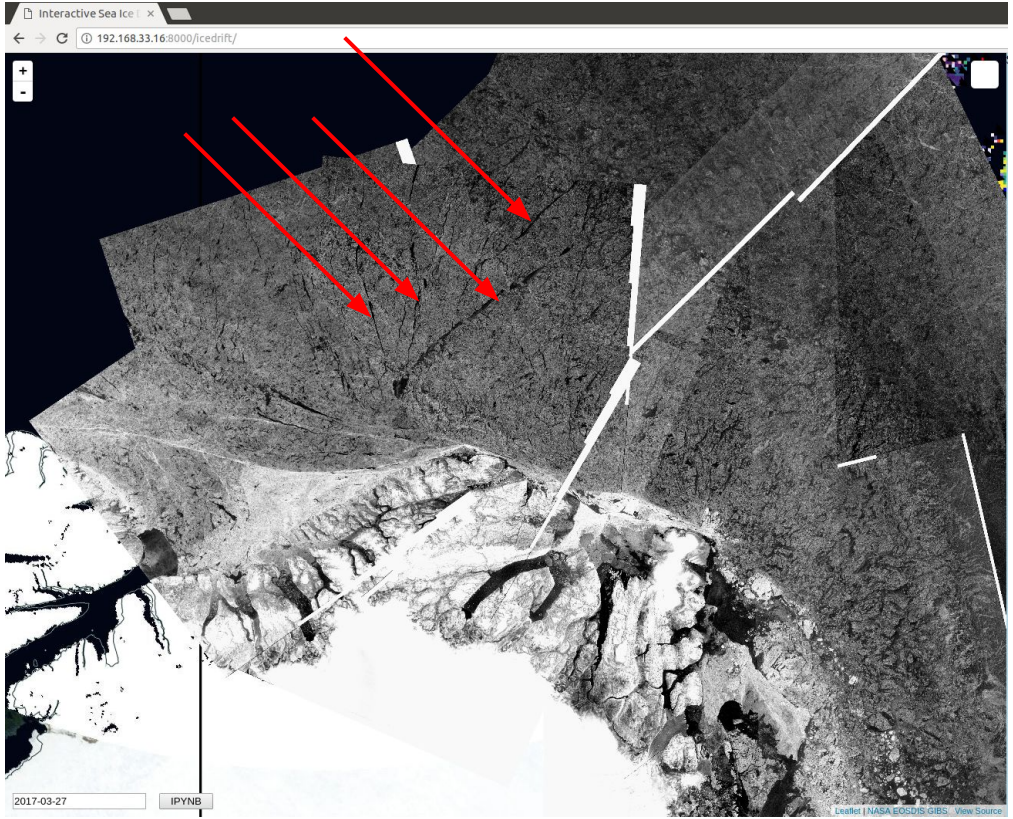

SAR images from Sentinel-1 for **26** March 2017 in a browser-based viewer

**Functionality of `django-sea-ice-drifters`:**

1. Remove thermal noise from Sentinel-1 data
2. Prepare quick-looks for web-browser display
3. **Find pairs of SAR images**
4. Run sea ice drift algorithm for each pair
5. Compute sea ice deformation
6. Create maps of averaged sea ice deformation

Step **#3** is very difficult without a database solution. Geo-SPaaS enables easy data search.

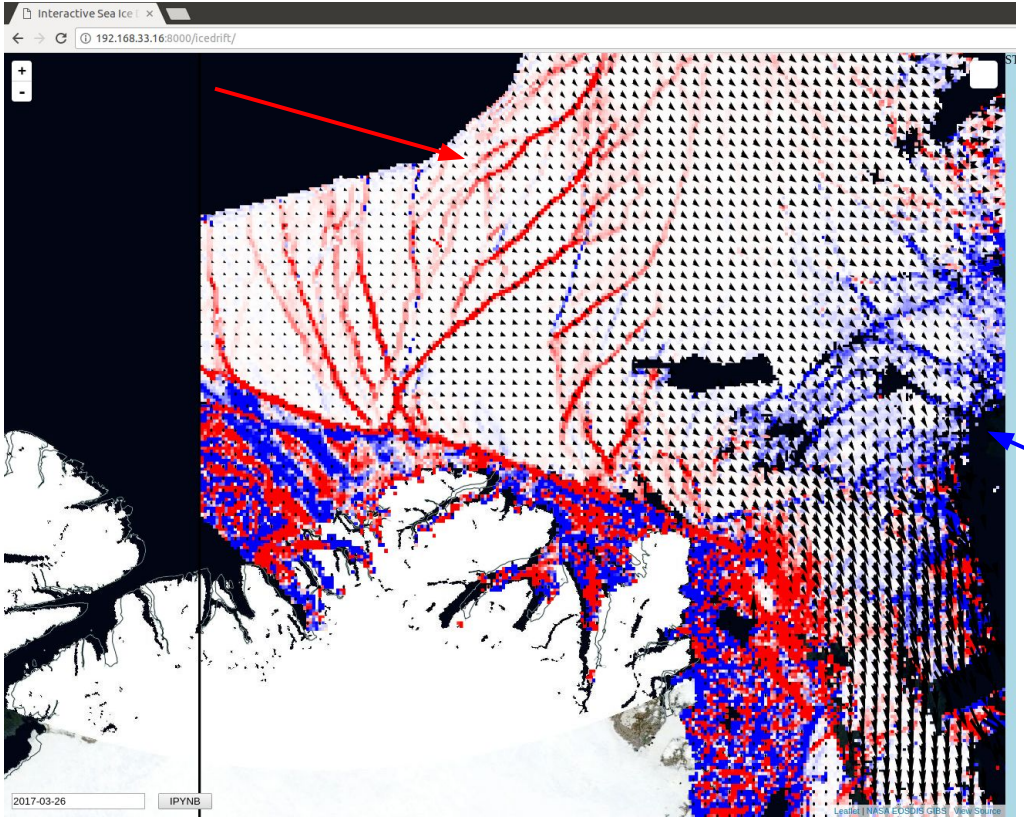
# Geo-SPaaS Virtual Machines for sea ice drift




**Leads** (cracks) in sea ice can be observed as dark lineaments. Leads appear and disappear due to convergence, divergence and shear of sea ice drift flow.

SAR images from Sentinel-1 for **27** March 2017 in a browser-based viewer

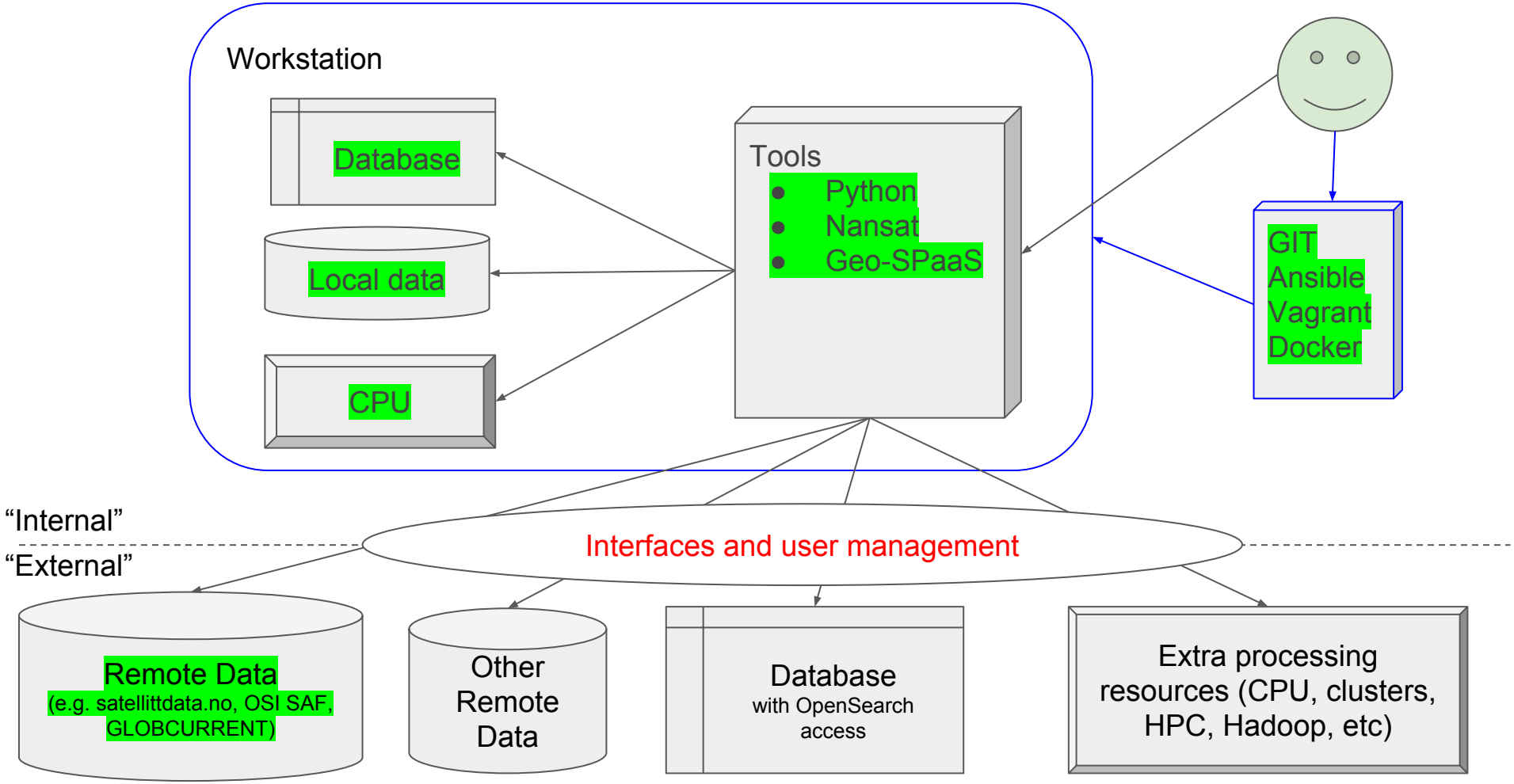
# Geo-SPaaS Virtual Machines for sea ice drift




Deformation of sea ice is derived from sea ice drift and is seen as convergence and divergence zones. They coincide in space with sea ice leads. Weak ice can be detected both by static (detection of leads) and dynamic (deformation of ice motion) approaches.

Ice drift and deformation from Sentinel-1 images for **23 - 27** March 2017

# What we have


Workstation
Database
Local data
CPU
Tools
Python
Nansat
Geo-SPaaS
GIT
Ansible
Vagrant
Docker
"Internal"
"External"
Interfaces and user management
Remote Data
(e.g. satellittdata.no, OSI SAF, GLOBCURRENT)
Other Remote Data
Database
with OpenSearch access
Extra processing resources (CPU, clusters, HPC, Hadoop, etc)

# Summary

New opportunities:

- Open data and standards
- Open-source solutions for provisioning of virtual machines
- Git version control

→ Simplified and improved workflow management and team collaboration

- Data storage is distributed
  - *“moving processing to the data”* is not straight-forward...
- Nansat/GDAL and OPeNDAP allows <u>access to pixels instead of images</u>