

# intake: take the pain out of data access on mistral


- Do you use python and xarray in your daily work on mistral?
- Do you work with common datasets like CMIP5, CMIP6, MiKlip or ICDC observations?
- Do you want to access data on mistral and feel the pain searching for the exact locations?
- Are you annoyed by all different filenames in the ICDC folder or your personal/group observations folder?

If so, then consider [intake-xarray](#) and [intake-esm](#).

## The idea behind intake

Defining and loading data-sets costs time and effort. The data scientist needs to know what data are available, and the characteristics of each data-set, before going to the effort of loading and beginning to analyze some specific data-set. Furthermore, they might need to learn the API of some Python package specific to the target format. The code to do such data loading often makes up the first block of every notebook or script, propagated by copy&paste.

Intake has been designed as a simple layer over other Python libraries to:

- provide a consistent API, so that you can investigate and load all of your data with the same commands, without knowing the details of the backend library
- a simple cataloging system using  [YAML](#) syntax, enabling, for every data-set, a description of
  - which plugin is to load it
  - arguments to pass
  - arbitrary metadata to associate with the data
- transparent access to remote catalogs and data, for most formats
- a minimalist plugin system, so that new loaders, remote containers, and many other components can be contributed to Intake with a minimum of fuss.
- an optional server-client infrastructure, so that you can serve a set of catalogs and stream data that the client doesn't have direct access to.

Source and further reading: <https://www.anaconda.com/intake-taking-the-pain-out-of-data-access/>

## intake-xarray

`intake-xarray` combines `intake` with `xarray`. You can easily access data from various locations and filenames you/someone predefined in a YAML file.

### Example: How to load different observations from ICDC hassle-free into

## xarray?

```
import intake
cat = intake.open_catalog("/home/mpim/m300524/pymistral/intake/obs.yml")
ds = cat['HadCRUT3'].to_dask()
```

- [ICDC observations Notebook](#)
- [ICDC YAML](#)
- [intake-xarray Documentation](#)
- [intake-xarray github](#)

Clone <https://gitlab.dkrz.de/m300524/pymistral> and install the conda environment [pymistral](#) to try out the notebooks yourself.

## intake-esm

intake-esm combines intake-xarray with pandas to make Earth-System-Model output easily accessible. A builder creates a collection, which is pandas.DataFrame from a catalog, which is a json file. Luckily, a few collections are available for [mistral](#). These collections can be searched with queries and directly load ESM output via dask into xarray. Developed at NCAR.

### Example: How to load CMIP6 hassle-free into xarray?

Also possible with other common experiment comparisons: Choose from CMIP5, CMIP6, MiKlip or MPI GE, see [/work/ik1017/Catalogs](#).

```
import intake
col_url = "/work/ik1017/Catalogs/mistral-cmip6.json"
col = intake.open_esm_datastore(col_url)
query = dict(experiment_id='esm-piControl', table_id='0mon',
             variable_id='fgco2', grid_label=['gn', 'gr'])
cat = col.search(**query)
dset_dict = cat.to_dataset_dict(cdf_kwargs={'chunks': {'time': 12*50}})
ds = dset_dict['CMIP.CCCma.CanESM5.esm-piControl.0mon.gn']
```

- [CMIP6 Notebook](#)
- [CMIP6 json](#)
- [intake-esm Documentation](#)
- [intake-esm github](#) [intake-esm-datastore github](#)

## What next?

Do you like the capabilities of intake? Consider writing your own yaml files and share them with your peers.

A collection of ideas how to use intake-esm for your own experiments:

- In case you have several sensitivity experiments, copy the json file `mistral-MPI-GE.json` from `intake-esm-datastore` and modify it so serve your needs.
- Create your own observations yaml file or extend [mine](#).

From:

<https://wiki.mpimet.mpg.de/> - **MPI Wiki**

Permanent link:

[https://wiki.mpimet.mpg.de/doku.php?id=analysis:pot\\_pourri:python:intake](https://wiki.mpimet.mpg.de/doku.php?id=analysis:pot_pourri:python:intake)

Last update: **2022/04/25 09:18**

