(https://binder.pangeo.io/v2/gh/earthcube2020/ec20_banihirwe_etal/master)

Intake / Pangeo Catalog: Making It Easier To Consume Earth's Climate and Weather Data

Authors

- Anderson Banihirwe (https://github.com/andersy005) (National Center for Atmospheric Research)
- Charles Blackmon-Luca (https://github.com/charlesbluca) (Columbia University / Lamont Doherty Earth Observatory)
- Ryan Abernathey (https://github.com/rabernat) (Columbia University / Lamont Doherty Earth Observatory)
- Joseph Hamman (https://github.com/jhamman) (National Center for Atmospheric Research)

Abstract

Computer simulations of the Earth's climate and weather generate huge amounts of data. These data are often persisted on HPC systems or in the cloud across multiple data assets of a variety of formats (netCDF, zarr, etc...). Finding, investigating, loading these data assets into compute-ready data containers costs time and effort. The data user needs to know what data sets are available, the attributes describing each data set, before loading a specific data set and analyzing it.

In this notebook, we demonstrate the integration of data discovery tools such as intake (https://intake.readthedocs.io/en/latest/) and intake-esm (https://intake-esm.readthedocs.io/en/latest/) (an intake plugin) with data stored in cloud optimized formats (zarr). We highlight (1) how these tools provide transparent access to local and remote catalogs and data, (2) the API for exploring arbitrary metadata associated with data, loading data sets into data array containers.

We also showcase the Pangeo catalog (https://catalog.pangeo.io/), an open source project to enumerate and organize cloud optimized climate data stored across a variety of providers, and a place where several intake-esm collections are now publicly available. We use one of these public collections as an example to show how an end user would explore and interact with the data, and conclude with a short overview of the catalog's online presence.

Try this notebook on Pangeo Binder

To try this notebook in your web browser, just click on the binder logo/image:

(https://binder.pangeo.io/v2/gh/earthcube2020/ec20_banihirwe_etal/master)

A static version is also available here (https://nbviewer.jupyter.org/github/earthcube2020/ec20_banihirwe_etal/blob/master/intake-pangeo-catalog.ipynb)

Intake / Pangeo Catalog: Making It Easier To Consume Earth's Climate and Weather Data

Anderson Banihirwe (abanihi@ucar.edu), Charles Blackmon-Luca (blackmon@ldeo.columbia.edu), Ryan Abernathey (rpa@ldeo.columbia.edu), Joseph Hamman (jhamman@ucar.edu)

- NCAR, Boulder, CO, USA
- Columbia University, Palisades, NY, USA

2020 EarthCube Annual Meeting ID: 133

Introduction

Computer simulations of the Earth's climate and weather generate huge amounts of data. These data are often persisted on high-performance computing (HPC) systems or in the cloud across multiple data assets in a variety of formats (netCDF, Zarr, etc.). Finding, investigating, and loading these data assets into compute-ready data containers costs time and effort. The user should know what data are available and their associated metadata, preferably before loading a specific data asset and analyzing it.

In this notebook, we demonstrate intake-esm, a Python package and an intake plugin with aims of facilitating:

- the discovery of earth's climate and weather datasets.
- the ingestion of these datasets into xarray dataset containers.

The common/popular starting point for finding and investigating large datasets is with a data catalog. A *data catalog* is a collection of metadata, combined with search tools, that helps data analysts and other users to find the data they need. For a user to take full advantage of intake-esm, they must point it to an *Earth System Model (ESM) data catalog*. This is a JSON-formatted file that conforms to the ESM collection specification.

ESM Collection Specification

The ESM collection specification provides a machine-readable format for describing a wide range of climate and weather datasets, with a goal of making it easier to index and discover climate and weather data assets. An asset is any netCDF/HDF file or Zarr store that contains relevant data.

An ESM data catalog serves as an inventory of available data, and provides information to explore the existing data assets. Additionally, an ESM catalog can contain information on how to aggregate compatible groups of data assets into singular xarray datasets.

Use Case: CMIP6 hosted on Google Cloud

The Coupled Model Intercomparison Project (CMIP) is an international collaborative effort to improve the knowledge about climate change and its impacts on the Earth System and on our society. CMIP began in 1995, and today we are in its sixth phase (CMIP6). The CMIP6 data archive consists of data models created across approximately 30 working groups and 1,000 researchers investigating the urgent environmental problem of climate change, and will provide a wealth of information for the next Assessment Report (AR6) of the Intergovernmental Panel on Climate Change (IPCC).

Last year, Pangeo partnered with Google Cloud to bring CMIP6 climate data to Google Cloud's Public Datasets program. You can read more about this process here. For the remainder of this section, we will demonstrate intake-esm's features using the ESM data catalog for the CMIP6 data stored on Google Cloud Storage. This catalog resides in a dedicated CMIP6 bucket.

Loading an ESM data catalog

To load an ESM data catalog with intake-esm, the user must provide a valid ESM data catalog as input:

In [1]:
```
import warnings
warnings.filterwarnings("ignore")

import intake

col = intake.open_esm_datastore('https://storage.googleapis.com/cmip6/pangeo-cmip6.json')
col
```

pangeo-cmip6 catalog with 3936 dataset(s) from 269868 asset(s):

	unique
activity_id	15
institution_id	33
source_id	73
experiment_id	103

	unique
member_id	169
table_id	29
variable_id	370
grid_label	10
zstore	269868
dcpp_init_year	60

The summary above tells us that this catalog contains over 268,000 data assets. We can get more information on the individual data assets contained in the catalog by calling the underlying dataframe created when it is initialized:

In [2]:
```
col.df.head()
```

Out[2]:

	activity_id	institution_id	source_id	experiment_id	member_id	table_id	variable_id	grid_label	zstore
0	AerChemMIP	AS-RCEC	TaiESM1	histSST	r1i1p1f1	AERmon	od550aer	gn	gs://cmip6/AerChemMIP/AS-RCEC/TaiESM1/histSST/...
1	AerChemMIP	BCC	BCC-ESM1	histSST	r1i1p1f1	AERmon	mmrbc	gn	gs://cmip6/AerChemMIP/BCC/BCC-ESM1/histSST/r1i...
2	AerChemMIP	BCC	BCC-ESM1	histSST	r1i1p1f1	AERmon	mmrdust	gn	gs://cmip6/AerChemMIP/BCC/BCC-ESM1/histSST/r1i...
3	AerChemMIP	BCC	BCC-ESM1	histSST	r1i1p1f1	AERmon	mmroa	gn	gs://cmip6/AerChemMIP/BCC/BCC-ESM1/histSST/r1i...
4	AerChemMIP	BCC	BCC-ESM1	histSST	r1i1p1f1	AERmon	mmrso4	gn	gs://cmip6/AerChemMIP/BCC/BCC-ESM1/histSST/r1i...

The first data asset listed in the catalog contains:

- the ambient aerosol optical thickness at 550nm (`variable_id='od550aer'`), as a function of latitude, longitude, time,
- in an individual climate model experiment with the Taiwan Earth System Model 1.0 model (`source_id='TaiESM1'`),
- forced by the *Historical transient with SSTs prescribed from historical* experiment (`experiment_id='histSST'`),
- developed by the Taiwan Research Center for Environmental Changes (`instution_id='AS-RCEC'`),
- run as part of the Aerosols and Chemistry Model Intercomparison Project (`activity_id='AerChemMIP'`)

And is located in Google Cloud Storage at `gs://cmip6/AerChemMIP/AS-RCEC/TaiESM1/histSST/r1i1p1f1/AERmon/od550aer/gn/` .

Note: the amount of details provided in the catalog is determined by the data provider who builds the catalog.

Searching for datasets

After exploring the CMIP6 controlled vocabulary, it's straightforward to get the data assets you want using intake-esm's `search()` method. In the example below, we are are going to search for the following:

- variables: `tas` which stands for near-surface air temperature
- experiments: `['historical', 'ssp245', 'ssp585']` :
  - `historical` : all forcing of the recent past.
  - `ssp245` : update of RCP4.5 based on SSP2.
  - `ssp585` : emission-driven RCP8.5 based on SSP5.
- table_id: `Amon` which stands for Monthly atmospheric data.
- grid_label: `gr` which stands for regridded data reported on the data provider's preferred target grid.

For more details on the CMIP6 vocabulary, please check this website.

In [3]:
```
# form query dictionary
query = dict(experiment_id=['historical', 'ssp245', 'ssp585'],
             table_id='Amon',
             variable_id=['tas'],
             member_id = 'r1i1p1f1',
             grid_label='gr')

# subset catalog and get some metrics grouped by 'source_id'
col_subset = col.search(require_all_on=['source_id'], **query)
col_subset.df.groupby('source_id')[['experiment_id', 'variable_id', 'table_id']].nunique()
```

Out[3]:

source_id	experiment_id	variable_id	table_id
CIESM	3	1	1
EC-Earth3	3	1	1
EC-Earth3-Veg	3	1	1

source_id	experiment_id	variable_id	table_id
FGOALS-f3-L	3	1	1
IPSL-CM6A-LR	3	1	1
KACE-1-0-G	3	1	1

Loading datasets

Once you've identified data assets of interest, you can load them into xarray dataset containers using the `to_dataset_dict()` method. Invoking this method yields a Python dictionary of high-level aggregated xarray datasets. The logic for merging/concatenating the query results into higher level xarray datasets is provided in the input JSON file, under `aggregation_control`:

```
"aggregation_control": {
  "variable_column_name": "variable_id",
  "groupby_attrs": [
    "activity_id",
    "institution_id",
    "source_id",
    "experiment_id",
    "table_id",
    "grid_label"
  ],
  "aggregations": [{
      "type": "union",
      "attribute_name": "variable_id"
    },

    {
      "type": "join_new",
      "attribute_name": "member_id",
      "options": {
        "coords": "minimal",
        "compat": "override"
      }
    },
    {
```

```
      "type": "join_new",
      "attribute_name": "dcpp_init_year",
      "options": {
        "coords": "minimal",
        "compat": "override"
      }
    }
  ]
}
```

Though these aggregation specifications are sufficient to merge individual data assets into xarray datasets, sometimes additional arguments must be provided depending on the format of the data assets. For example, Zarr-based assets can be loaded with the option `consolidated=True`, which relies on a consolidated metadata file to describe the assets with minimal data egress.

In [4]:
```
dsets = col_subset.to_dataset_dict(zarr_kwargs={'consolidated': True}, storage_options={'token': 'anon'})

# list all merged datasets
[key for key in dsets.keys()]
```

```
--> The keys in the returned dictionary of datasets are constructed as follows:
        'activity_id.institution_id.source_id.experiment_id.table_id.grid_label'
                                    100.00% [18/18 00:01<00:00]
```

Out[4]:
```
['ScenarioMIP.CAS.FGOALS-f3-L.ssp245.Amon.gr',
 'ScenarioMIP.IPSL.IPSL-CM6A-LR.ssp245.Amon.gr',
 'ScenarioMIP.EC-Earth-Consortium.EC-Earth3.ssp585.Amon.gr',
 'ScenarioMIP.IPSL.IPSL-CM6A-LR.ssp585.Amon.gr',
 'CMIP.IPSL.IPSL-CM6A-LR.historical.Amon.gr',
 'CMIP.EC-Earth-Consortium.EC-Earth3.historical.Amon.gr',
 'ScenarioMIP.NIMS-KMA.KACE-1-0-G.ssp245.Amon.gr',
 'ScenarioMIP.EC-Earth-Consortium.EC-Earth3-Veg.ssp245.Amon.gr',
 'ScenarioMIP.THU.CIESM.ssp585.Amon.gr',
 'CMIP.EC-Earth-Consortium.EC-Earth3-Veg.historical.Amon.gr',
 'ScenarioMIP.THU.CIESM.ssp245.Amon.gr',
 'ScenarioMIP.EC-Earth-Consortium.EC-Earth3.ssp245.Amon.gr',
 'ScenarioMIP.EC-Earth-Consortium.EC-Earth3-Veg.ssp585.Amon.gr',
 'CMIP.NIMS-KMA.KACE-1-0-G.historical.Amon.gr',
 'ScenarioMIP.NIMS-KMA.KACE-1-0-G.ssp585.Amon.gr',
 'ScenarioMIP.CAS.FGOALS-f3-L.ssp585.Amon.gr',
 'CMIP.CAS.FGOALS-f3-L.historical.Amon.gr',
 'CMIP.THU.CIESM.historical.Amon.gr']
```

When the datasets have finished loading, we can extract any of them like we would a value in a Python dictionary:

In [5]:

```
ds = dsets['ScenarioMIP.THU.CIESM.ssp585.Amon.gr']
ds
```

Out[5]: xarray.Dataset

▶ Dimensions: (bnds: 2, **lat**: 192, **lon**: 288, **member_id**: 1, **time**: 1032)

▼ Coordinates:

lon_bnds	(lon, bnds)	float64	dask.array<chunksize=(288, 2), meta=...
lat_bnds	(lat, bnds)	float64	dask.array<chunksize=(192, 2), meta=...
time_bnds	(time, bnds)	object	dask.array<chunksize=(1032, 2), meta...
height	()	float64	...
lon	(lon)	float64	0.0 1.25 2.5 ... 356.2 357.5 358.8
time	(time)	object	2015-01-16 12:00:00 ... 2100-12-16 12...
lat	(lat)	float64	-90.0 -89.06 -88.12 ... 89.06 90.0
member_id	(member_id)	<U8	'r1i1p1f1'

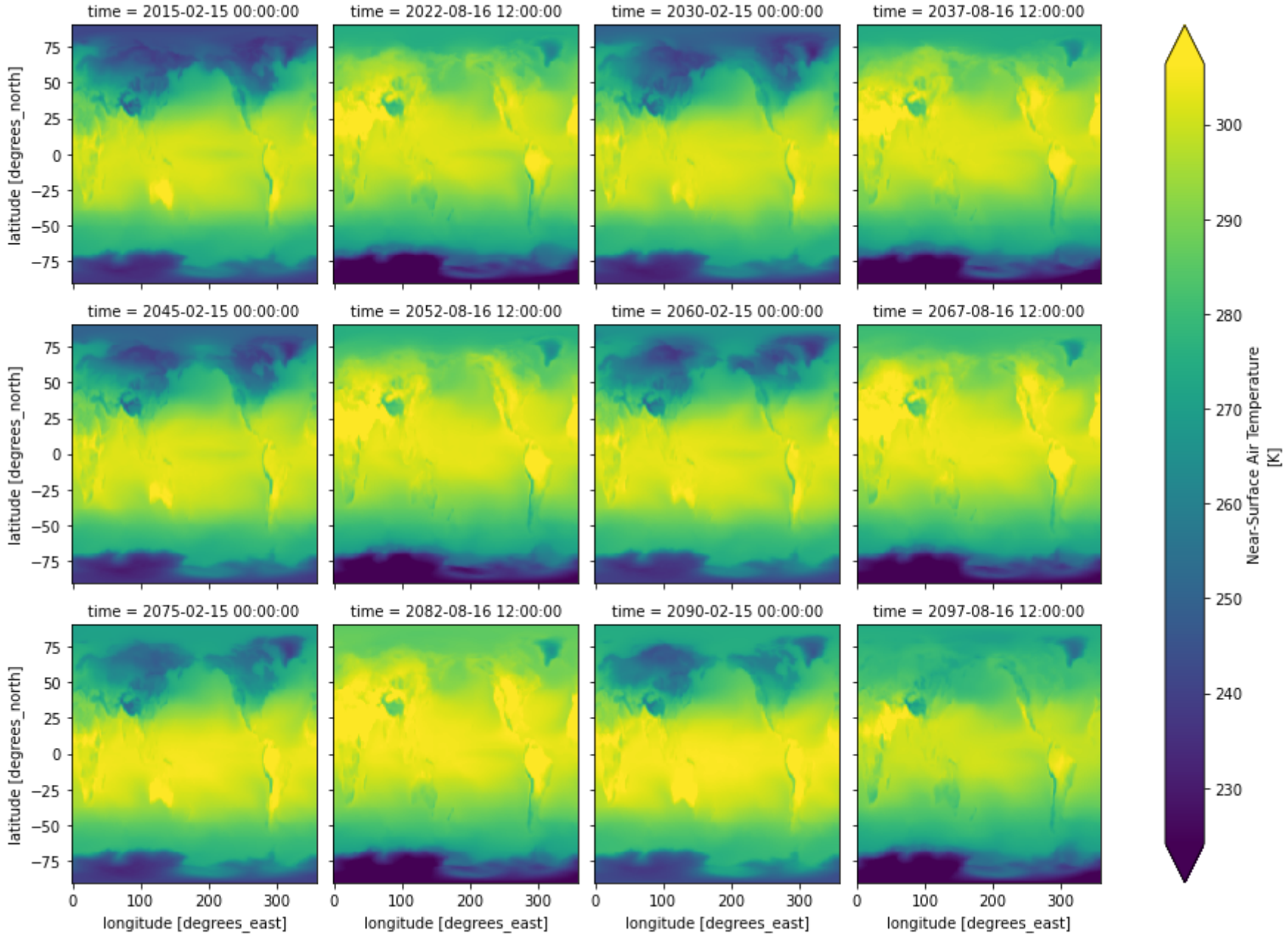
▼ Data variables:

tas	(member_id, time, lat, lon)	float32	dask.array<chunksize=(1, 303, 192, 28...

▶ Attributes: (49)

In [6]:
```
# Let's create a quick plot for a slice of the data:
ds.tas.isel(time=range(1, 1000, 90))\
      .plot(col="time", col_wrap=4, robust=True)
```

Out[6]: <xarray.plot.facetgrid.FacetGrid at 0x7f3f315ca1f0>



Pangeo Catalog

Pangeo Catalog is an open-source project to enumerate and organize cloud-optimized climate data stored across a variety of providers. In addition to offering various useful climate datasets in a consolidated location, the project also serves as a means of accessing public ESM data catalogs.

Accessing catalogs using Python

At the core of the project is a GitHub repository containing several static intake catalogs in the form of YAML files. Thanks to plugins like intake-esm and intake-xarray, these catalogs can contain links to ESM data catalogs or data assets that can be loaded into xarray datasets, along with the arguments required to load them.

By editing these files using Git-based version control, anyone is free to contribute a dataset supported by the available intake plugins. Users can then browse these catalogs by providing their associated URL as input into intake's `open_catalog()`; their tree-like structure allows a user to explore their entirety by simply opening the root catalog and recursively walking through it:

In [7]:
```
cat = intake.open_catalog('https://raw.githubusercontent.com/pangeo-data/pangeo-datastore/master/intake-catalog
entries = cat.walk(depth=5)

[key for key in entries.keys()]
```

Out[7]:
```
['ocean.sea_surface_height',
 'ocean.cesm_mom6_example',
 'ocean.ECCOv4r3',
 'ocean.SOSE',
 'ocean.GODAS',
 'ocean.ECCO_layers',
 'ocean.altimetry.al',
 'ocean.altimetry.alg',
 'ocean.altimetry.c2',
 'ocean.altimetry.e1',
 'ocean.altimetry.e1g',
 'ocean.altimetry.e2',
 'ocean.altimetry.en',
 'ocean.altimetry.enn',
 'ocean.altimetry.g2',
 'ocean.altimetry.h2',
 'ocean.altimetry.j1',
 'ocean.altimetry.j1g',
 'ocean.altimetry.j1n',
 'ocean.altimetry.j2',
 'ocean.altimetry.j2g',
 'ocean.altimetry.j2n',
 'ocean.altimetry.j3',
```

```
'ocean.altimetry.s3a',
'ocean.altimetry.s3b',
'ocean.altimetry.tp',
'ocean.altimetry.tpn',
'ocean.altimetry',
'ocean.LLC4320.LLC4320_grid',
'ocean.LLC4320.LLC4320_SST',
'ocean.LLC4320.LLC4320_SSS',
'ocean.LLC4320.LLC4320_SSH',
'ocean.LLC4320.LLC4320_SSU',
'ocean.LLC4320.LLC4320_SSV',
'ocean.LLC4320',
'ocean.GFDL_CM2_6.GFDL_CM2_6_control_ocean',
'ocean.GFDL_CM2_6.GFDL_CM2_6_control_ocean_surface',
'ocean.GFDL_CM2_6.GFDL_CM2_6_control_ocean_3D',
'ocean.GFDL_CM2_6.GFDL_CM2_6_control_ocean_transport',
'ocean.GFDL_CM2_6.GFDL_CM2_6_control_ocean_boundary_flux',
'ocean.GFDL_CM2_6.GFDL_CM2_6_control_ocean_budgets',
'ocean.GFDL_CM2_6.GFDL_CM2_6_one_percent_ocean',
'ocean.GFDL_CM2_6.GFDL_CM2_6_one_percent_ocean_surface',
'ocean.GFDL_CM2_6.GFDL_CM2_6_one_percent_ocean_3D',
'ocean.GFDL_CM2_6.GFDL_CM2_6_one_percent_ocean_transport',
'ocean.GFDL_CM2_6.GFDL_CM2_6_one_percent_ocean_boundary_flux',
'ocean.GFDL_CM2_6.GFDL_CM2_6_one_percent_ocean_budgets',
'ocean.GFDL_CM2_6.GFDL_CM2_6_grid',
'ocean.GFDL_CM2_6',
'ocean.CESM_POP.CESM_POP_hires_control',
'ocean.CESM_POP.CESM_POP_hires_RCP8_5',
'ocean.CESM_POP',
'ocean.channel.MITgcm_channel_flatbottom_02km_run01_phys-mon',
'ocean.channel.MITgcm_channel_flatbottom_02km_run01_phys_snap15D',
'ocean.channel.channel_ridge_resolutions_01km',
'ocean.channel.channel_ridge_resolutions_05km',
'ocean.channel.channel_ridge_resolutions_20km',
'ocean.channel.run_tracers_restored_zarr',
'ocean.channel',
'ocean.MEOM_NEMO.NATL60_coord',
'ocean.MEOM_NEMO.NATL60_horizontal_grid',
'ocean.MEOM_NEMO.NATL60_vertical_grid',
'ocean.MEOM_NEMO.NATL60_SSH',
'ocean.MEOM_NEMO.NATL60_SSH_1',
'ocean.MEOM_NEMO.NATL60_SSU',
'ocean.MEOM_NEMO.NATL60_SSV',
'ocean.MEOM_NEMO.eNATL60_grid',
'ocean.MEOM_NEMO.eNATL60_BLBT02_SSH',
'ocean.MEOM_NEMO.eNATL60_BLBT02_SSU',
'ocean.MEOM_NEMO.eNATL60_BLBT02_SSV',
'ocean.MEOM_NEMO.eNATL60_BLB002_SSU',
'ocean.MEOM_NEMO.eNATL60_BLB002_SSV',
```

```
'ocean.MEOM_NEMO',
'ocean',
'atmosphere.gmet_v1',
'atmosphere.trmm_3b42rt',
'atmosphere.sam_ngaqua_qobs_eqx_3d',
'atmosphere.sam_ngaqua_qobs_eqx_2d',
'atmosphere.gpcp_cdr_daily_v1_3',
'atmosphere.wrf50_erai',
'atmosphere.era5_hourly_reanalysis_single_levels_sa',
'atmosphere',
'climate.cmip6_gcs',
'climate.GFDL_CM2_6',
'climate.tracmip',
'climate',
'hydro.cgiar_pet',
'hydro.hydrosheds_dir',
'hydro.hydrosheds_acc',
'hydro.hydrosheds_dem',
'hydro.soil_grids_single_level',
'hydro.soil_grids_multi_level',
'hydro.camels.basin_mean_forcing_gcp',
'hydro.camels.basin_mean_forcing_aws',
'hydro.camels.usgs_streamflow_gcp',
'hydro.camels.usgs_streamflow_aws',
'hydro.camels.attributes_aws',
'hydro.camels',
'hydro']
```

The catalogs can also be explored using intake's own `search()` method:

In [8]:
```
cat_subset = cat.search('cmip6')

list(cat_subset)
```

Out[8]:
```
['climate.cmip6_gcs', 'climate.GFDL_CM2_6', 'climate.tracmip', 'climate']
```

Once we have found a dataset or collection we want to explore, we can do so without the need of any user inputted argument:

In [9]:
```
cat.climate.tracmip()
```

pangeo-tracmip catalog with 187 dataset(s) from 7067 asset(s):

	unique
frequency	3

	unique
experiment	11
model	14
variable	47
version	10
source	7067

Accessing catalogs using catalog.pangeo.io

For those who don't want to initialize a Python environmemt to explore the catalogs, catalog.pangeo.io offers a means of viewing them from a standalone web application. The website directly mirrors the catalogs in the GitHub repository, with previews of each dataset or collection loaded on the fly:

PANGEO CATALOG Blog Forum

TRACMIP

master / climate / tracmip

TRACMIP IN PANGEO GOOGLE CLOUD STORAGE

Load in Python

```
from intake import open_catalog

cat = open_catalog("https://raw.githubusercontent.com/pangeo-data/pangeo-datastore/master/intake-catalogs/climate.yaml")
ds  = cat.tracmip()
```

Metadata

```
{
    esmcat_version: 0.1.0,
    id: "pangeo-tracmip",
    description: "TRACMIP zarr stores residing in Pangeo's Google Cloud Storage",
    catalog_file: https://storage.googleapis.com/cmip6/tracmip.csv,
  ▶ attributes: [  5 items  ],
  ▶ assets: {  2 items  },
  ▶ aggregation_control: {  3 items  }
}
```

Dataset Contents

frequency	experiment	model	variable	version	source
A3hr	aqua4xCO2	AM21	hur	v20190116	gs://cmip6/tracmip/A3hr/aq...
A3hr	aqua4xCO2	AM21	hus	v20190116	gs://cmip6/tracmip/A3hr/aq...
A3hr	aqua4xCO2	AM21	ta	v20190116	gs://cmip6/tracmip/A3hr/aq...
A3hr	aqua4xCO2	AM21	ua	v20190116	gs://cmip6/tracmip/A3hr/aq...
A3hr	aqua4xCO2	AM21	va	v20190116	gs://cmip6/tracmip/A3hr/aq...

From here, users can view the JSON input associated with an ESM collection and sort/subset its contents:

frequency	experiment	model	variable	version	source
A3hr	aqua4xCO2	AM21	hur	v20190116	gs://cmip6/tracmip/A3hr/aq...
A3hr	aqua4xCO2	AM21	hus	v20190116	gs://cmip6/tracmip/A3hr/aq...
A3hr	aqua4xCO2	AM21	ta	v20190116	gs://cmip6/tracmip/A3hr/aq...
A3hr	aqua4xCO2	AM21	ua	v20190116	gs://cmip6/tracmip/A3hr/aq...
A3hr	aqua4xCO2	AM21	va	v20190116	gs://cmip6/tracmip/A3hr/aq...
A3hr	aqua4xCO2	AM21	wap	v20190116	gs://cmip6/tracmip/A3hr/aq...
A3hr	aqua4xCO2	AM21	zg	v20190116	gs://cmip6/tracmip/A3hr/aq...
A3hr	aqua4xCO2	CAM4	clivi	v20190409	gs://cmip6/tracmip/A3hr/aq...
A3hr	aqua4xCO2	CAM4	clt	v20190409	gs://cmip6/tracmip/A3hr/aq...
A3hr	aqua4xCO2	CAM4	clwvi	v20190409	gs://cmip6/tracmip/A3hr/aq...
A3hr	aqua4xCO2	CAM4	hfls	v20190409	gs://cmip6/tracmip/A3hr/aq...
A3hr	aqua4xCO2	CAM4	hfss	v20190409	gs://cmip6/tracmip/A3hr/aq...
A3hr	aqua4xCO2	CAM4	hus	v20190409	gs://cmip6/tracmip/A3hr/aq...
A3hr	aqua4xCO2	CAM4	huss	v20190409	gs://cmip6/tracmip/A3hr/aq...
A3hr	aqua4xCO2	CAM4	pr	v20190409	gs://cmip6/tracmip/A3hr/aq...
A3hr	aqua4xCO2	CAM4	prc	v20190409	gs://cmip6/tracmip/A3hr/aq...
A3hr	aqua4xCO2	CAM4	prsn	v20190409	gs://cmip6/tracmip/A3hr/aq...
A3hr	aqua4xCO2	CAM4	prw	v20190409	gs://cmip6/tracmip/A3hr/aq...
A3hr	aqua4xCO2	CAM4	ps	v20190409	gs://cmip6/tracmip/A3hr/aq...

Conclusion

With intake-esm, much of the toil associated with discovering, loading, and consolidating data assets can be eliminated. In addition to making computations on huge datasets more accessible to the scientific community, the package also promotes reproducibility by providing simple methodology to create consistent datasets. Coupled with Pangeo Catalog (which in itself is powered by intake), intake-esm gives climate scientists the means to create and distribute large data collections with instructions on how to use them essentially written into their ESM specifications.

There is still much work to be done with respect to intake-esm and Pangeo Catalog; in particular, goals include:

- Merging ESM collection specifications into SpatioTemporal Asset Catalog (STAC) specification to offer a more universal specification standard
- Development of tools to verify and describe catalogued data on a regular basis
- Restructuring of catalogs to allow subsetting by cloud provider region

Please reach out if you are interested in participating in any way.

References

- intake-esm documentation
- intake documentation
- Pangeo Catalog on GitHub
- Pangeo documentation
- A list of existing, "known" catalogs