



ACCESS-NRI Intake Catalog

Searching and accessing climate model datasets

Anton Steketee, Dougie Squire, Marc White, Charles Turner, Romain Beucher, Aidan Heerdegen, Andy Hogg



Pre-work: Start ARE

Set-up like a normal session for COSIMA Recipes:

Projects:

```
gdata/xp65+gdata/hh5+gdata/ik11+gdata/cj50+gdata/$PROJECT  
(+gdata/oi10+gdata/ol01+gdata/fs38+gdata/p73)
```

Module directories

```
/g/data/hh5/public/modules
```

Modules

```
conda/analysis3
```

Compute Size of large or greater.

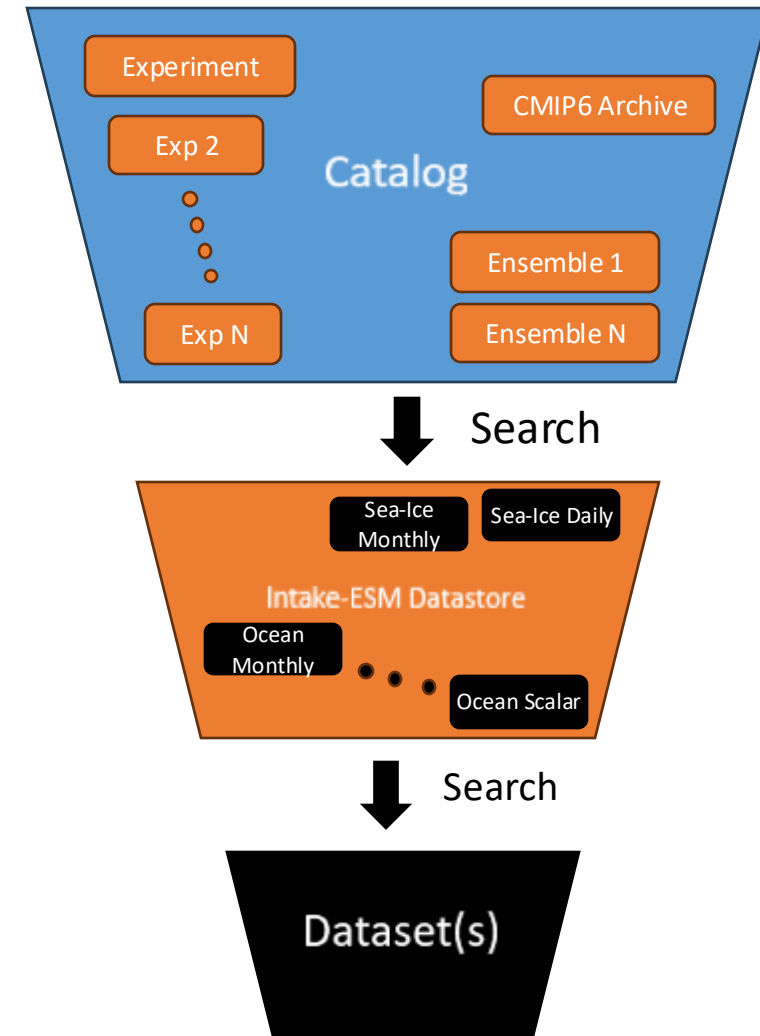
ACCESS-NRI Intake Catalog

A catalog provides functionality for searching, discovering and loading data.

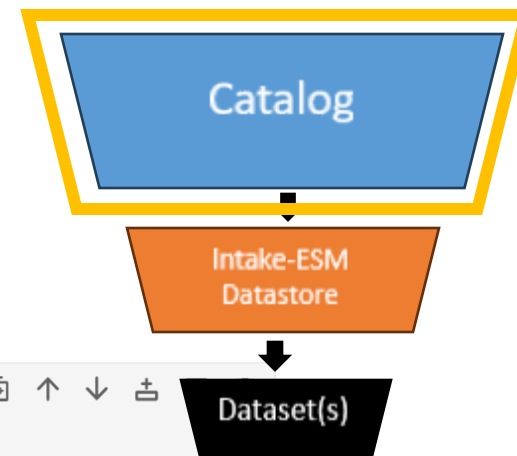
Data can be searched by many attributes, including:

- Experiment name
- Model
- Realm / Model Component
- Data Frequency
- Variable Name
- Variable Standard & Long Names

The ACCESS-NRI Intake Catalog is built upon Intake-ESM, and only shows data stored at the NCI.



ACCESS-NRI Intake Catalog



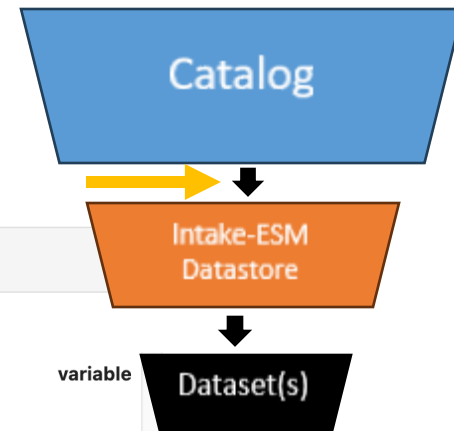
```
[1]: import intake  
catalog = intake.cat.access_nri
```

```
[2]: catalog
```

access_nri catalog with 94 source(s) across 2268 rows:

name	model	description	realm	frequency	variab
01deg_jra55v13_ryf9091	{ACCESS-OM2}	{0.1 degree ACCESS-OM2 global model configuration with JRA55-do v1.3 RYF9091 repeat year forcing (May 1990 to Apr 1991)}	{ocean, sealce}	{3mon, 1mon, fx, 3hr, 1day}	{ANGLET, buoyfreq2_wt, dyu, fsalt_ai_m, tx_tran sss_m, frazil_r total_ocean_fprec, ke_tc total_ocean_lw_hee bih_fric_v, vocn_r frazil_3d_int_z, sig2_r tx_trans_rho, tau_sfc_hflux_fr
01deg_jra55v13_ryf9091	{ACCESS-OM2}	{Cycle 1 of 0.1 degree ACCESS-OM2 global model configuration with JRA55-do v1.3 RYF9091 repeat year forcing (May 1990 to Apr 1991)}	{ocean, sealce}	{1day,	{ANGLET, buoyfreq2_w bmf_u, u, dyu, fsalt_ai_r tx_trans, frazil_r total_ocean_fprec, ke_tc

Experiment filtering and data discovery



```
In [3]: catalog_filtered_example = catalog.search(model="ACCESS-OM2")
catalog_filtered_example
```

Intake dataframe catalog with 76 source(s) across 334 rows:

name	model	description	realm	frequency	variable
01deg_jra55v13_ryf9091	{ACCESS-OM2}	{0.1 degree ACCESS-OM2 global model configuration with JRA55-do v1.3 RYF9091 repeat year forcing (May 1990 to Apr 1991)}	{ocean, sealce}	{3mon, 1mon, fx, 3hr, 1day}	{ANGLET, buoyfreq2_wt, u, dyu, fsalt_ai_m, tx_trans, sss_m, frazil_m, total_ocean_fprec, ke_tot, total_ocean_lw_heat, bih_fric_v, vocn_m, frazil_3d_int_z, sig2_m, tx_trans_rho, tau_y, sfc_hflux_fr...
01deg_jra55v140_iaf	{ACCESS-OM2}	{Cycle 1 of 0.1 degree ACCESS-OM2 global model configuration with JRA55-do v1.4.0 OMIP2 interannual forcing}	{ocean, sealce}	{1day, 1mon, fx}	{ANGLET, buoyfreq2_wt, bmf_u, u, dyu, fsalt_ai_m, tx_trans, frazil_m, total_ocean_fprec, ke_tot, sea_level_sq, total_ocean_lw_heat, bih_fric_v, bottom_temp, frazil_3d_int_z, daidtt_m, tau_y, temp...
01deg_jra55v140_iaf_cycle2	{ACCESS-OM2}	{Cycle 2 of 0.1 degree ACCESS-OM2 global model configuration with JRA55-do v1.4.0 OMIP2 interannual forcing}	{ocean, sealce}	{1day, 1mon, fx}	{dvirdgdt_m, ANGLET, melts, buoyfreq2_wt, fresh_m, bmf_u, u, dyu, fsalt_ai_m, tx_trans, frazil_m, total_ocean_fprec, ke_tot, sea_level_sq, total_ocean_lw_heat, bih_fric_v, bottom_temp,

```
[ ]: catalog.search(model="ACCESS-OM2", frequency="1day", variable="wdet100")
```

```
[ ]: catalog.search(model="ACCESS-OM2", frequency="1day", variable="w.*")
```

```
[6]: catalog.search(model="ACCESS-*").df.model.unique()
```

```
[6]: array([('ACCESS1-0',), ('ACCESS1-3',), ('ACCESS-CM2',),
        ('ACCESS-ESM1-5',), ('ACCESS-OM2',), ('ACCESS-OM2-025',)],
        dtype=object)
```

Each experiment is an Intake-ESM Datastore



```
[12]: catalog['025deg_jra55_iaf_omip2_cycle6']
```

025deg_jra55_iaf_omip2_cycle6 catalog with 8 dataset(s) from 1830 asset(s):

	unique
path	1830
realm	2
variable	296
frequency	3
start_date	855
end_date	854
variable_long_name	271
variable_standard_name	57
variable_cell_methods	6
variable_units	63
filename	1470
file_id	8
derived_variable	0



```
[13]: catalog['025deg_jra55_iaf_omip2_cycle6'].keys()
```

```
[13]: ['iceh_XXXX_XX.1mon',  
       'iceh_XXXX_XX_daily.1day',  
       'ocean_budget.1mon',  
       'ocean_daily.1day',  
       'ocean_grid.fx',  
       'ocean_month.1mon',  
       'ocean_scalar.1mon',  
       'ocean_scalar_snapshot.1day']
```

Refine to one dataset

```
[26]: ocn_1mon_search = catalog['025deg_jra55_iaf_omip2_cycle6'].search(frequency='1mon', realm='ocean')
      ocn_1mon_search
```

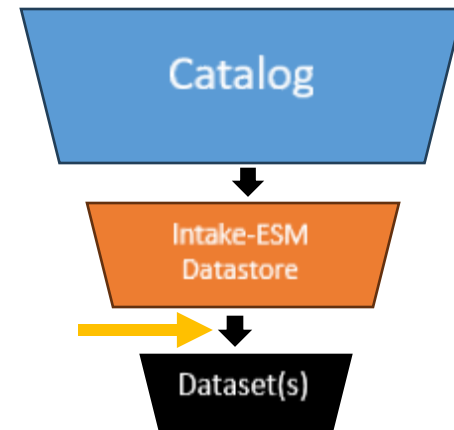
025deg_jra55_iaf_omip2_cycle6 catalog with 3 dataset(s) from 183 asset(s):

	unique
path	183
realm	1
variable	154
frequency	1
start_date	61
end_date	61
variable_long_name	144
variable_standard_name	56
variable_cell_methods	5
variable_units	45
filename	3
file_id	3
derived_variable	0

```
[27]: ocn_1mon_search.keys()
```

```
[27]: ['ocean_budget.1mon', 'ocean_month.1mon', 'ocean_scalar.1mon']
```

```
[28]: ocn_1mon_gridded_search = ocn_1mon_search.search(file_id='ocean_month')
```



Dataframe view helps with data discovery

```
[29]: ocn_1mon_gridded_search.df.variable[0]
```

```
[29]: ['pbot_t',  
      'patm_t',  
      'rho_dzt',  
      'dht',  
      'sea_level',  
      'sea_level_sq',  
      'pot_temp',  
      'temp',  
      'sst',  
      'sst_sq',  
      'bottom_temp',  
      'salt',  
      'sss',  
      'sss_sq',  
      'bottom_salt',  
      'age_global',  
      'mld',  
      'mld_max',  
      'mld_min',  
      'mld_sq',  
      'psiu',  
      'psiv',  
      'bv_freq',  
      'buoyfreq2_wt',  
      'hblt_max',  
      'pot_rho_0',  
      'pot_rho_2',  
      'rho',  
      'eta_t',  
      'u',  
      'v',  
      'wt',  
      'tx_trans']
```


Finding a Variable

```
[30]: sss_search = ocn_1mon_gridded_search.search(variable='sss')
```

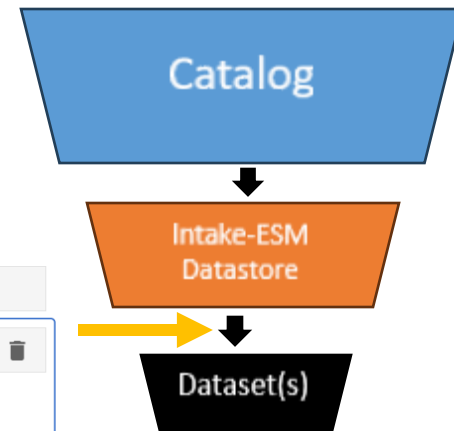
```
[9]: sss_search
```

by647 catalog with 1 dataset(s) from 50 asset(s):

	unique
path	50
realm	1
variable	265
frequency	1
start_date	10
end_date	10
member	5
variable_long_name	260
variable_standard_name	51
variable_cell_methods	4
variable_units	50
filename	10
file_id	1
derived_variable	0

① If we had a priori knowledge that the variable would or might be labelled sss, then we can skip straight to searching for it:

```
[7]: sss_search = catalog['by647'].search(frequency='1mon', variable='sss', file_id='ocean_month')
```



Exercise 1:

In a terminal, from ~ or your gdata folder:

```
$ git clone https://github.com/ACCESS-  
NRI/intake-training/
```

In ARE, open `cosima_exercises_202409.ipynb`

```
import intake  
catalog = intake.cat.access_nri
```

View the catalog:
`catalog`

Search the catalog by the column in the catalog (e.g. model, experiment name, variable etc) and select an experiment

Find a datastore:
`catalog.search()`
`datastore = catalog['experiment']`

View the datastore:
`datastore`
`datastore.df`

Search the datastore by the column in the datastore (e.g. model, experiment name, variable etc) and select at experiment

Find a variable:
`datastore.search()`

Refine your search to reach 1 dataset.





Use `.keys()` to assist in refining

Opening one dataset



```
[34]: from dask.distributed import Client
[35]: client = Client(threads_per_worker=1)
[40]: search = catalog['01deg_jra55v140_iaf'].search(variable='temp_surface_ave')
      search.to_dask()
[40]: xarray.Dataset
```

► Dimensions: (time: 22280, scalar_axis: 1)

▼ Coordinates:

scalar_axis	(scalar_axis)	float64	0.0	 
time	(time)	datetime64[ns]	1958-01-02 ... 2019-01-01	 

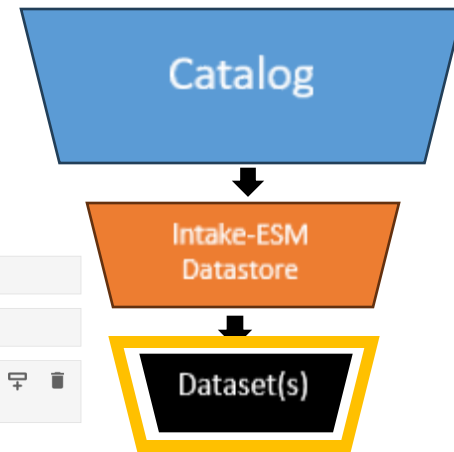
▼ Data variables:

temp_surface_...	(time, scalar_axis)	float64	dask.array<chunks=(1, 1), meta=np....	 
-------------------------	---------------------	---------	---------------------------------------	---

► Indexes: (2)

▼ Attributes:

title : ACCESS-OM2-01
grid_type : mosaic
grid_tile : 1
intake_esm_var... ['temp_surface_ave']
intake_esm_attr... ocean
intake_esm_attr... 1day
intake_esm_attr... ocean_scalar_1_daily_ym_XXXX_XX
intake_esm_attr... netcdf
intake_esm_dat... ocean_scalar_1_daily_ym_XXXX_XX.1day



Datasets can have more than one variable

```
[41]: search = catalog['01deg_jra55v140_iaf'].search(variable=['temp_surface_ave', 'salt_surface_ave'])
```

```
[42]: search
```

01deg_jra55v140_iaf catalog with 1 dataset(s) from 244 asset(s):

unique	
path	244
realm	1
variable	34
frequency	1
start_date	244
end_date	244
variable_long_name	34
variable_standard_name	3
variable_cell_methods	1
variable_units	11
filename	244
file_id	1
derived_variable	0

```
[43]: search.to_dask()
```




```
[43]: xarray.Dataset
```

► Dimensions: (time: 22280, scalar_axis: 1)

▼ Coordinates:

scalar_axis	(scalar_axis)	float64	0.0		
time	(time)	datetime64[ns]	1958-01-02 ... 2019-01-01		

▼ Data variables:

salt_surface_ave	(time, scalar_axis)	float64	dask.array<chunksize=(1, 1), meta=np....		
temp_surface_...	(time, scalar_axis)	float64	dask.array<chunksize=(1, 1), meta=np....		

► Indexes: (2)

► Attributes: (9)

Related variables are often in different datasets

```
[52]: search = catalog['01deg_jra55v140_iaf'].search(variable=['surface_salt', 'surface_temp'], frequency='1mon')
search
```

01deg_jra55v140_iaf catalog with 2 dataset(s) from 488 asset(s):

	unique
path	488
realm	1
variable	6
frequency	1
start_date	244
end_date	244
variable_long_name	6
variable_standard_name	3
variable_cell_methods	2
variable_units	4
filename	488
file_id	2
derived_variable	0

```
[53]: search.keys()
```

```
[53]: ['ocean_2d_surface_salt_1_monthly_mean_ym_XXXX_XX.1mon',
      'ocean_2d_surface_temp_1_monthly_mean_ym_XXXX_XX.1mon']
```

Related variables on the same grid can be merged

```
[54]: search = catalog['01deg_jra55v140_iaf'].search(variable=['surface_salt', 'surface_temp'], frequency='1mon')
```

```
[56]: ds_dict = search.to_dataset_dict()
```

--> The keys in the returned dictionary of datasets are constructed as follows:
'file_id.frequency'

100.00% [2/2 00:08<00:00]







```
[58]: import xarray as xr
```

```
[61]: xr.merge(  
      ds_dict.values(),  
      )
```

```
[61]: xarray.Dataset
```

► Dimensions: (time: 732, yt_ocean: 2700, xt_ocean: 3600)

▼ Coordinates:

xt_ocean	(xt_ocean)	float64	-279.9 -279.8 ... 79.85 79.95	 
yt_ocean	(yt_ocean)	float64	-81.11 -81.07 ... 89.94 89.98	 
time	(time)	datetime64[ns]	1958-01-16T12:00:00 ... 2018...	 

▼ Data variables:

surface_salt	(time, yt_ocean, xt_ocean)	float32	dask.array<chunksize=(1, 540...	 
surface_temp	(time, yt_ocean, xt_ocean)	float32	dask.array<chunksize=(1, 540...	 

► Indexes: (3)

► Attributes: (14)

Experiment extensions can be concatenated

```
[36]: search = catalog.search(name='01deg_jra55v140_iaf_cycle4.*')
```

```
[37]: search
```

Intake dataframe catalog with 2 source(s) across 14 rows:

	name	model	description	realm	frequency	variable
	01deg_jra55v140_iaf_cycle4	{ACCESS-OM2}	{Cycle 4 of 0.1 degree ACCESS-OM2 global model configuration with JRA55-do v1.4.0 OMIP2 interannual forcing}	{sealce, ocean}	{6hr, 3hr, fx, 1day, 1mon}	{Tinz_m, npp_intmld, Sinz_m, tau_y, rhoave, sfc_salt_flux_ice, mld, surface_pot_temp, temp_yflux_adv_int_z, surface_o2, sea_level, divu_h, total_ocean_mh_flux, surface_pot_temp_min, geolat_t, tx_t...
	01deg_jra55v140_iaf_cycle4_jra55v150_extension	{ACCESS-OM2}	{Extensions of cycle 4 of 0.1 degree ACCESS-OM2 + WOMBAT BGC global model configuration with JRA55-do v1.5.0 and v1.5.0.1 interannual forcing}	{sealce, ocean}	{subhr, fx, 1day, 1mon}	{congel, algal_N, diff_cbt_t, total_ocean_swflx_vis, ANGLE, congel_m, src05, skl_Nit, pot_rho_0, sfc_hflux_from_runoff, npp_intmld, dvidtd_m, adic, o2, pprod_gross_2d, divu, aicen_m, ty_trans_int...

```
[38]: datastore_dict = search.to_source_dict()
```

```
[39]: dataset_dict = {
      name: datastore.search(variable="temp_surface_ave").to_dask()
      for name, datastore in datastore_dict.items()
}





ds = xr.concat(
    dataset_dict.values(),
    dim='time'
)
```

```
[40]: ds
```



```
[40]: xarray.Dataset
```

► Dimensions: (time: 23984, scalar_axis: 1)

▼ Coordinates:

scalar_axis	(scalar_axis)	float64	0.0	 
time	(time)	datetime64[ns]	2019-01-02 ... 2019-01-01	 

▼ Data variables:

temp_surface_...	(time, scalar_axis)	float64	dask.array<chunks=(1, 1), meta=np....	 
------------------	---------------------	---------	---------------------------------------	---

► Indexes: (2)

► Attributes: (14)



Exercise 2:

Find 1 dataset:

```
search = datastore.search()
```

Start a dask cluster

Open the dataset:

```
search.to_dask()
```

Try another search which returns two variables.

```
search = datastore.search(variable=[...,...])
```

Do you need to use ?

```
search.to_dataset_dict()
```

Try search with multiple experiments:

```
catalog.search(name='1deg_*')
```

And use `.to_source_dict()`



Ongoing Work

Marc is adding some datasets to the catalog.

- Panantarctic (GFDL-OM4) results need a new *builder*

Request additional datasets at

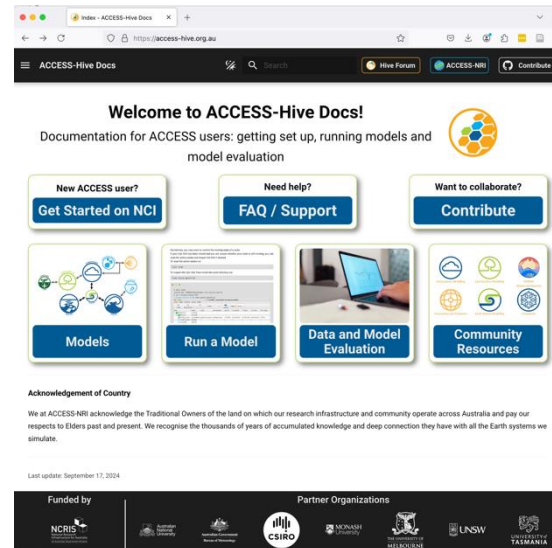
- <https://github.com/ACCESS-NRI/access-nri-intake-catalog>

List of cosima recipes converted to intake and open pull-requests:

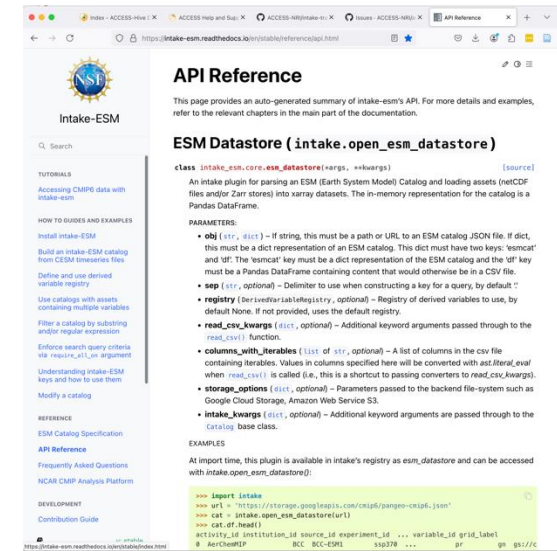
- <https://github.com/COSIMA/cosima-recipes/issues/313>

NCI also have and add intake catalogs (e.g. ERA5, BARRA/BARPA, CMIP6)

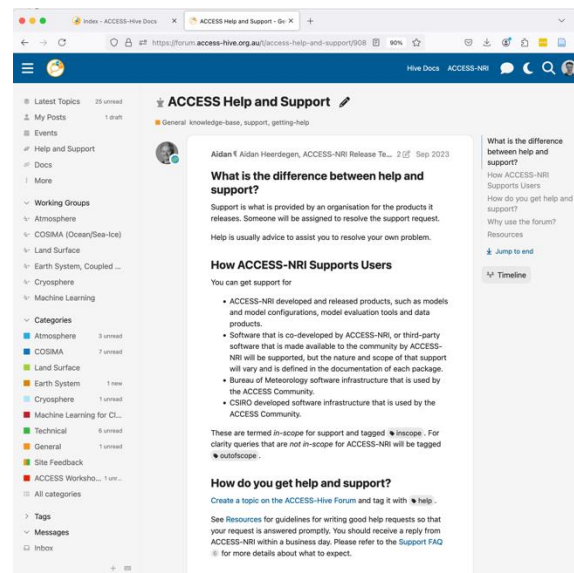
For ACCESS documentation – see ACCESS-HIVE



For intake documentation – see intake-esm website



For general support – use ACCESS-HIVE FORUM



For bugs—use access-nri-intake-catalog github

