



An EERIE Santa Cloud

Fabian Wachsmann – 07.12.2023 – EERIE Science Hour

EERIE funding

This project has received funding from the European Union's Horizon Europe research and innovation programme under Grant Agreement No. 101081383



This work was funded by UK Research and Innovation (UKRI) under the UK government's Horizon Europe funding guarantee (grant number 10040510).

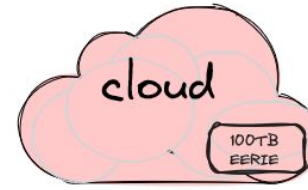


**UK Research
and Innovation**

This work has received funding from the Swiss State Secretariat for Education, Research and Innovation (SERI) under contract #22.00366.

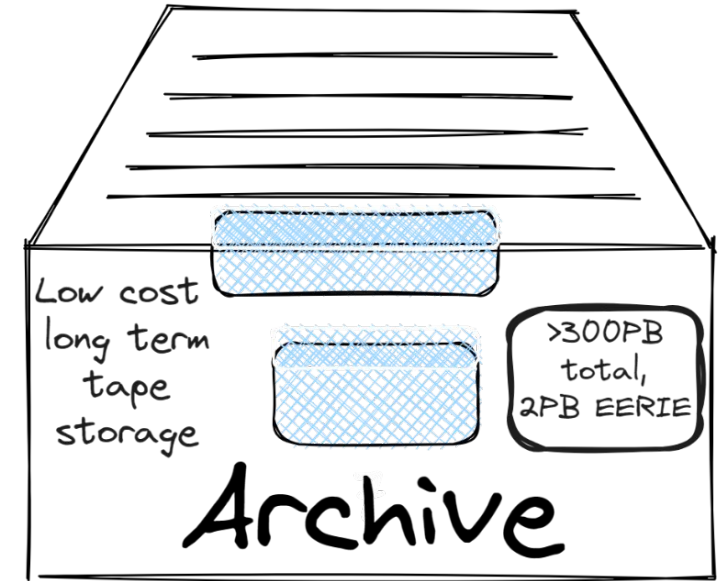
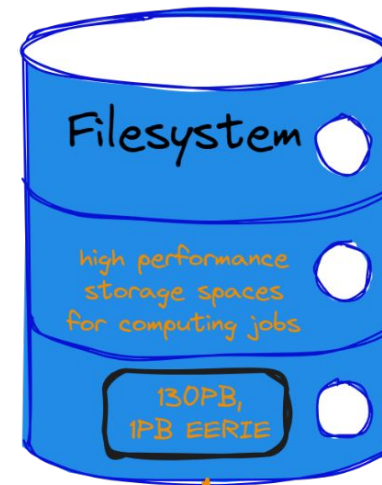
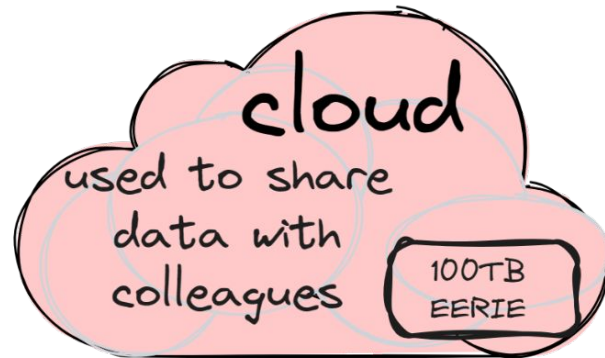


Schweizerische Eidgenossenschaft
Confédération suisse
Confederazione Svizzera
Confederaziun svizra



1. What is cloud storage?
2. How can we make use of institutional cloud storages like DKRZ's swift?
 - a. What is [fsspec](#)?
 - b. What is [Zarr](#)?
 - c. How to host existing climate data in the cloud?
3. EERIE data workflow

3 Types of Storages at DKRZ

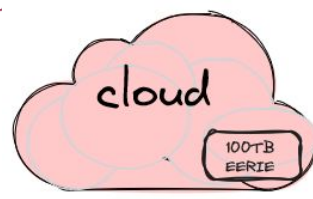


“Oh no! We are running out of disk space again!
But wait! There is still this cloud storage resource...”

The goal 🎯 for today:

You ☐ will be able to use cloud storage wisely!
... if you behave well





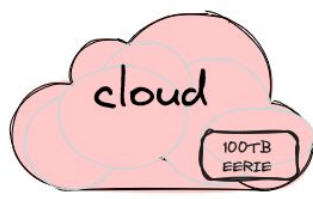
What is a cloud storage? 🤔

A cloud is

- accessible via http 🌐
- an object storage without a file system 📁

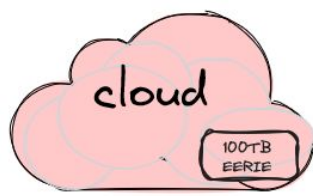
What is the *purpose* of a cloud storage?

- Host the same kind of immutable data without a server:
 - images, videos, music 🎬
 - climate data?

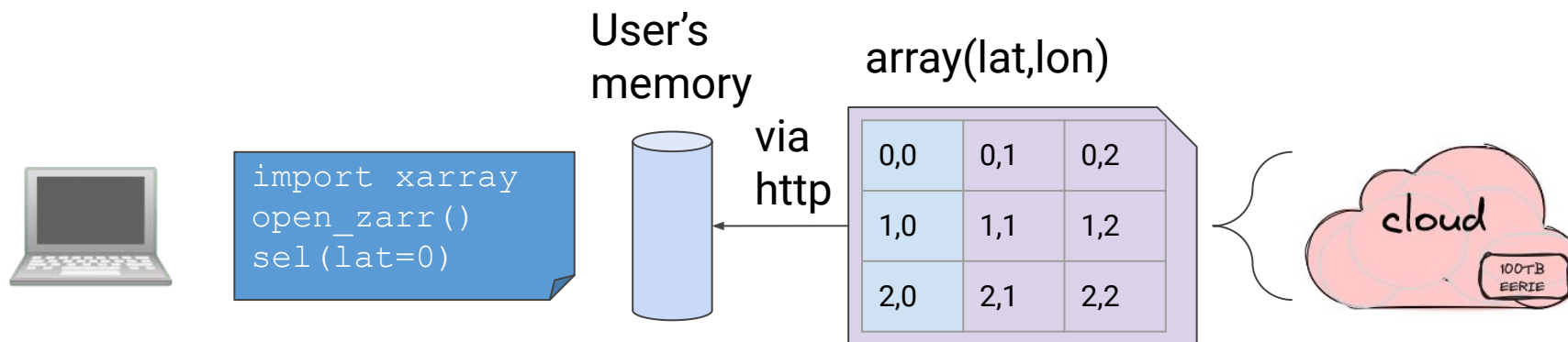


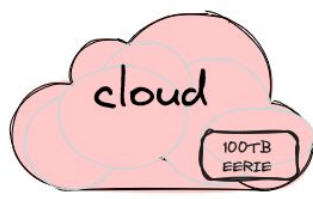
The cloud is no high performance storage like lustre.
But it is **FAIR** to have some climate data in the cloud:

Feature	User advantage
Access protocol is <i>http</i>	Most software is able to understand http -> enhanced Interoperability
No filesystem, but a 🗝 key-value object storage	No namespace conflicts when sharing scripts -> enhanced Reusability
Access control lists allow no authentication	Unrestricted open and FAIR data for everyone
Independent from computational “cloud”	More resilient but no back-up
Object size must be rather small.	Enables more granular access by chunk access



The data workflow: Load small chunks from everywhere to your work environment

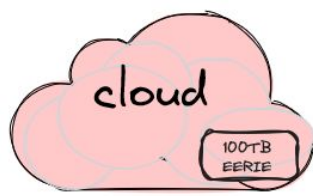




How is DKRZ's swift storage organized if not with a file system?

On only **three** hierarchy levels:

- **Account:**
The top-level of the hierarchy. This can be your user account (own space) or the project id (shared space).
- **Container:**
The only "directory" of a Object Storage. Container can be temporarily and permanently shared with an access control list (ACL).
- **Object:**
Stores bytes of data. Has no individual ACL.



Swift URLs consists of

https://swiftbrowser.dkrz.de/public/dkrz_0b2a0dcc-1430-4a8a-9f25-a6cb8924d92b/cmip6-zarr-2021/

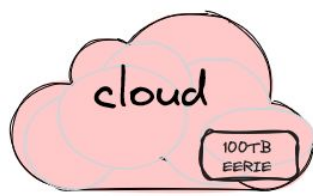
Account hash

Container

https://swift.dkrz.de/v1/dkrz_0b2a0dcc-1430-4a8a-9f25-a6cb8924d92b/cmip6-zarr-2021/

<CMIP6.CMIP.AWI.AWI-CM-1-1-MR.historical.r1i1p1f1.3hr.clt.gn.v20181218/.zattrs>

Object

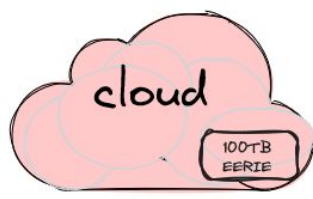


Swift interfaces:

- GUI: swiftbrowser.dkrz.de
- CLI: `module load
py-python-swiftclient/3.12.0-gcc-11.2.0`
- Python: via [swiftspec](#)

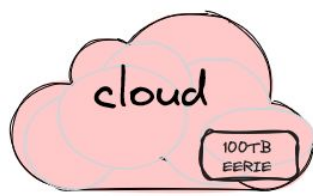
Introduction to fsspec (where the magic happens):

https://github.com/eerie-project/EERIE_hackathon_2023/blob/main/nereus/tutorial_cloud_fsspec.ipynb






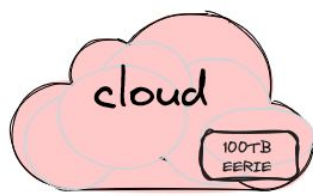
How can we host host useful climate data in the cloud?

- Define robust path templates
 - a move is expensive, so that the location of data is fixed
- Allow zarr to read the cloud data by either create zarr data or map other formats to zarr
 - swift allows only object sizes <5GB, some zarr compressors allow only chunk sizes of <2GB



The cloud-optimised data format [Zarr](#)

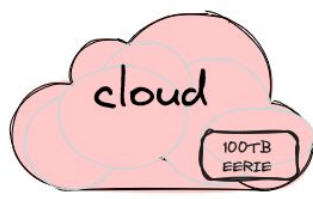
-  enables **http** access with parallel read and writes
-  allows *chunk-based* access to the highest granularity of data so that the volume of data transfer is reduced.
-  saves metadata (comparable to a `ncdump -h`) next to the binary data. This allows software to quickly create a virtual dataset representation without opening and loading data. Metadata across storages can be *consolidated*.



The cloud-optimised data format [Zarr](#)

- can read other formats like netcdf and grib by mapping the files with reference jsons
- Is a backend for NetCDF, see [Nczarr](#)

Zarr - example

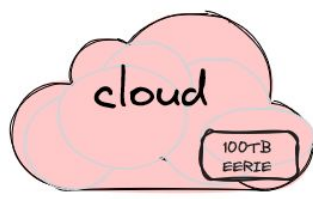


An example Zarr-file from FESOM-NG5 tests:




https://swiftbrowser.dkrz.de/public/dkrz_7fa6baba-db43-4d12-a295-8e3ebb1a01ed/zarr/

Openable with xarray by replacing
“swiftbrowser.dkrz.de/public” with “swift.dkrz.de/v1” in
the url string

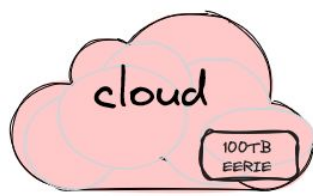
```
import xarray as xr
xr.open_zarr(
    "https://swift.dkrz.de/v1/dkrz_7fa6baba-db43-4d12-a295-8e3ebb1a01ed/zarr/NG5",
)
```



What about non-zarr formatted existing climate data?

1.  Convert into the cloud-native format zarr
2.  Create reference files to map zarr format on netcdf or grib and move all data to the cloud
3. ( “Cloudify” the source storage and use a web server)

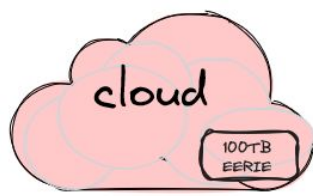
Is 1. worth the effort?



Things we learned about zarr conversion

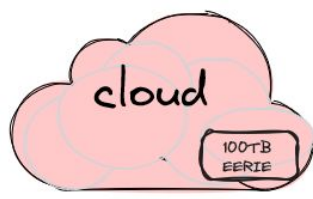
Advantages 😊	Considerations 🤔
best chunking configuration to match access patterns	may duplicate data
highest aggregation possible (subsetting easier than merging)	Conversion can be tricky (check the tool tzis if you need help) <ul style="list-style-type: none">- Rechunking costs compute resources, disk configuration not easy
modular compression and storage (similar to URL chaining)	Completeness of a zarr dataset hard to verify. Metadata is editable.

Into zarr in



Future use of the zarr data:

Keep in mind that small chunk sizes (10MB), as optimal as they are for granular cloud access, can be hard for lustre and cannot be put into archives where each file counts 1GB

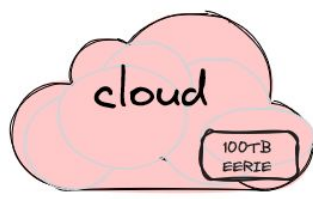


- 1.
2. Create reference files to map zarr format on netcdf or grib and move all data to the cloud

The reference files make use of the “reference file system” implementation of fsspec which is a **file system on a file**: A chunk is mapped to a set consisting of a file, a start byte where to start reading the chunk from the file and an end byte where to stop. E.g.:

```
"x/0": ["s3://bucket/path/file.nc", 294094376, 73825960]
```

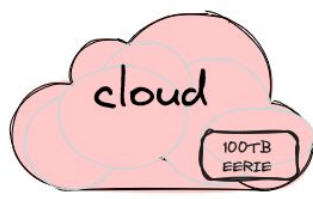
Reference files



For Netcdf, [kerchunk](#) exists. For Grib, we have [gribscan](#).



Advantages	Disadvantage
No data conversion	No cloud optimization: Does the chunk size match the access pattern?

A reference file system, in contrast to pure zarr, moves the chunk indication i.e. the identification of where the chunk data is to the user side and relieves the backend location solver.



Recommendation for bringing existing non-zarr data to the cloud:

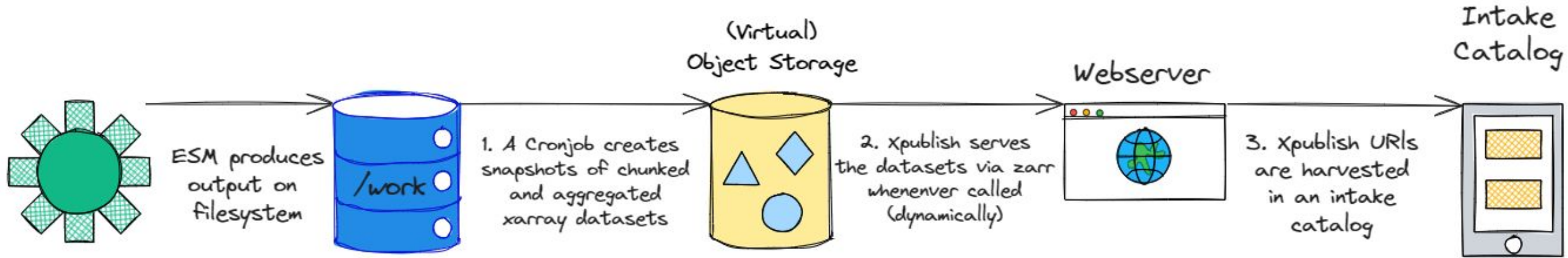
Check if

-  chunk sizes are “ok” (match access patterns and storage requirements)
-  used compression is compatible with the reference file system
- you can live without a disk storage version of the data

If all yes -> Use Option 2

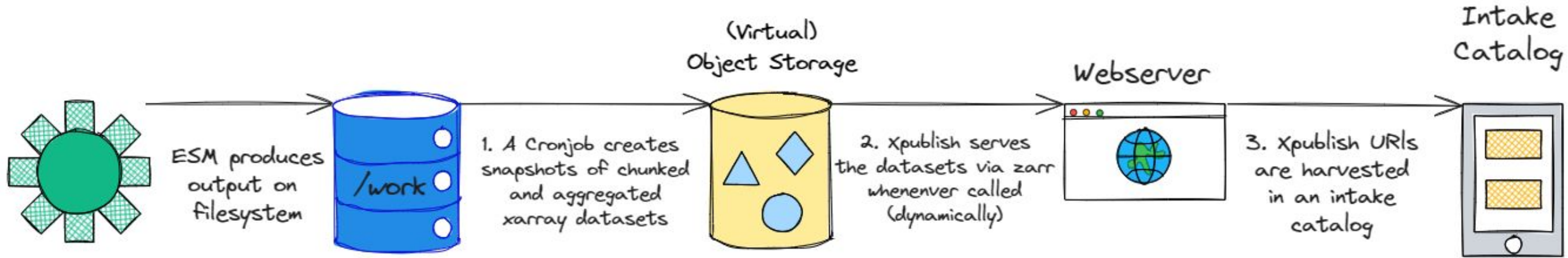
All other cases: Convert to zarr

DKRZ - data workflow



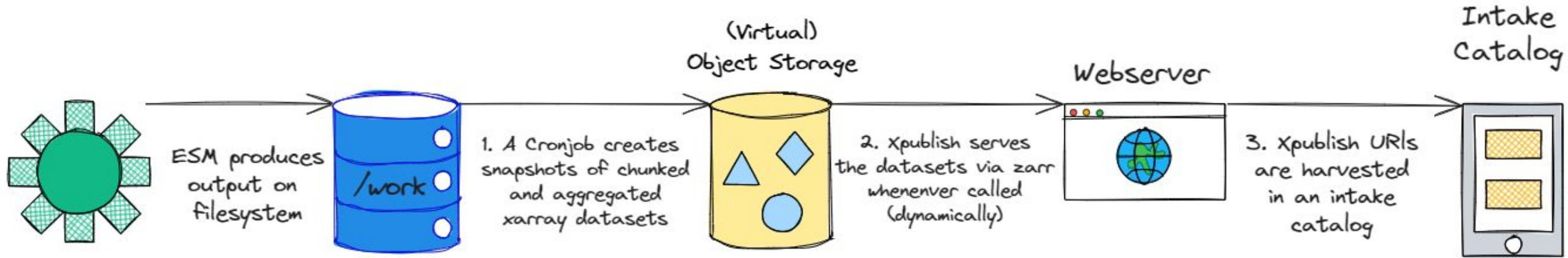
EERIE Data Workflow for Raw NetCDF Output

- Using zarr to open data is helpful because only 1 metadata file needs to be opened instead of e.g. 1000 NetCDF files
→ we create reference files anyway (if not stored in zarr)
- ★ Achieved switching to parquet reference files: O(1MB) for 100TB of data




EERIE Data Workflow for Raw NetCDF Output

- We use data servers to host the project's data to the world. At DKRZ, we use xpublish to host data.
- The intake catalog collects the URLs, intake users do not need to know the backend format



EERIE Data Workflow for Raw NetCDF Output

-  Xpublish allows to add server-side processing for data volume reduction like compression, aggregation or diagnostics

- ★ Extended the eerie.cloud server: <https://eerie.cloud.dkrz.de/docs>

(work in progress)

Easy groupby operations:

- https://eerie.cloud.dkrz.de/datasets/ICON-ESM-ER_eerie-control-1950_atm_2d_1d_min_remap025/groupby/tas/time.month/mean

A interactive plot generator:

- https://eerie.cloud.dkrz.de/datasets/ICON-ESM-ER_eerie-control-1950_oce_2d_1d_mean_remap025/plot/to/image/viridis/time/0_3

- ★ Updated notebooks to access data via intake on
 - [dkrz disk storage](#)
 - including full guideline on intake
 - support plots via GUI
 - with support for intake-esm
 - [dkrz xpublish](#)
 - packages for environment to access
 - what is offered

Call for feedback:

- What obstacles did you experience at the hackathon regarding data access?
- Can we share specific hackathon results like custom diagnostics by hosting the data or the script?
 - What variables are suitable for the cloud?

contact: wachsmannATdkrz.de



Thank you!

contact: wachsmannATdkrz.de