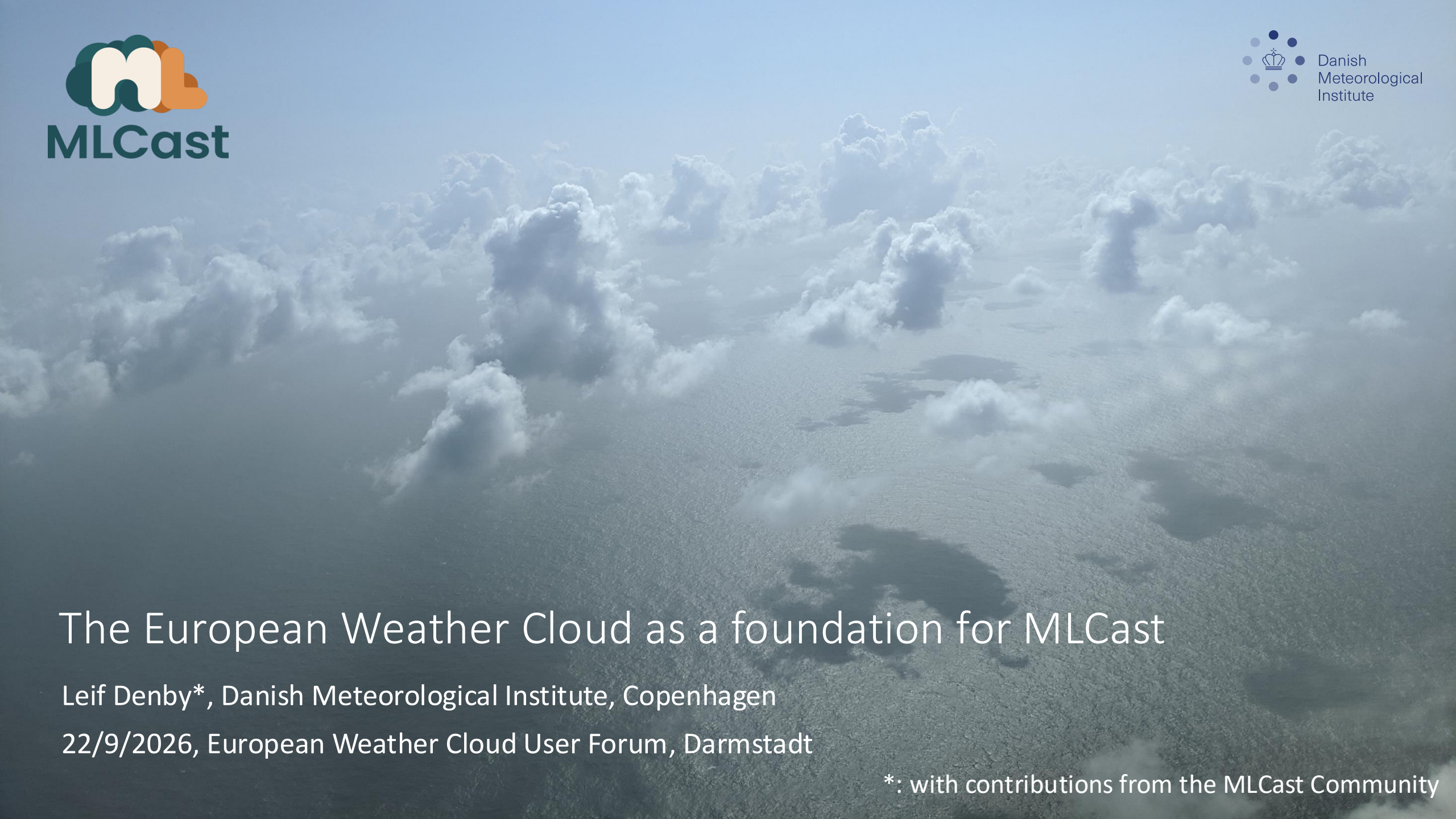


The European Weather Cloud as a foundation for MLCast

Leif Denby*, Danish Meteorological Institute, Copenhagen

22/9/2026, European Weather Cloud User Forum, Darmstadt

*: with contributions from the MLCast Community

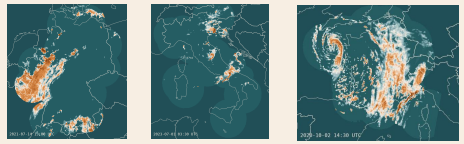
THE COMMUNITY

MLCast: deep learning based nowcasting community

MLCast is **Working Group 6 on Nowcasting** of EUMETNET's E-AI Programme: national met services, research institutes and universities building one open ecosystem; each partner contributes its own observations, models, and expertise.

Our goal: jointly developments towards nowcasting capacity

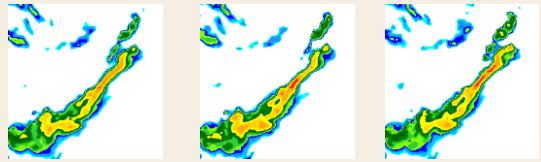
Observations in: radar today, satellite & solar next



mlcast · nowcasting-in-a-box

one clean API: pretrained models · training · benchmarking

Skilful 0 - 6 h nowcasts out: deployable anywhere



Research demos exist; a robust operational package doesn't. That is the gap; target: an operational tool within 2 years.

Why EWC is our foundation

Free and shared: an S3-compatible object store, no cost to contributors

Open access: public buckets, anonymous read, no accounts needed

One neutral home: many national services, one place, one standard

Close to compute: the same platform can host CI, GPUs and deployments

80+ people

29 organisations

14 countries

+ 10,000 GPU hours on Leonardo secured for community training · growing beyond Europe (ECCC, INPE, WMO AINPP)

Adapted from G. Franch et al., MLESM 2026

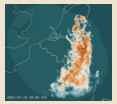
THE BIG PICTURE

From national archives to trained models

DATA PIPELINE

National source files

NetCDF · GRIB · GeoTIFF



Conversion tooling

open, on mlcast-community GitHub

`pip install mlcast-dataset-validator`

Dataset validator

format & CF checks; every dataset passes it

CF Zarr

Stored in European Weather Cloud (EWC) S3

`pip install mlcast-datasets`

Intake catalog

one entry point to all archives

read via the catalog: one xarray call away

MODEL PIPELINE

`pip install mlcast`

mlcast

sampling · training · baselines · losses

Trained models

checkpoints + pretrained weights

Model repository

Hugging Face

Benchmarking

mlcast-benchmarks

Deployment

containerised inference engine with deployment blueprint

End user

Jupyter · your own science

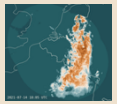
This talk: what the European Weather Cloud makes possible in the data pipeline, and where we would like to take it next

Diagram adapted from the MLCast community deck (G. Franch, I. Kruse et al.)

Where the European Weather Cloud fits today

DATA PIPELINE

National source files
NetCDF · GRIB · GeoTIFF



Conversion tooling
open, on mlcast-community GitHub

`pip install mlcast-dataset-validator`

Dataset validator
format & CF checks; every dataset passes it

CF Zarr
Stored in European Weather Cloud (EWC) S3

`pip install mlcast-datasets`

Intake catalog
one entry point to all archives

ON EWC TODAY: storage + catalog

read via the catalog: one xarray call away

MODEL PIPELINE

`pip install mlcast`

mlcast
sampling · training · baselines · losses

Trained models
checkpoints + pretrained weights

Model repository
Hugging Face

Benchmarking
mlcast-benchmarks

Deployment
containerised inference engine with deployment blueprint

End user
Jupyter · your own science

On EWC today

S3 object storage for the datasets, and the intake catalog that points to it

Diagram adapted from the MLCast community deck (G. Franch, I. Kruse et al.)

What is the European Weather Cloud?

A cloud platform for meteorological application development and operations, run by ECMWF and EUMETSAT, where software runs close to the data

Data proximity

Access to ECMWF and EUMETSAT data repositories, next to the compute

Compute

Virtual machines and application environments at both sites

We've been experiment with this

Object storage

S3-compatible; buckets can be public with anonymous read

MLCast uses this today

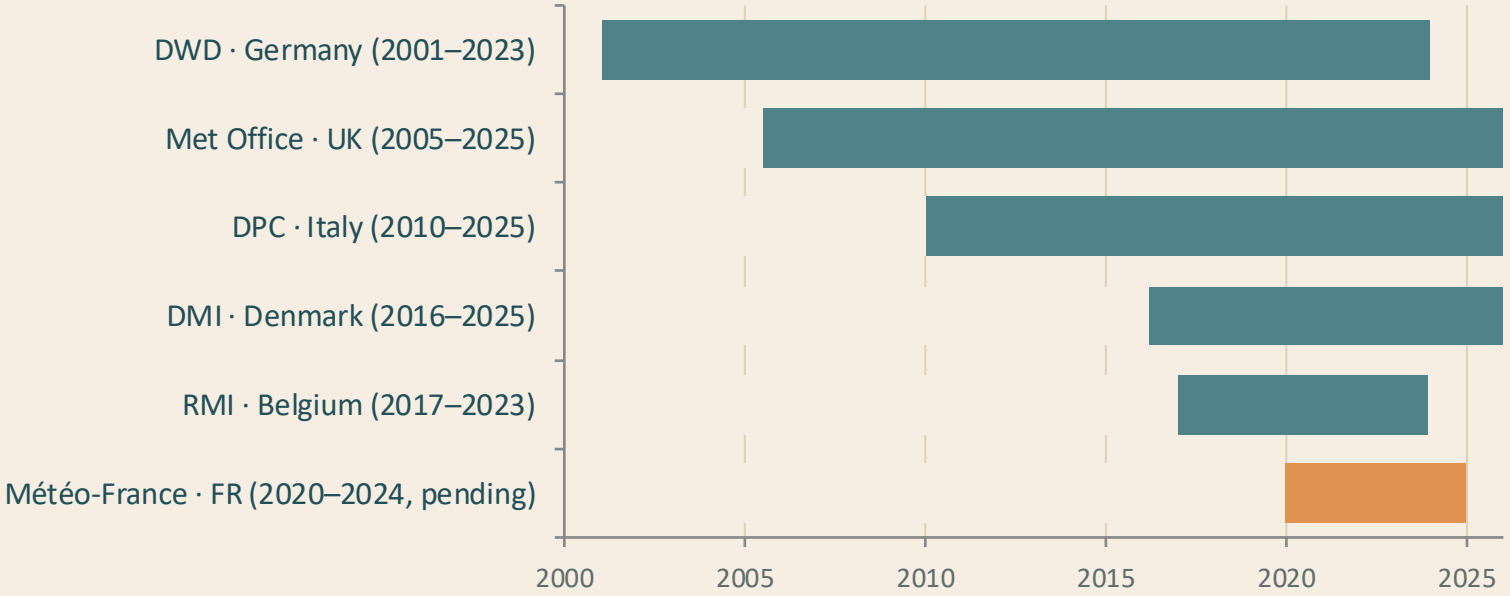
Deployment

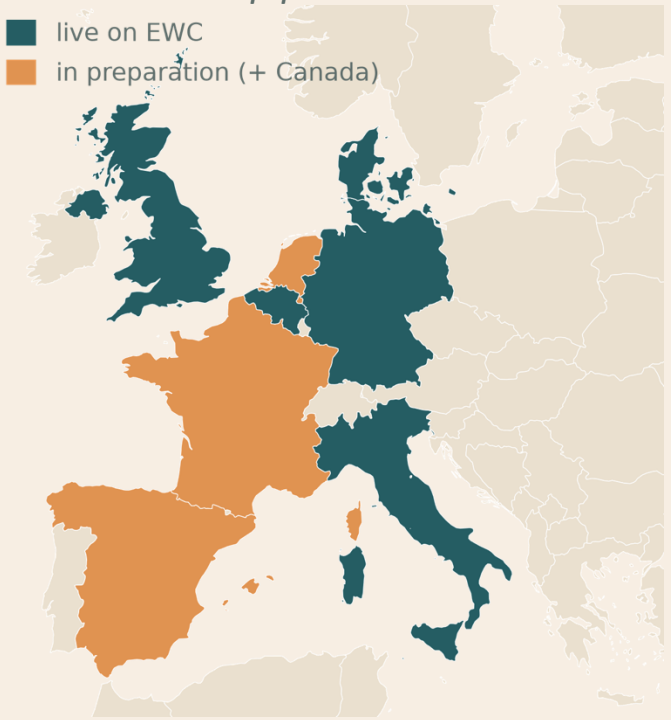
Customise and expose your own applications and services

Not a code forge: source code and CI triggers stay on GitHub, EWC is where things run

76 years of radar, live on EWC

Five national networks live today · France (approval pending), Spain, Canada, the Netherlands & OPERA in the pipeline





1 km or finer

grid spacing (down to 500 m)

5 min cadence

10–15 min in older archive years

CF-compliant Zarr

analysis-ready, cloud-optimised

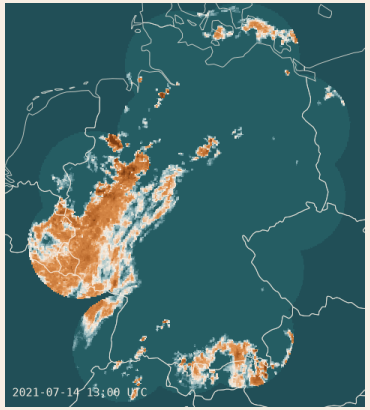
European Weather Cloud

one place, open access

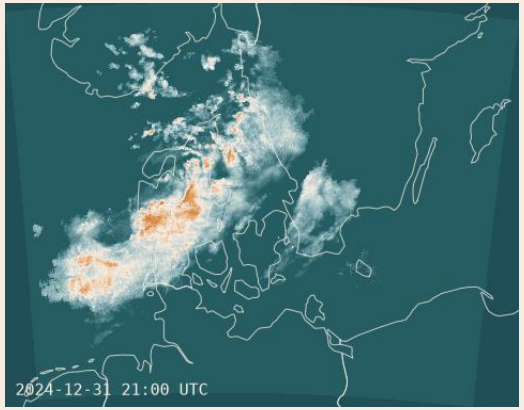
Adapted from G. Franch et al., MLESM 2026

One format, six national networks, in motion

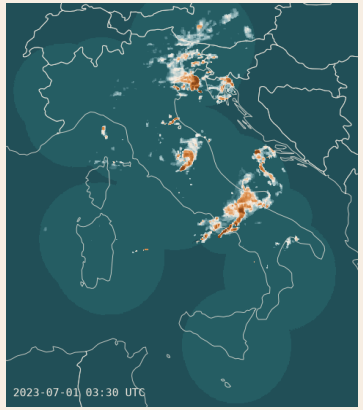
DWD · Germany



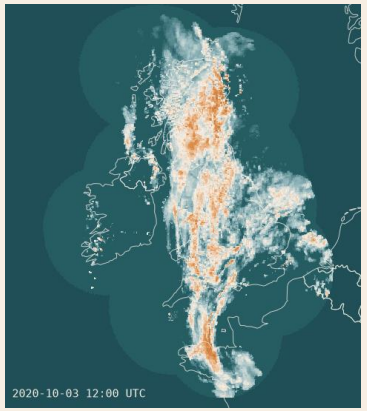
DMI · Denmark



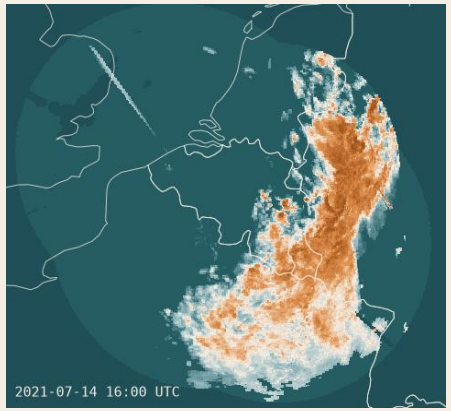
DPC · Italy



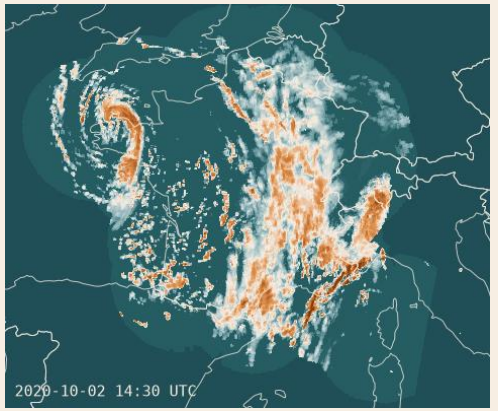
Met Office · UK



RMI · Belgium



Météo-France · France*



*Same product, same format, same xarray call, all from EWC. DE & BE: July 2021 floods; UK & FR: Storm Alex, Oct 2020. *France: open pull-request*

Adapted from G. Franch et al., MLESM 2026

From raw radar to catalog entry, stage by stage

on EWC today

Raw radar files → **Convert to Zarr** → **Validate** → **Publish on EWC** → **Intake catalog**

national formats, e.g. GeoTIFF, NIMROD, npz

CF-compliant, one conversion repo per dataset

mlcast-dataset-validator

public S3 bucket, anonymous read

versioned YAML on GitHub, pip-installable

Contributors publish their conversion code in a public repository, so every dataset is reproducible

s3 access tokens provided by me (lcd@dmi.dk)

```
leifdenby@thixo: ~
† ~ rclone sync \
 "2001_2023.zarr/" \
 "ewc:mlcast-source-datasets/radklim/v0.1.1/5_minutes.zarr/" \
 -P \
 --stats=2s \
 --stats-one-line \
 --fast-list \
 --transfers=32 \
 --checkers=64 \
 --buffer-size=16M \
 --s3-chunk-size=64M \
 --s3-upload-concurrency=8 \
 --multi-thread-streams=4
```

for example https://github.com/mlcast-community/mlcast-dataset-DE-DWD-radklim

The spec is code, and CI runs it on EWC data

1 The spec is code

Requirements are written as inline text, right beside the check that enforces them

```
spec_text += """
### 3.2 Spatial Requirements
- The dataset MUST provide 2D gridded
  radar data with a spatial resolution
  of 1 kilometer (with 1% tolerance)
  or finer.
"""
report += check_spatial_requirements(
    ds, max_resolution_km=1.0, ...)
```

which renders as the spec document

> 3.2 Spatial Requirements
>
> - The dataset MUST provide 2D gridded radar data with a spatial resolution of 1 kilometer (with 1% tolerance) or finer.

mlcast-community.github.io/mlcast-dataset-validator

2 Two ways to run it

The same spec either way; the Python call also returns the spec text

CLI

```
uvx --from mlcast-dataset-validator \
  mlcast.validate_dataset \
  source_data radar_precipitation \
  s3://mlcast-source-datasets/… \
  --s3-endpoint-url <EWC S3 endpoint>
```

Python API

```
ds = xr.open_zarr("s3://…/5_minutes.zarr/",
                  storage_options=ewc)
report, spec_text = (
    radar_precipitation.validate_dataset(ds)
)
report.console_print()
```

3 The report

One line per requirement: pass, warning or fail

Section	Requirement	Status
3.3	Spatial resolution ≤1km	✅
3.3	256×256 pixel support	✅
3.4	Minimum 3-year coverage	✅
3.5	Timestep consistency	✅
4.1	Chunking strategy for RR	✅
4.2	DataArray compression RR	⚠
4.3	Dimension order for RR	✅
4.3	Data type for RR	✅
4.4	Variable name validation	❌
4.4	Units validation	✅
4.5	CRS attributes for RR	✅
5.2	License metadata	❌

Summary: 2 fail(s), 1 warning(s), 19 pass(es).

Early CLI run on the original RADKLIM zarr: the variable name (RR) and a missing licence field fail. RADKLIM was re-released as v0.1.1 to pass.

4 In CI

Pull request or weekly schedule → pytest on GitHub Actions → validate_dataset() on every catalog dataset → **Zarr read from EWC object storage** → Pass or fail on the PR

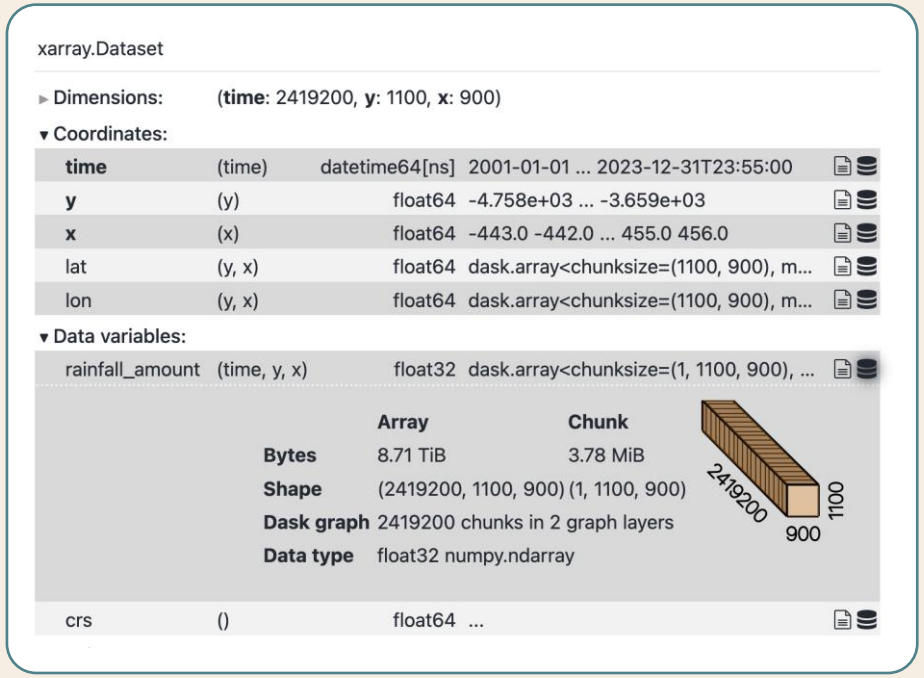
code on https://github.com/mlcast-community/mlcast-dataset-validator

THE PAYOFF

From catalog to lazy xarray in three lines

```
$ pip install mlcast-datasets
>>> cat = mlcast_datasets.open_catalog()
>>> ds  = cat.precipitation.radklim_5_minutes.to_dask()
# any of the national archives, same call
```

Catalog → **Notebook or training job**



what the end user sees: 8.7 TiB of DWD radar, one lazy xarray variable

Lazy, dask-backed xarray

Any time and space slice

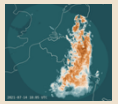
No account, no download

code and notebooks on https://github.com/mlcast-community/mlcast-datasets

An even stronger foundation on EWC: four ideas

DATA PIPELINE

National source files
NetCDF · GRIB · GeoTIFF



Conversion tooling
open, on mlcast-community GitHub

`pip install mlcast-dataset-validator`

Dataset validator
format & CF checks; every dataset passes it

1

CF Zarr
Stored in European Weather Cloud (EWC) S3

3

`pip install mlcast-datasets`

Intake catalog
one entry point to all archives

read via the catalog: one xarray call away

MODEL PIPELINE

`pip install mlcast`

mlcast
sampling · training · baselines · losses

2

Trained models
checkpoints + pretrained weights

Model repository
Hugging Face

Benchmarking
mlcast-benchmarks

Deployment
containerised inference engine with deployment blueprint

4

End user
Jupyter · your own science

1 CI next to the data

run the validator's full checks on the whole archive, not slices over the network

2 GPU smoke tests

one epoch on a tiny slice after every code change, without a separate eligibility process

3 Self-service S3 tokens

bucket-scoped tokens and policies without going through support

4 A reference deployment

a blueprint repo for a nowcasting model, checked out and actually run on EWC

Code and CI triggers stay on GitHub; EWC would be where the jobs run

So, what's next? :)

1	**EWC platform teams**	Happy to discuss self-service S3 tokens and CI (GPU) compute next to the data
2	**Met services with radar data**	Open an issue or a pull request on mlcast-datasets
3	**ML researchers**	Try it: pip install mlcast-datasets, three lines to a multi-terabyte dataset, and there's `pip install mlcast` too!
4	**Building ML infrastructure elsewhere**	Reuse the blueprint: object store + catalog + validator

mlcast.org **github.com/mlcast-community**

Leif Denby
lcd@dmi.dk

MLCast