DISCOVER ANEMOI · WEBINAR SERIES · PART 1

Introduction + Anemoi Datasets

Building machine-learning–ready datasets for data-driven weather forecasting.

PRESENTER
Ana Prieto Nemesio
Machine Learning Team Lead
ana.prietonemesio@ecmwf.int

PRESENTER
Harrison Cook
Research Software Engineer
harrison.cook@ecmwf.int

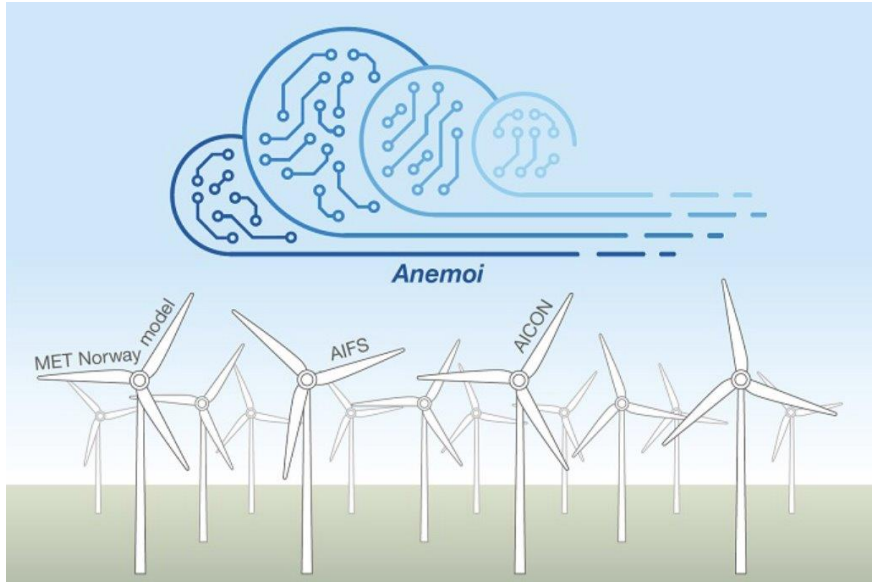
Introduction Overview

- What's Anemoi?
- Overview of the framework
  - Main packages
  - Earthkit
- Examples of the use cases
- What's next?
- Keep learning about Anemoi

What's Anemoi?

Anemoi: a European framework for operational AI in weather and climate

- Anemoi is an open-source framework that provides a complete toolkit to develop data-driven weather models – from data preparation through to inference
- It's a highly modular framework organised into different Python packages
- The framework builds upon on established Python tooling including PyTorch, Pytorch Lighting, Hydra, Pydantic, and earthkit.



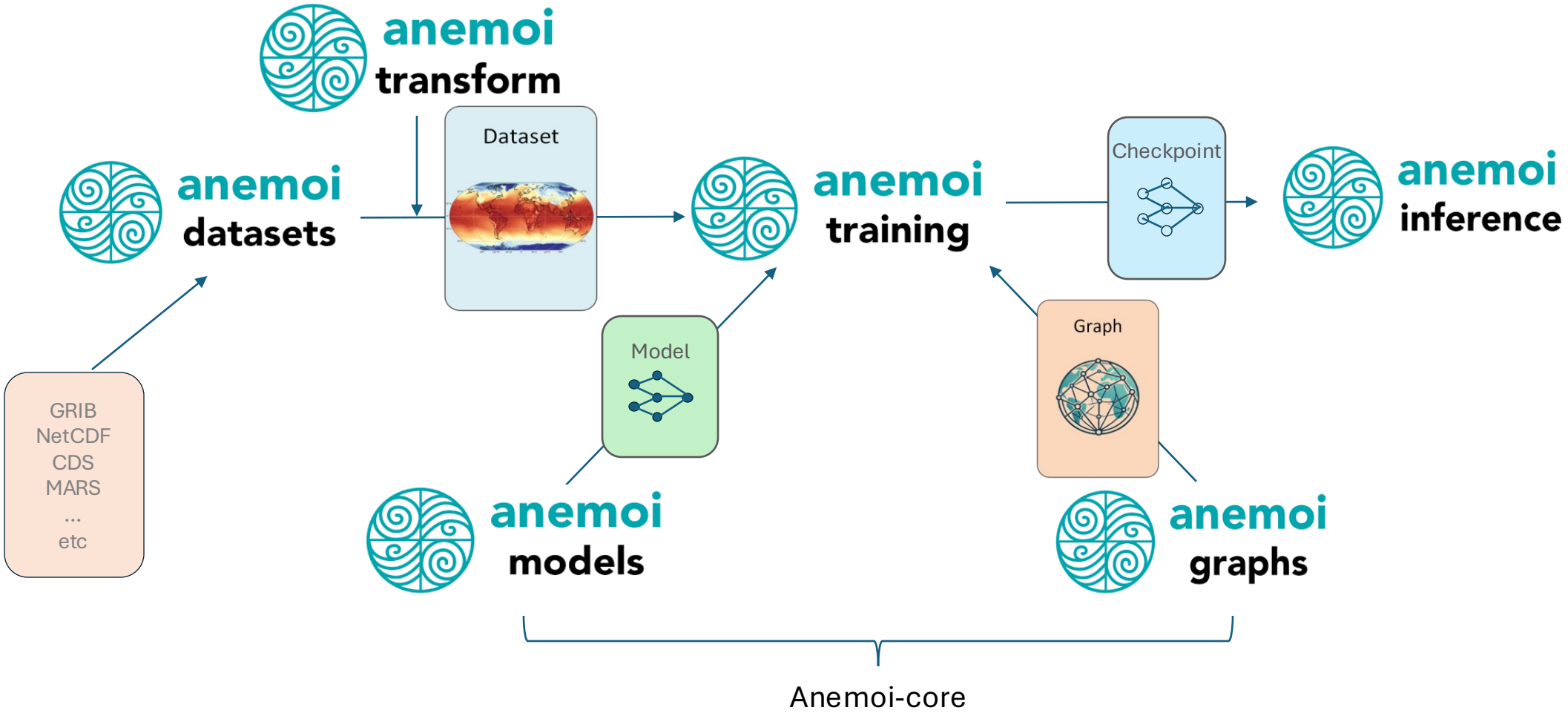
Goals

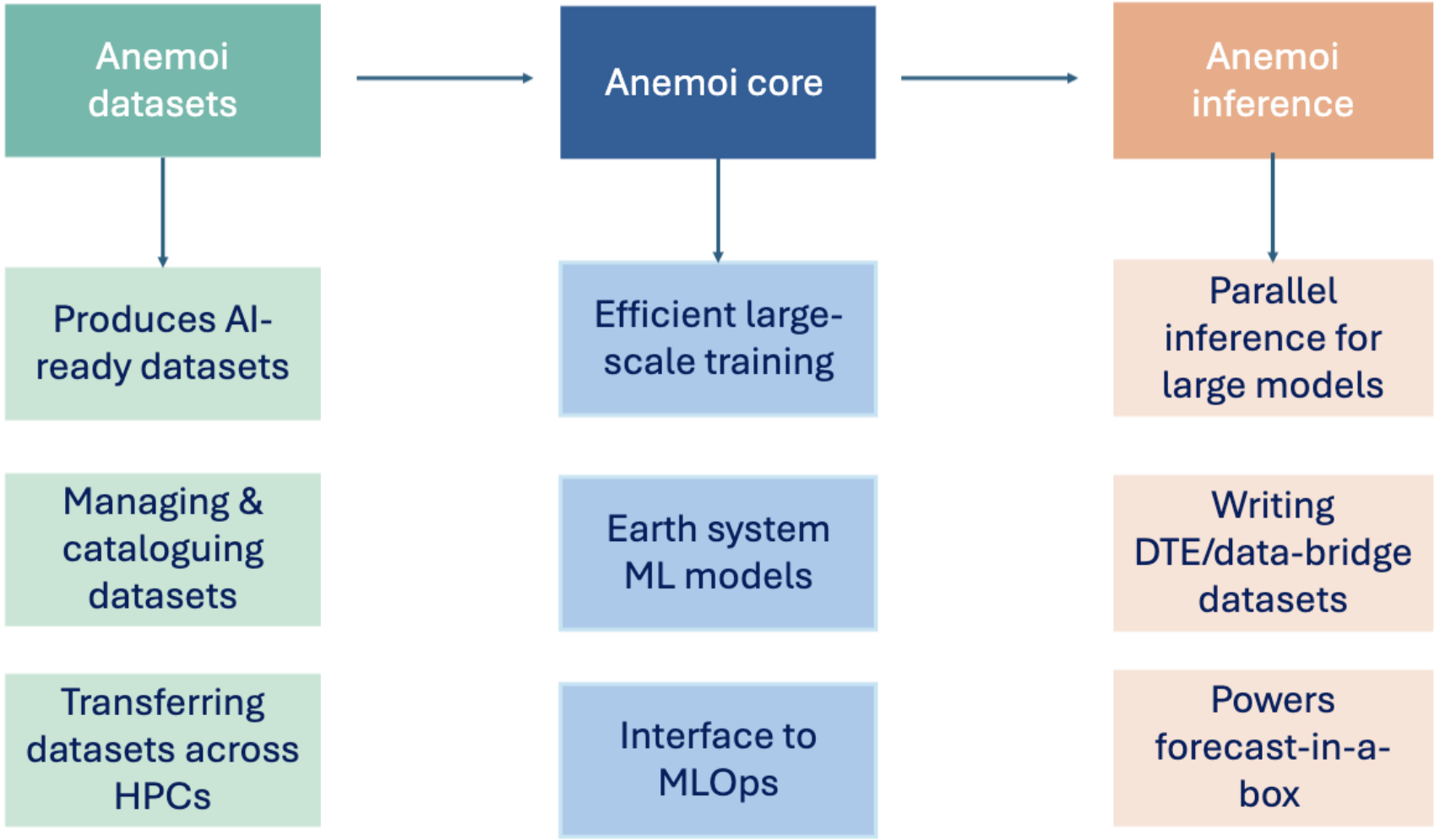
Anemoi: a European framework for operational AI in weather and climate

- Set of tools, shared and co-developed across Europe for building data driven forecasting systems
- Users can bring their data and pick a suitable architecture and training method
- Advanced users can design new architectures or change training strategies
- Enable research, but designed with operations in mind

https://www.ecmwf.int/en/about/media-centre/aifs-blog/2026/anemoi-european-framework-ai

Anemoi in a Nutshell




Anemoi datasets
Anemoi core
Anemoi inference
Produces AI-ready datasets
Efficient large-scale training
Parallel inference for large models
Managing & cataloguing datasets
Earth system ML models
Writing DTE/data-bridge datasets
Transferring datasets across HPCs
Interface to MLOps
Powers forecast-in-a-box

User Base

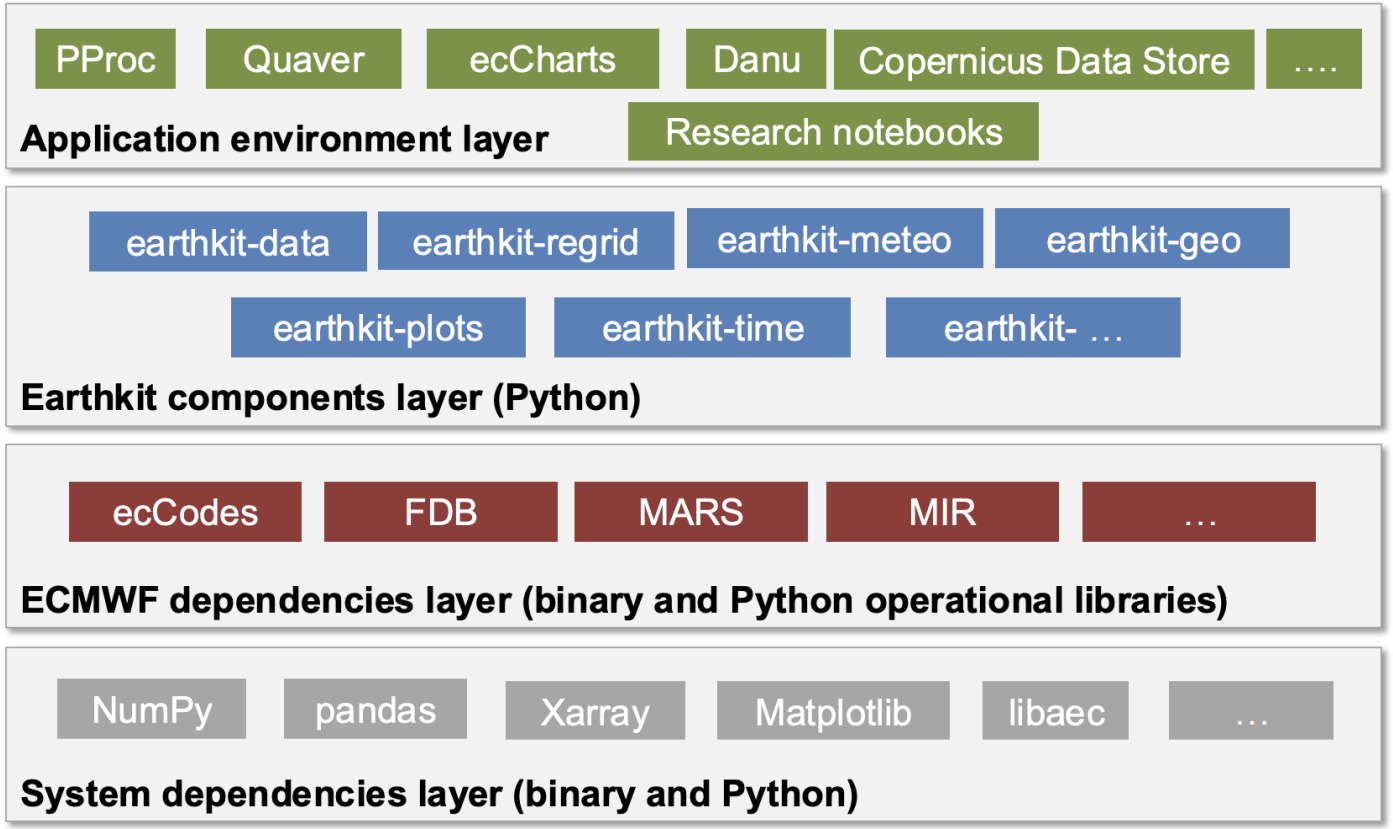
User Base

Modifies Configs for Experimentation and Implemenation of Operational Models

Modifies Codebase to implement new Features and Augment Anemoi Libraries

Runs the Anemoi Model in a common interface on reliable infrastructure

What's Earthkit aiming at from an ECMWF software stack perspective



Anemoi uses Earthkit components across the framework

anemoi

earthkit

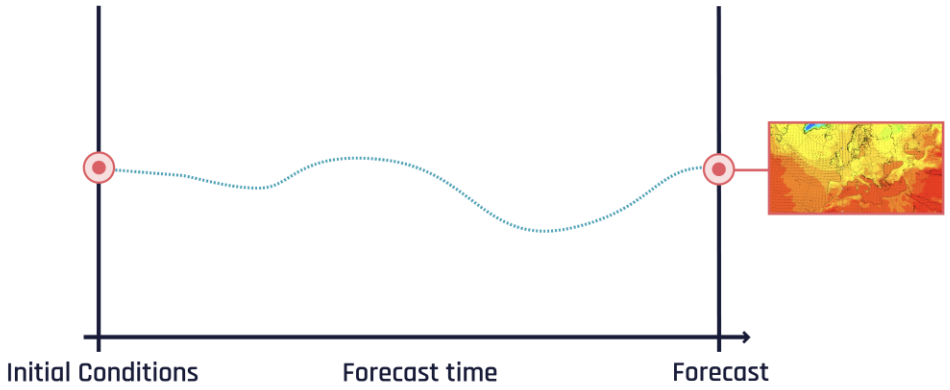
https://ecmwf.github.io/earthkit-website/

earthkit data
earthkit geo
earthkit hydro
earthkit workflows
earthkit meteo
earthkit utils
earthkit plots
earthkit regrid
earthkit time
earthkit transforms

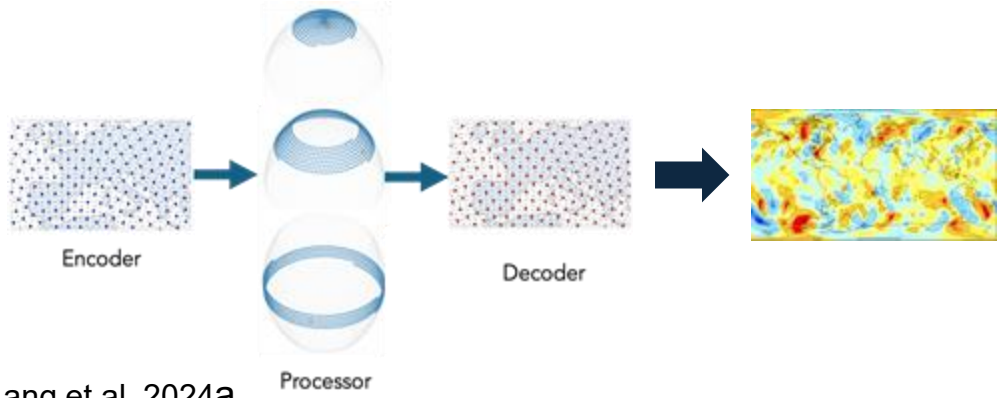
What does Anemoi enable?

AI forecasts ready in minutes rather than hours

Data-driven Deterministic Forecast

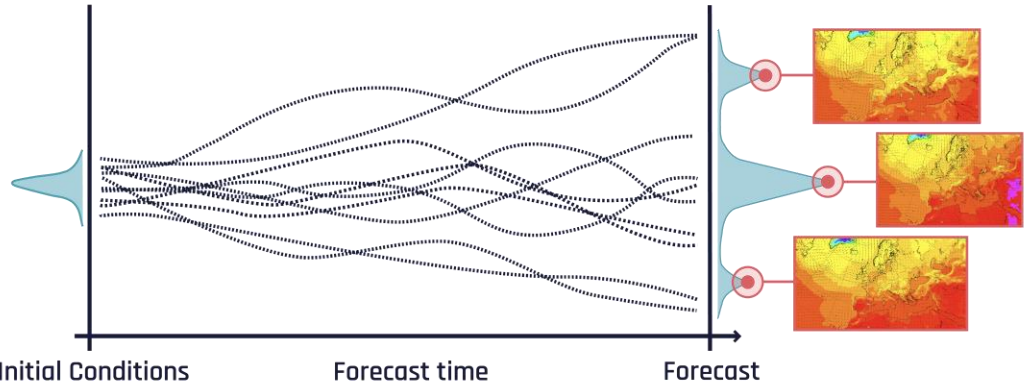


Deterministic Model

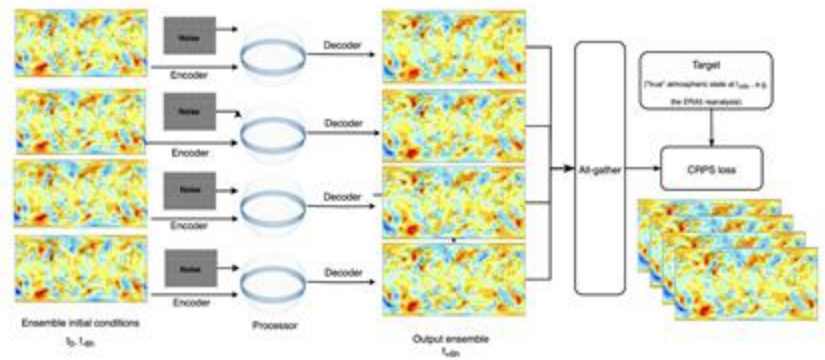


Lang et al. 2024a

Data-driven Ensemble Forecast



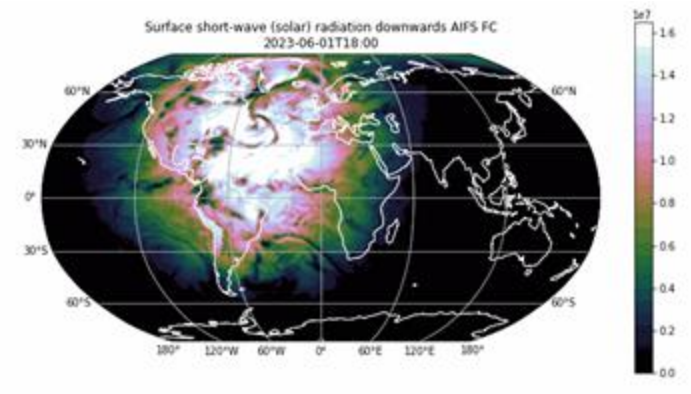
Probabilistic Model - Learn an ensemble via optimization of a probabilistic score



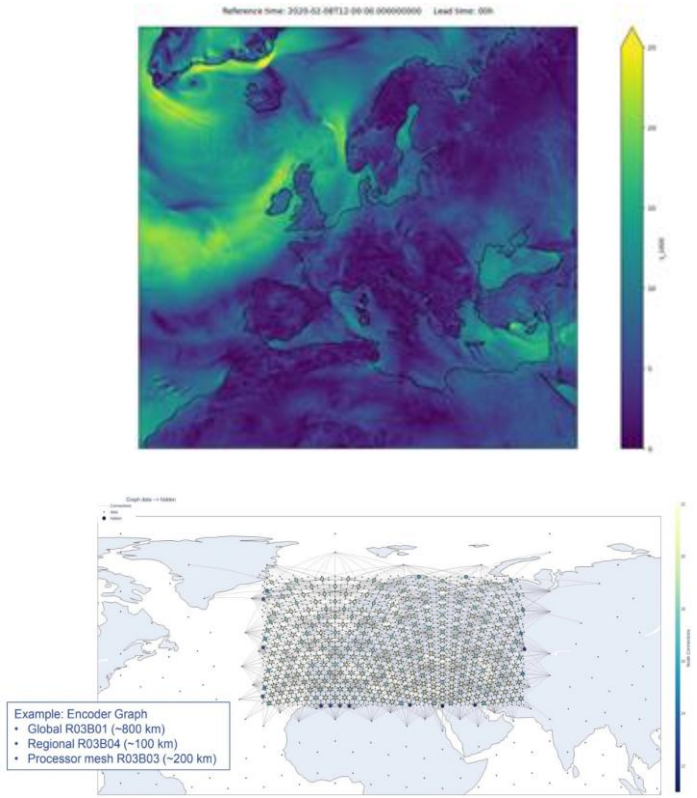
Lang et al. 2024b

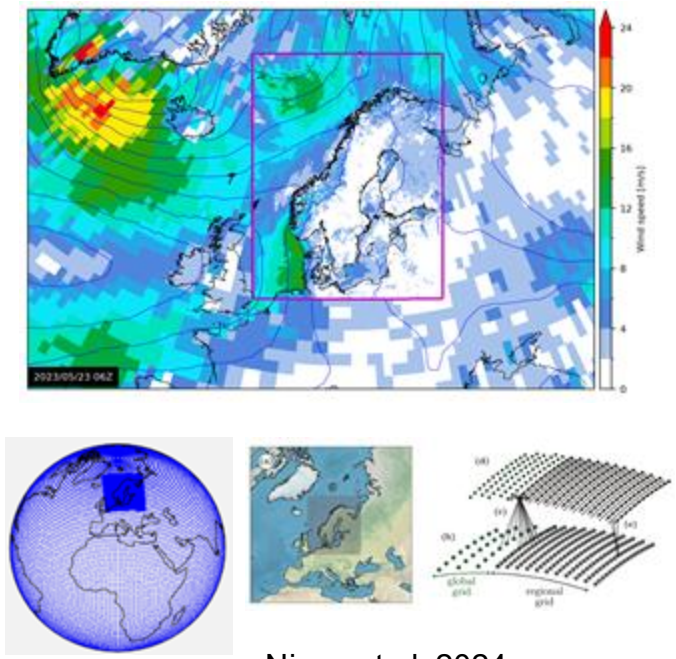
What does Anemoi enable?

Global



Regional Area Modelling



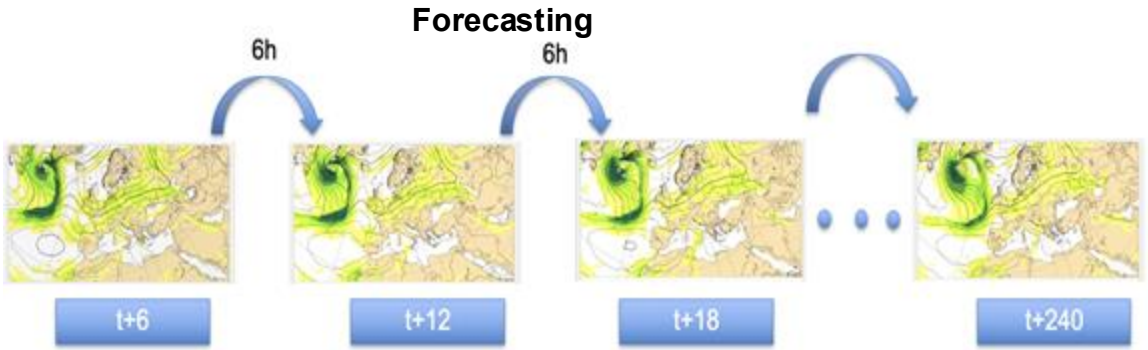


Nipen et al. 2024

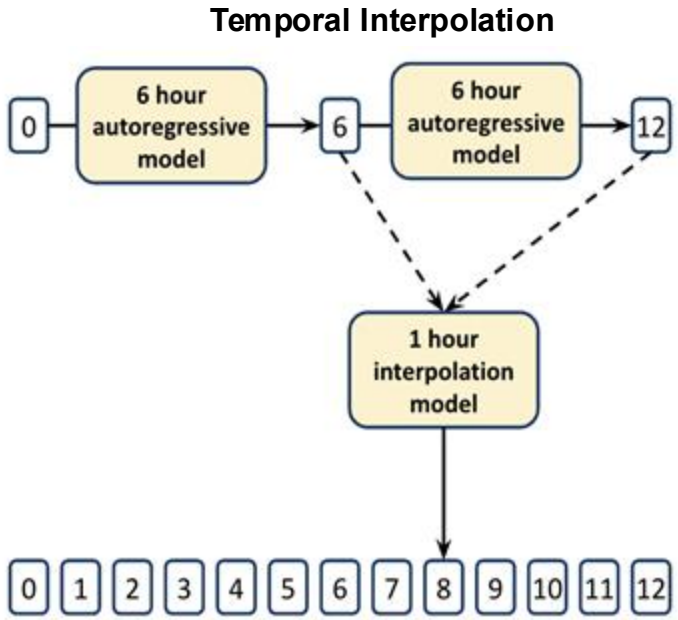
Stretched Grid Model

What does Anemoi enable?

- Forecasting and Time Interpolation

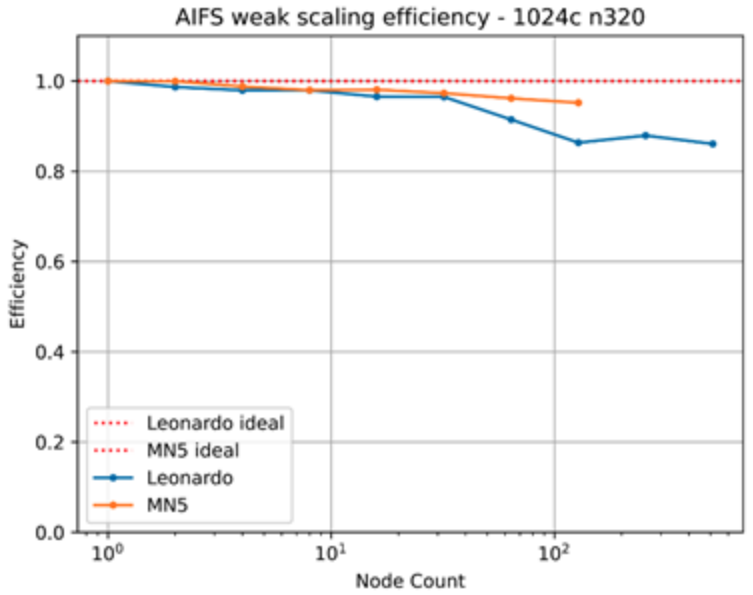


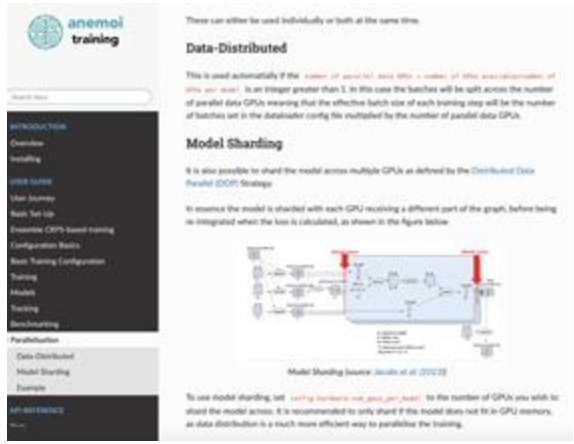
Autoregressive forecasts stepping 6h into the future xn = f(xn-1) …



How does Anemoi achieve this?

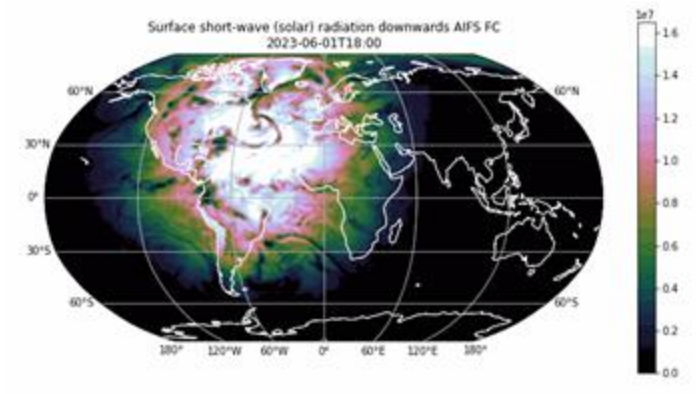
- Generation of **ML-optimised datasets** from common meteorological sources (NetCDF, HDF5, GRIB, MARS, CDS)
- Cutting-edge **Model architecture blocks**
- Support to **track training results using MLflow**
- Ability to **train large scale models with both data and model parallelization**
- **Testing Suite – Unit tests and Integration Tests**
- **Highly configurable framework** both for training and inference
- End-to-end capturing of **metadata lineage**
- **Open research and open development**
- **Collaboration** – investigating and bringing new features (MLPP and E-AI programmes)
- End-end workflow – allow to develop **DDWF suitable for NWP operationalisation**



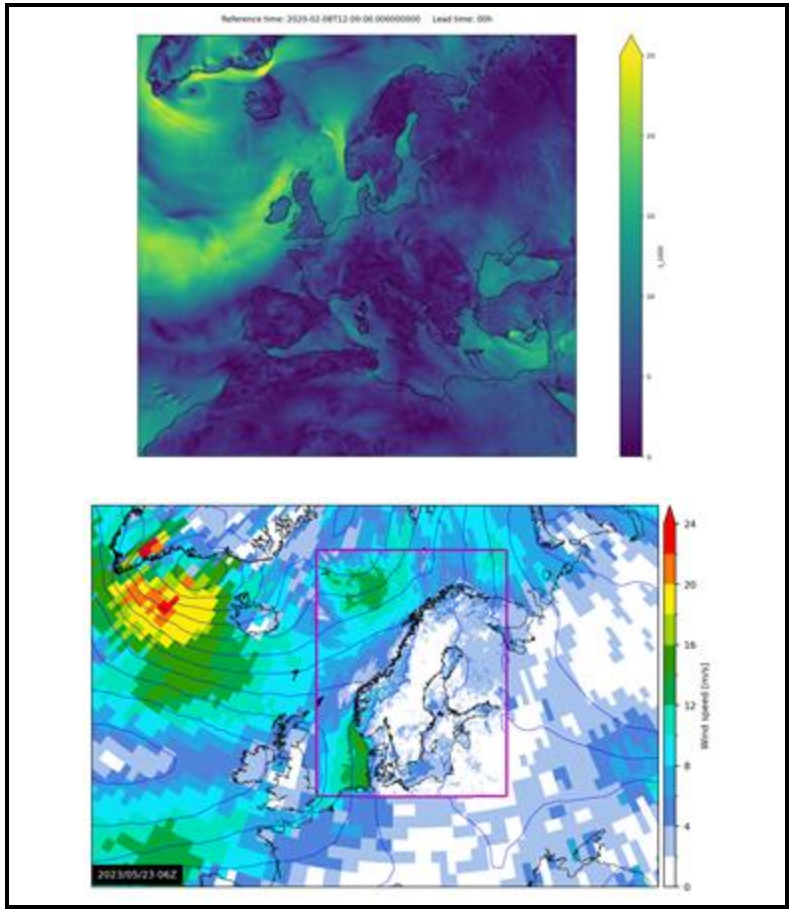


What's next?

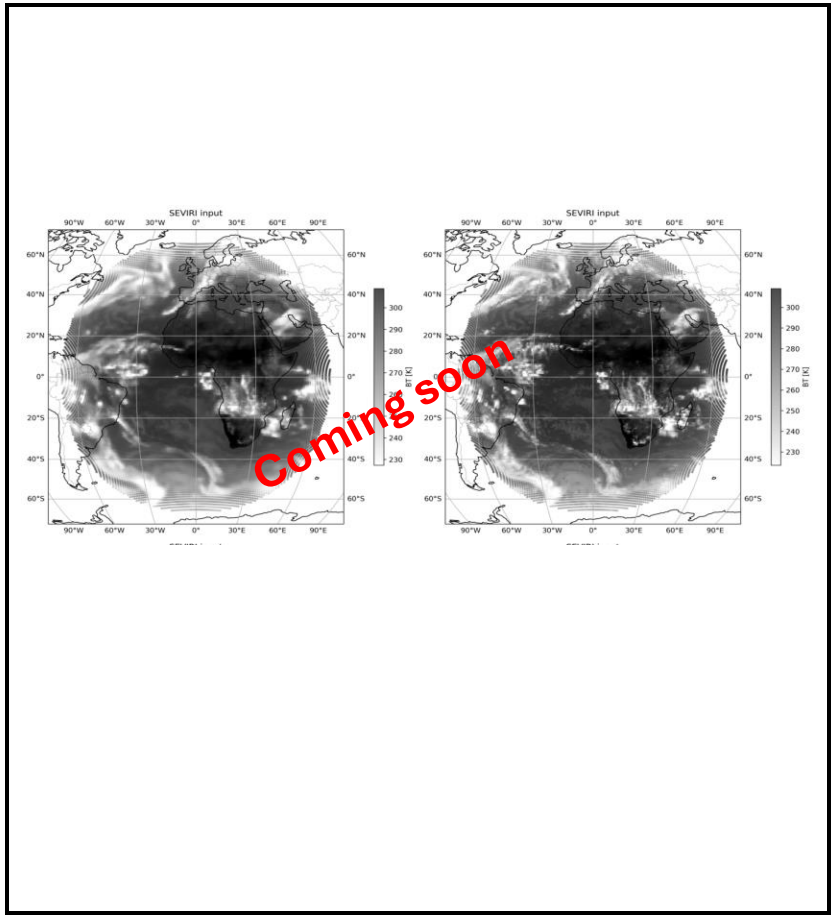
Global



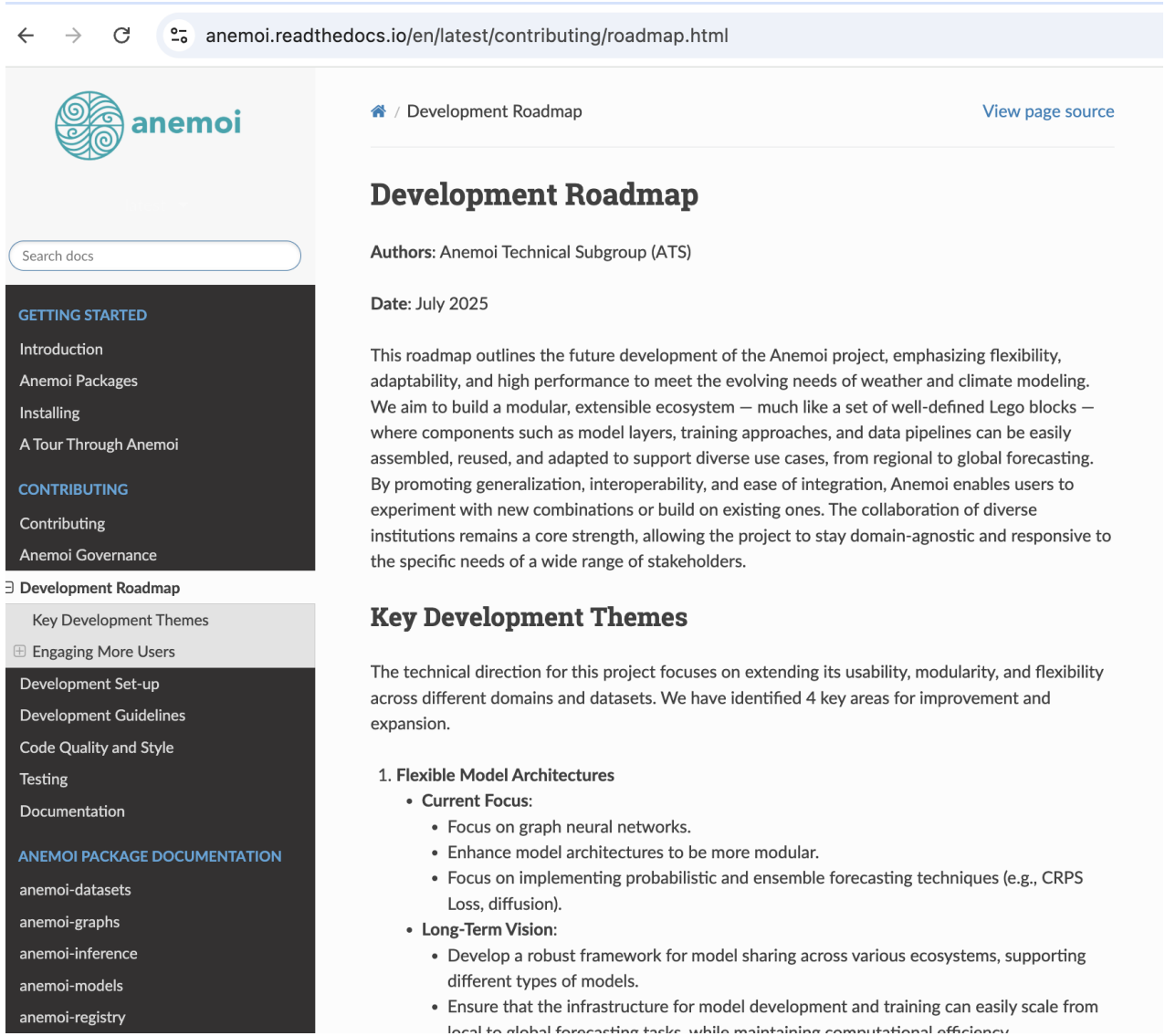
Limited area



Direct observation prediction

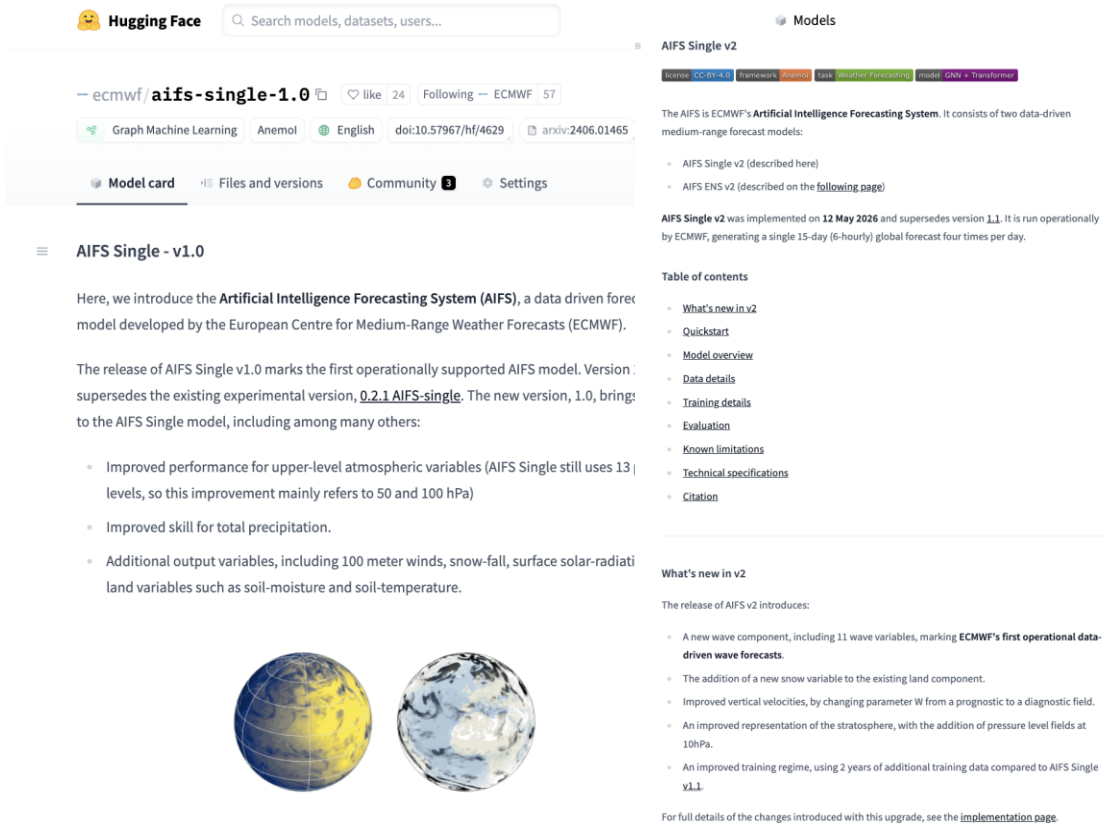


What's next?

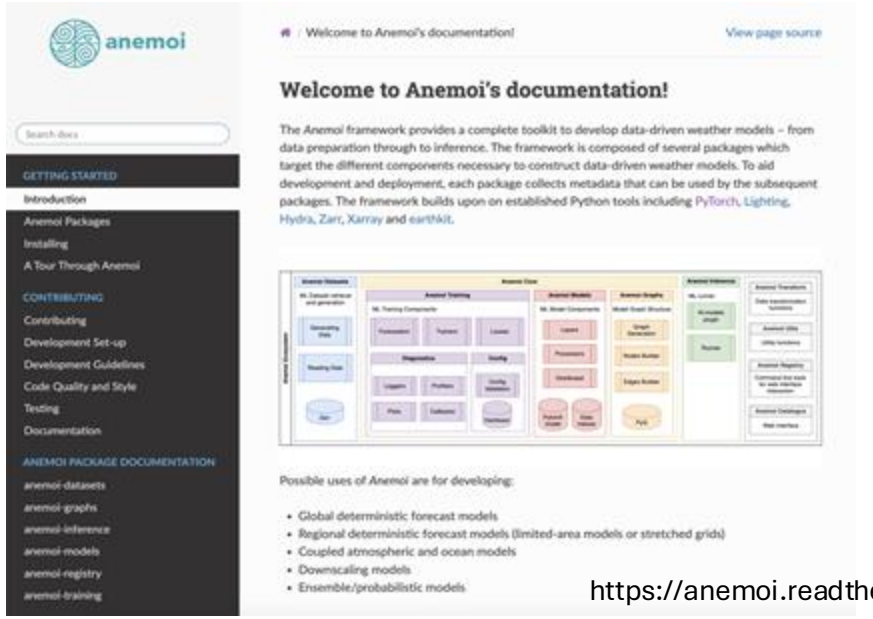

anemoi.readthedocs.io/en/latest/contributing/roadmap.html
anemoi
Search docs
GETTING STARTED
Introduction
Anemoi Packages
Installing
A Tour Through Anemoi
CONTRIBUTING
Contributing
Anemoi Governance
Development Roadmap
Key Development Themes
Engaging More Users
Development Set-up
Development Guidelines
Code Quality and Style
Testing
Documentation
ANEMOI PACKAGE DOCUMENTATION
anemoi-datasets
anemoi-graphs
anemoi-inference
anemoi-models
anemoi-registry
Development Roadmap
View page source
Development Roadmap
Authors: Anemoi Technical Subgroup (ATS)
Date: July 2025
This roadmap outlines the future development of the Anemoi project, emphasizing flexibility, adaptability, and high performance to meet the evolving needs of weather and climate modeling. We aim to build a modular, extensible ecosystem — much like a set of well-defined Lego blocks — where components such as model layers, training approaches, and data pipelines can be easily assembled, reused, and adapted to support diverse use cases, from regional to global forecasting. By promoting generalization, interoperability, and ease of integration, Anemoi enables users to experiment with new combinations or build on existing ones. The collaboration of diverse institutions remains a core strength, allowing the project to stay domain-agnostic and responsive to the specific needs of a wide range of stakeholders.
Key Development Themes
The technical direction for this project focuses on extending its usability, modularity, and flexibility across different domains and datasets. We have identified 4 key areas for improvement and expansion.
1. Flexible Model Architectures
• Current Focus:
• Focus on graph neural networks.
• Enhance model architectures to be more modular.
• Focus on implementing probabilistic and ensemble forecasting techniques (e.g., CRPS Loss, diffusion).
• Long-Term Vision:
• Develop a robust framework for model sharing across various ecosystems, supporting different types of models.
• Ensure that the infrastructure for model development and training can easily scale from

Keep learning about Anemoi

- Webinars and documentation
- HuggingFace – run AIFS v2 yourself!



https://huggingface.co/ecmwf/aifs-single-2.0



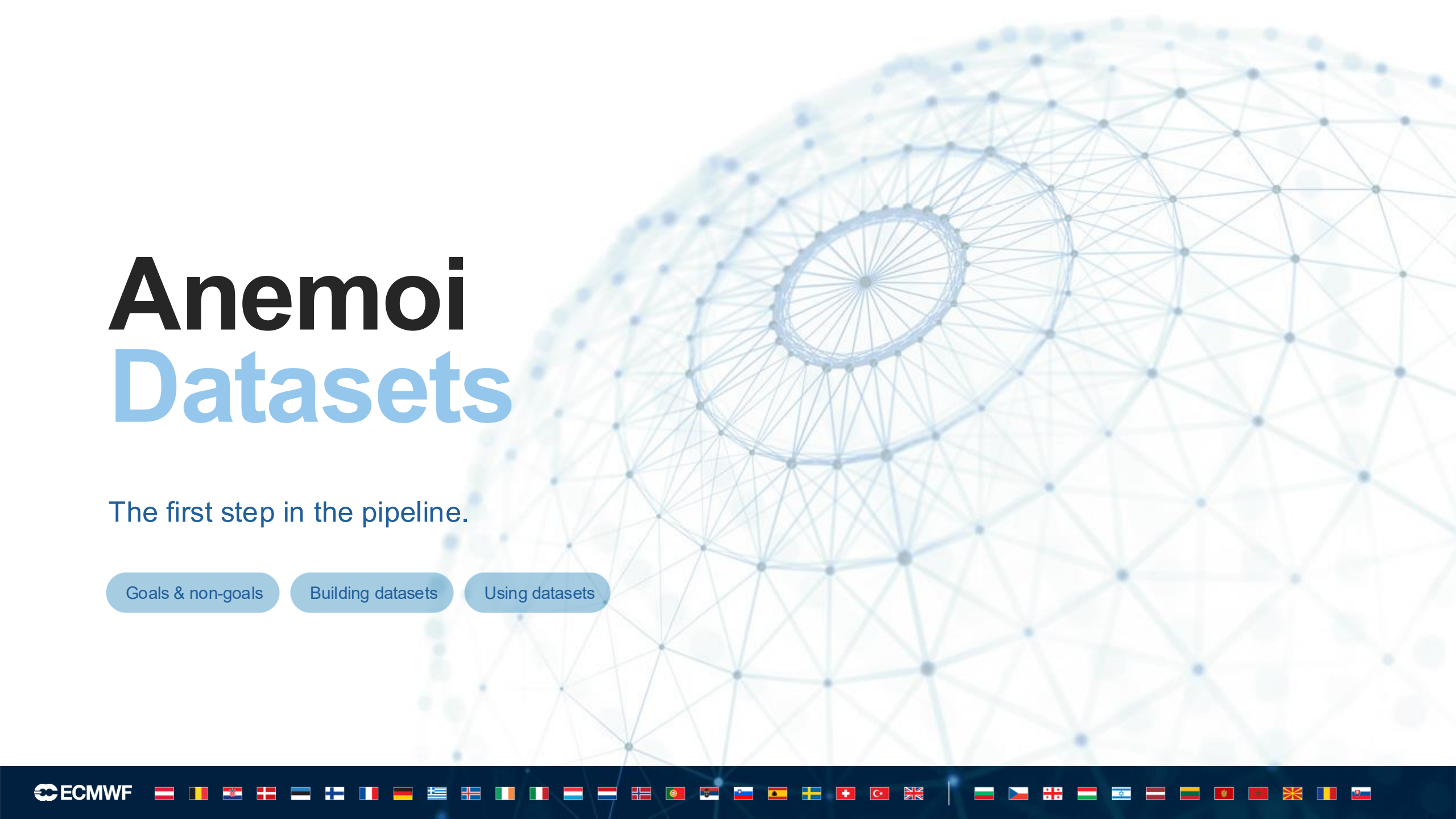
https://anemoi.readthedocs.io/en/latest/

https://events.ecmwf.int/event/446/

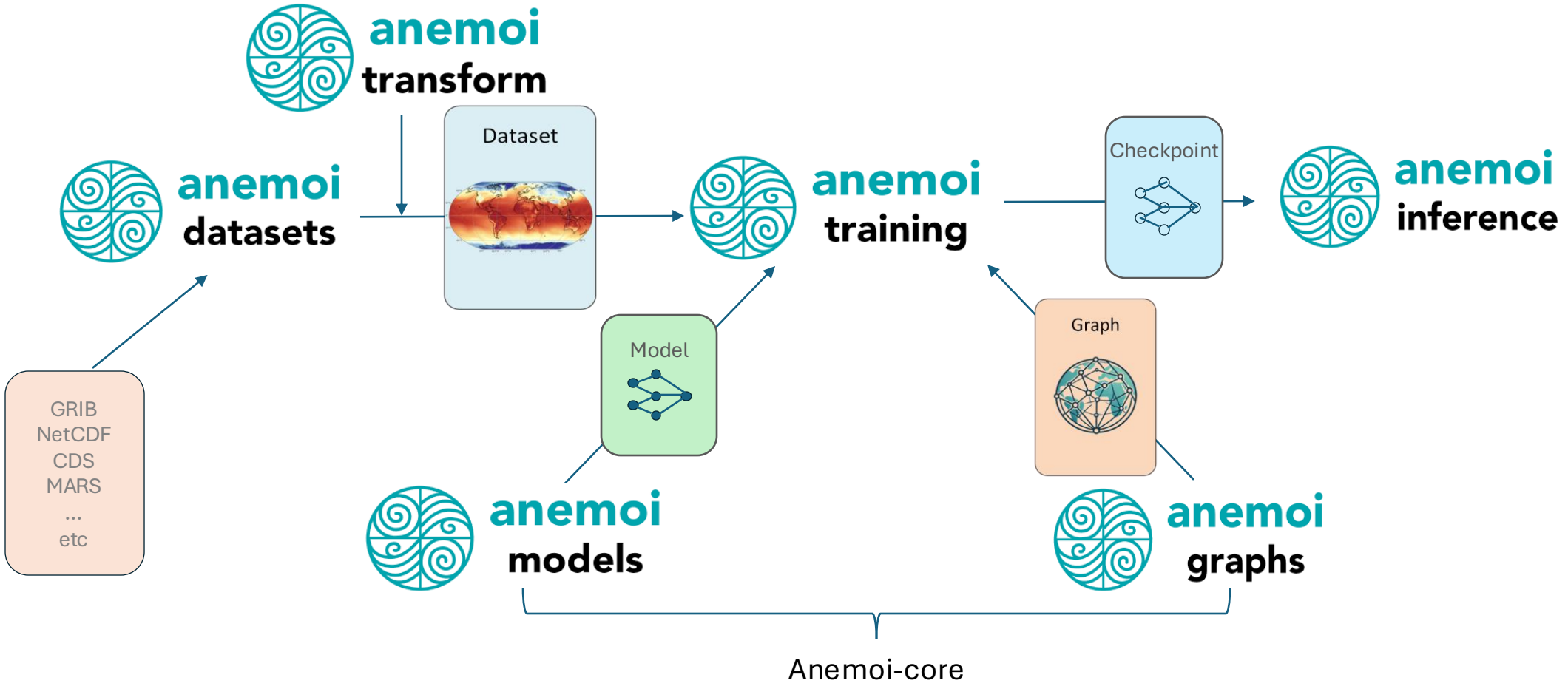
Anemoi
Datasets

The first step in the pipeline.

Goals & non-goals | Building datasets | Using datasets



Anemoi Pipeline



Goals

01

Machine-learning–ready

Create datasets ready to train data-driven weather forecasts

no last-mile pre-processing required.

02

Efficient loading

Make the loading of individual training samples as fast and lightweight as possible

keep the GPU fed.

03

Rich metadata

Carry everything training and inference need to interpret a sample

- variables, grid, dates, statistics.

Non-goals

IT IS NOT

A replacement for the original source of data

Keep the original GRIB/NetCDF archive. Anemoi datasets exist alongside it — they don't supersede it.

IT IS NOT

An analysis-ready collection

The read patterns for training differ from those for analysis.
Extracting a single point as a time series will be slow.

THINK OF IT AS

Pre-processed data, kept **as close as possible to the needs of training** — and nothing more.

Terminology

`sample`

The state of the atmosphere at a time `T`.

`model`

Given the state at `T`, returns the state at `T+1`.

`variables`

A sample is a collection of meteorological fields.

EXAMPLE VARIABLES

SURFACE

`2m_temperature` `surface_pressure` `10u, 10v` `tp`

PRESSURE LEVELS

geopotential, winds, temperature on

`1000 hPa` `500 hPa` `...`

Anemoi enforces no constraint on what a sample contains — that's a modelling choice.

The simplest abstraction

Everything that follows exists to make these three lines fast, reproducible and reusable.

```
x, y = ds[n : n + 1]      # pull one (input, target) pair
y_hat = model.predict(x)  # forward pass
loss = model.loss(y, y_hat)  # compare to truth
```

1 · Get the data → 2 · Make the prediction → 3 · Loss & back-prop

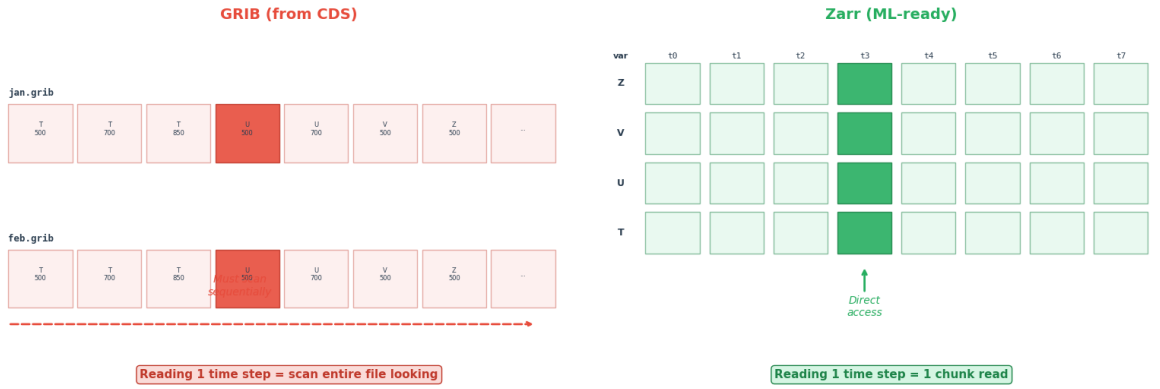
The problems

- Our best approximation of the **"true"** state of the atmosphere: **ERA5**.
- A dataset like ERA5 is **too large to fit in memory**.
- We need to store the data on a **filesystem** ready to **go**.
- We want each sample to be **one retrieval operation** — fast, predictable, parallelisable.

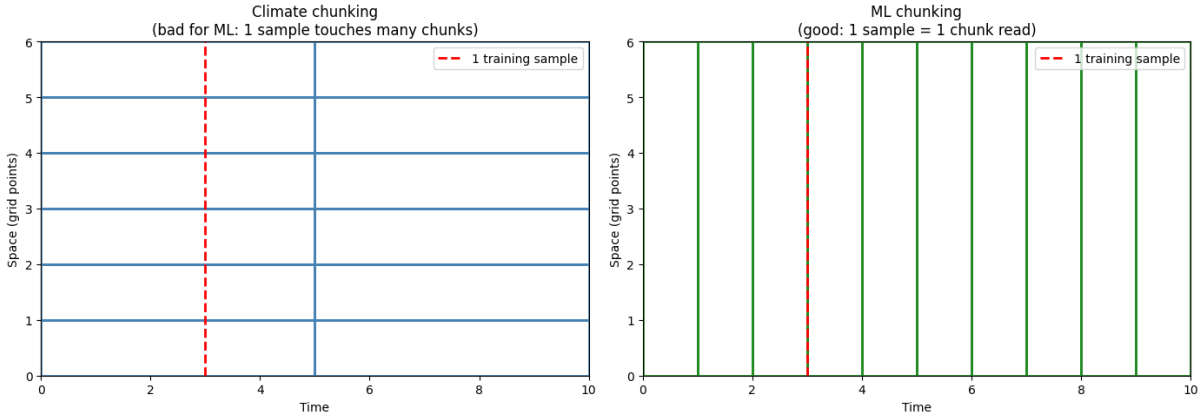
IN OTHER WORDS

The on-disk layout matters as much as the data itself.

STORAGE LAYOUT MATTERS



CHUNKING



Variable 1

$$\begin{bmatrix} 7.43 & 0.90 & 1.53 & 7.99 & 5.16 & 6.42 \\ 5.30 & 1.72 & 5.45 & 6.48 & 5.11 & 0.43 \\ 8.08 & 9.51 & 7.09 & 2.97 & 8.09 & 6.43 \end{bmatrix}$$

Variable 2

$$\begin{bmatrix} 8.20 & 2.55 & 8.02 & 4.34 & 6.46 & 5.71 \\ 5.97 & 1.98 & 4.61 & 6.66 & 6.25 & 6.45 \\ 7.04 & 6.41 & 1.20 & 8.71 & 7.65 & 8.23 \end{bmatrix}$$

Variable 3

$$\begin{bmatrix} 8.01 & 3.56 & 4.70 & 0.73 & 3.39 & 8.73 \\ 2.87 & 5.43 & 5.76 & 7.57 & 1.25 & 0.11 \\ 9.41 & 9.26 & 0.12 & 6.63 & 4.81 & 8.90 \end{bmatrix}$$

Variable 4

$$\begin{bmatrix} 5.78 & 1.77 & 8.64 & 6.94 & 0.30 & 8.29 \\ 5.97 & 2.40 & 9.11 & 3.22 & 0.67 & 3.54 \\ 9.26 & 5.62 & 8.29 & 0.19 & 8.64 & 0.29 \end{bmatrix}$$

dataset

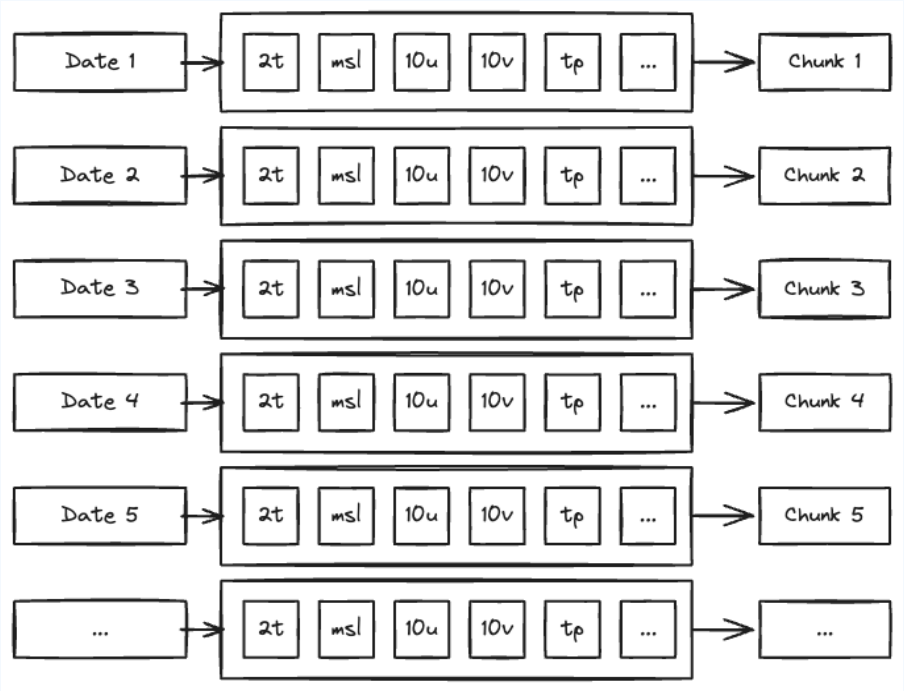
$$\begin{bmatrix} nan & nan & nan & nan & nan & nan & nan & nan & nan & nan & nan & nan \\ nan & nan & nan & nan & nan & nan & nan & nan & nan & nan & nan & nan \\ nan & nan & nan & nan & nan & nan & nan & nan & nan & nan & nan & nan \\ nan & nan & nan & nan & nan & nan & nan & nan & nan & nan & nan & nan \\ nan & nan & nan & nan & nan & nan & nan & nan & nan & nan & nan & nan \\ nan & nan & nan & nan & nan & nan & nan & nan & nan & nan & nan & nan \end{bmatrix}$$

Using Zarr

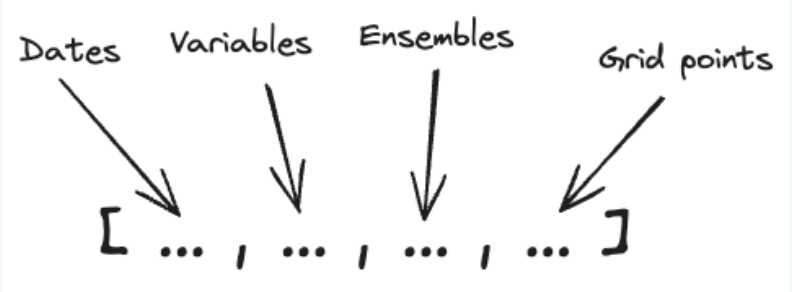
Zarr offers an array-like view on a collection of files.

We use it to define an array whose chunks match a training sample *exactly* — **one file per date**.

ONE SAMPLE = ONE CHUNK READ



INTERNAL DIMENSIONS

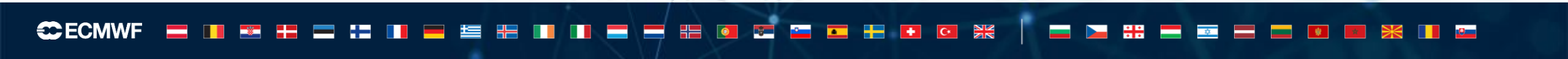


WHY ZARR FITS

- Array-like view over a directory of chunk files
- Chunk shape mirrors the training access pattern
- Cloud- and POSIX-friendly, language-agnostic

Building datasets

YAML-controlled pipelines — self-documenting and reproducible.



Dataset layouts

Three types of data organisation — pick the one that matches how your data lives in time and space.

Gridded

Model fields on a (possibly unstructured) spatial grid, regular in time.

most common

Tabular

Unstructured in both time and space.

observations

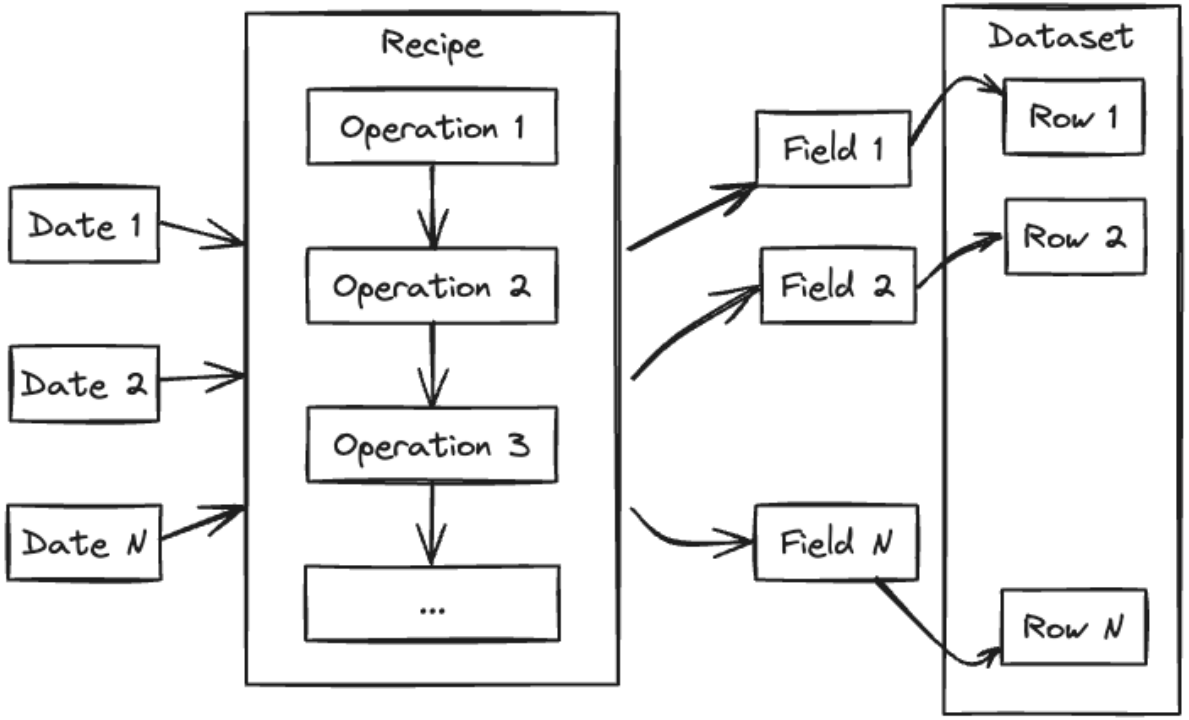
Trajectories

Forecast fields indexed by base date *and* forecast step — a 5-D on-disk array.

5-D

Datasets are built from a recipe

Dates flow through a sequence of operations. Each date produces one field, which becomes one row of the final dataset.



- Self Documenting
- Reproducible

Recipes are YAML files

recipe.yaml

```yaml
dates:
  start: 2024-01-01T00:00:00Z
  end: 2024-01-01T18:00:00Z
  frequency: 6h

input:
  grib:
    param: [2t, msl, 10u, 10v, lsm]
    path: /path/to/some/data.grib
```

TWO MAIN SECTIONS

dates

What time range and frequency.

input

Where the data comes from and how it's combined.

THEN BUILD IT

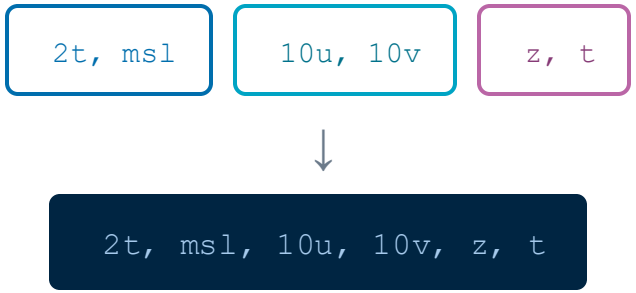
```
$ anemoi-datasets create $recipe.yaml dataset.zarr
```

join

combine variables

Combine several sources of data. Each source provides **different variables** at the **same dates**.

```
input:
  join:
  - source1: …
  - source2: …
  - source3: …
  - …
```

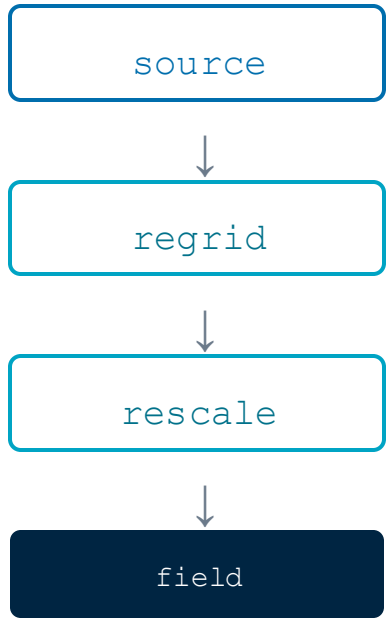


pipe

transform fields

Transform fields using a sequence of filters. The first step is typically a **source**; the rest are **filters**.

```
input:
  pipe:
  - source: …
  - filter1: …
  - filter2: …
  - …
```

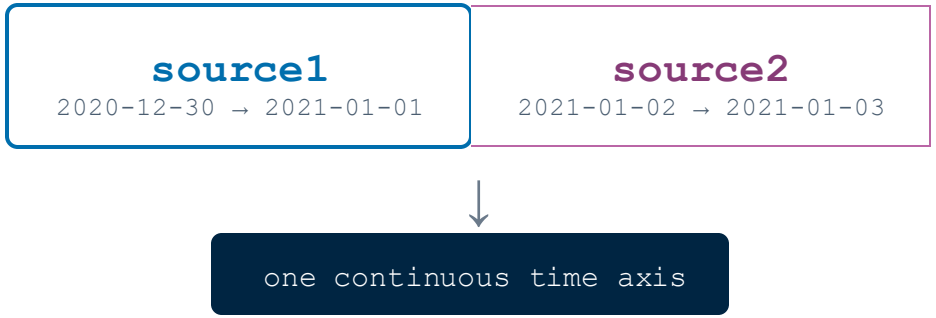


concat

combine dates

Combine different sets of operations that handle **different dates**.

```yaml
input:
  concat:
    - dates:
        start: 2020-12-30 00:00:00
        end: 2021-01-01 12:00:00
        frequency: 12h
        source1: …
    - dates:
        start: 2021-01-02 00:00:00
        end: 2021-01-03 12:00:00
        frequency: 12h
        source2: …
```



Trajectories

Create a **forecast dataset**. Same sources, same operations — now five-dimensional, indexed by base date *and* lead time.

Enables: Temporal Downscaler, and including the forecast dimension

```yaml
steps:
   start: 6h
   end: 30h
   frequency: 1h

input:
# same sources / operations
```

FIVE DIMENSIONS

dates	base date — when the forecast starts
steps	lead time — *new*, 5-D axis
variables	meteorological fields
ensembles	members
grid	spatial points

Statistics gathering

Statistics are computed on a **subset** of dates by default — to keep a tail of truly unseen data.

DEFAULT SUBSET RULE

`≥20y` If the dataset covers **20 years or more**, the last **3 years** are excluded.

`≥10y` If the dataset covers **10 years or more**, the last **year** is excluded.

`else` Otherwise, **80%** of the dataset is used.

USER-PROVIDED DATES

```
statistics:
    end: 2020
```

ALLOW NANS

By default, fails if a NaN is found.

```
statistics:
    allow_nans: true
```

STATISTICS OF INCREMENTS

Also computable for common lead times:

`1h` `3h` `6h` `12h` `24h`

General-purpose sources

NAME	DESCRIPTION	EXAMPLE

`grib`	Access GRIB files on disk.	`grib:`
		`param: [msl, 2t]`
		`path: /path/to/file.grib`

`netcdf`	Access NetCDF files on disk (xarray-based).	`netcdf:` `path: /path/to/file.nc`
`opendap`	Access OpenDAP servers (xarray-based).	`opendap:` `url: http://…`
`xarray-zarr`	File-based or cloud-based Zarr (xarray-based).	`xarray-zarr:` `url: http://…`
`xarray-kerchunk`	Access kerchunk datasets (xarray-based).	EXPERIMENTAL
`zenodo`	Access a dataset given a Zenodo DOI.	EXPERIMENTAL

General-purpose sources

NAME	DESCRIPTION	EXAMPLE
`bufr`	Access BUFR files on disk.	`bufr:` `…`
`anemoi-dataset`	Access an existing Anemoi dataset.	`anemoi_dataset:` `path: /path/to/anemoi/dataset`
`more…`	And more, plus organisation-specific sources contributed by individual sites.	EXTENSIBLE

REFERENCE

Filters anemoi-transforms

A small selection — work in progress. Some sites implement their own filters too.

NAME	DESCRIPTION
`apply-mask`	Apply a mask to the fields (set values to NaN according to a mask).
`rotate_wind / unrotate_wind`	Rotate wind vectors.
`rename`	Rename variables.
`regrid`	Interpolate input fields to a different grid.
`rescale`	Apply unit conversions.
`orog_to_z`	Convert orography (m) to z (geopotential height).
`uv-2-ddff / ddff-2-uv`	Convert wind components between cartesian and polar coordinates.
`wz_to_w`	Convert geometric vertical velocity (m/s) to vertical velocity (Pa/s).

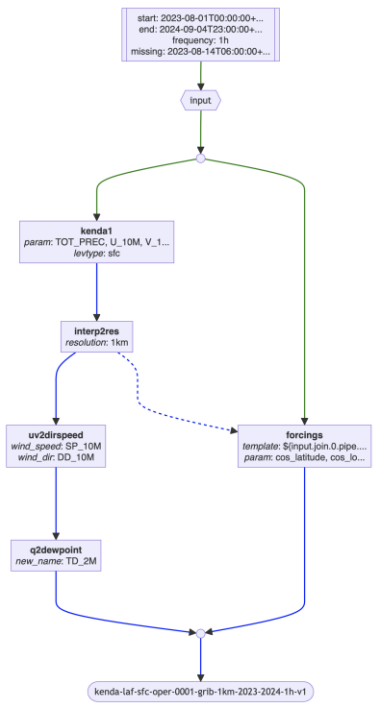
IN PRACTICE

Real recipes scale up

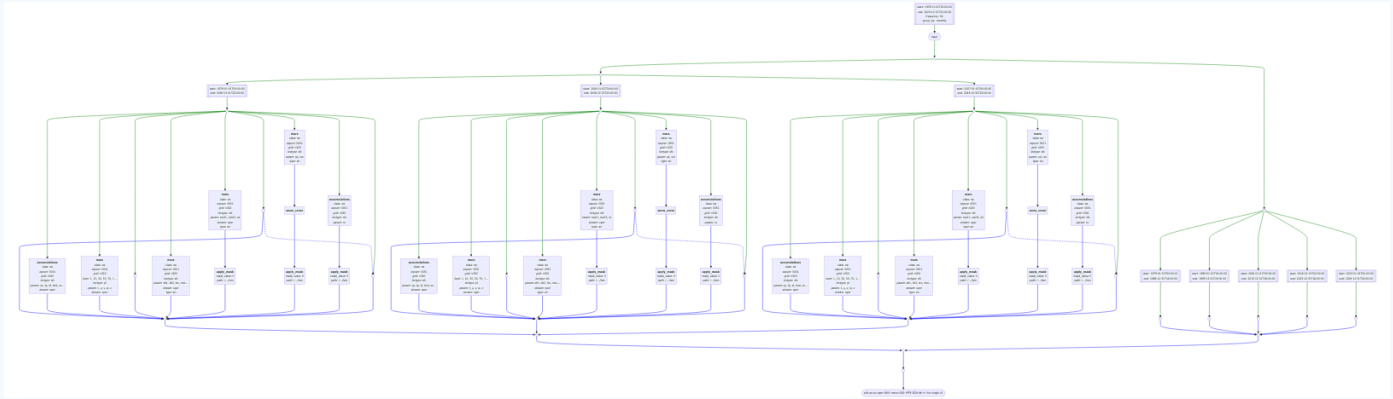
Production datasets stitch together many sources, filters and date ranges.

A SMALL ONE

~10 nodes

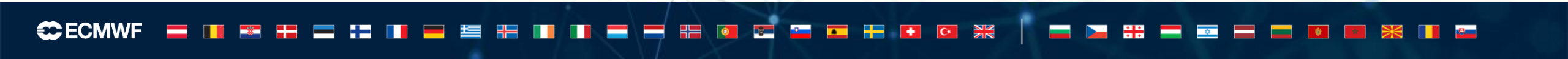


A BIGGER ONE



Using datasets

One Python entry point. Lazy slicing, selecting, combining.



Basics

To open a dataset, use the `open_dataset` function:

```
from anemoi.datasets import open_dataset

ds = open_dataset("/path/to/dataset.zarr")
```

Then access data via `ds` as if it were a `NumPy` array:

```
print(ds.shape)
print(len(ds))
print(ds[0])
print(ds[10:20])
```

Basics (cont.)

1 Ability to **subset**:

```python
from anemoi.datasets import open_dataset

ds = open_dataset("path/to/dataset.zarr",
     start=2000,
     end=2020)
```

2 Or **combine** datasets:

```python
from anemoi.datasets import open_dataset

ds = open_dataset("path/to/dataset1.zarr",
     "path/to/dataset2.zarr")
```

> **Lazy by default** — these operations compose without touching the disk. Reading happens only when you index into `ds`.

Opening a dataset

Two equivalent forms — pick whichever fits your config file or call site.

KEYWORD ARGUMENTS

```python
from anemoi.datasets import open_dataset

ds = open_dataset(dataset,
     option1=value1,
     option2=…)
```

or

DICTIONARY FORM

```python
from anemoi.datasets import open_dataset

ds = open_dataset({
   "dataset": dataset,
   "option1": value1,
   "option2": …,
})
```

Selecting variables

`list` keep + preserve order

```
ds = open_dataset(
  dataset,
  select=["2t", "tp"],
)
```

`set` keep, any order

```
ds = open_dataset(
  dataset,
  select={"2t", "tp"},
)
```

`drop` remove specific ones

```
ds = open_dataset(
  dataset,
  drop=["10u", "10v"],
)
```

Reordering variables

… using a **list**

```
ds = open_dataset(
dataset,
   reorder=[
   "2t", "msl", "sp",
   "10u", "10v",
   ],
)
```

… or using a **dictionary**

```
ds = open_dataset(
dataset,
   reorder={
   "2t": 0,
   "msl": 1,
   "sp": 2,
   "10u": 3,
   "10v": 4,
   },
)
```

cutout

limited-area + global

Use a **limited-area model** where you have it; fall back to the **global model** everywhere else.

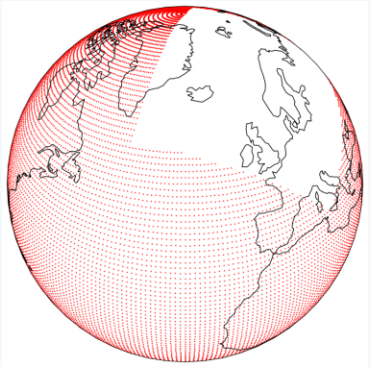
```python
from anemoi.datasets import open_dataset

ds = open_dataset(
  cutout=[
    lam_dataset,
    global_dataset,
  ],
)
```

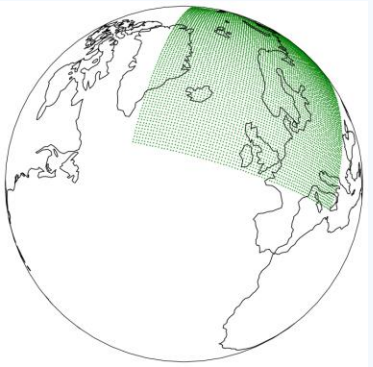
LEGEND

- LAM points (high resolution, regional)
- Global points (used outside the LAM region)

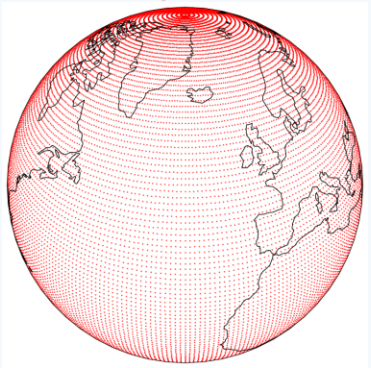
Global only



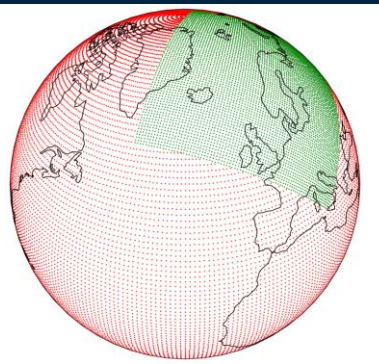
LAM only



Overlap removed



Cutout result



REFERENCE

Attributes

Rich metadata is attached to every dataset — available without reading a single chunk.

shape	A tuple of the dataset's dimensions.
field_shape	Original 1D or 2D shape of a single field (fields are flattened to 1D when building).
dtype	The dataset's NumPy data type.
dates	Dates as a NumPy vector of `datetime64`.
frequency	Delta between two consecutive dates.
latitudes	Latitudes as a NumPy vector.
longitudes	Longitudes as a NumPy vector.
resolution	The dataset's resolution.
statistics	`mean`, `stdev`, `minimum`, `maximum` per variable.
name_to_index	Dict mapping variable names → channel index. `ds.name_to_index["2t"]`
variables	Variable names in dataset order.
missing	Set of indices of the missing dates.

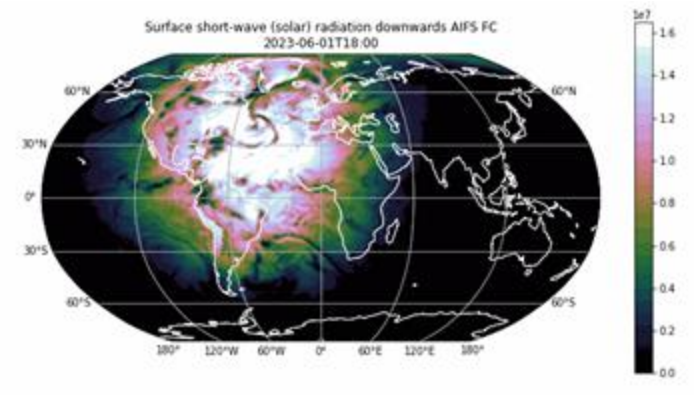
```
stats = ds.statistics; stats["mean"][ds.name_to_index["msl"]]
```

When combining datasets, the statistics of the first encountered dataset are used.

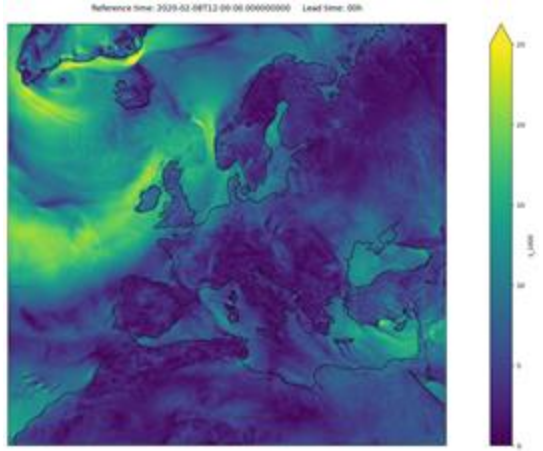
What does Anemoi enable?

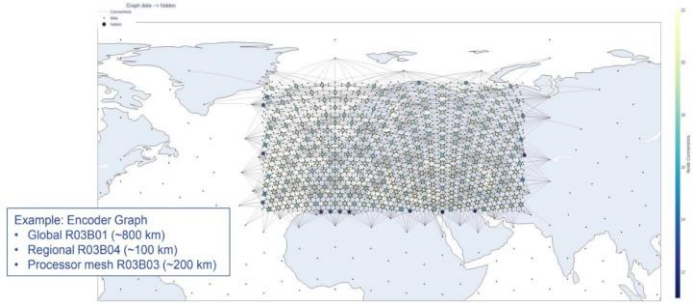
Datasets

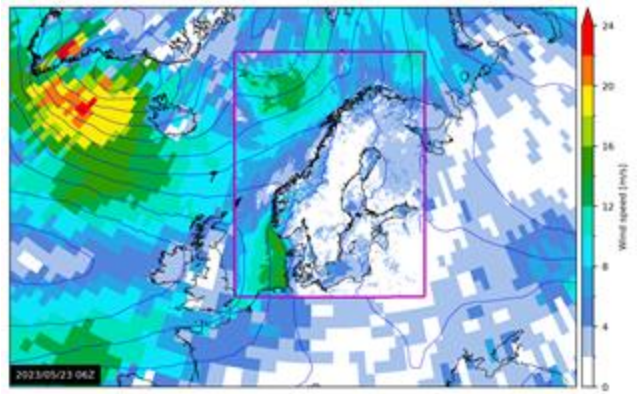
Global



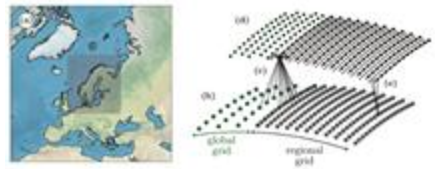
Regional Area Modelling







Stretched Grid Model



Nipen et al. 2024

DISCOVER ANEMOI · WEBINAR SERIES · PART 1

Thank you

Questions?

www.ecmwf.int

PRESENTER
Ana Prieto Nemesio
Machine Learning Team Lead
ana.prietonemesio@ecmwf.int

PRESENTER
Harrison Cook
Research Software Engineer
harrison.cook@ecmwf.int