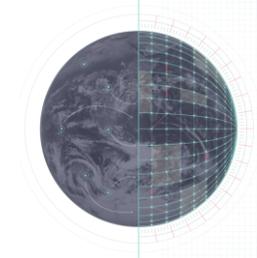


Anemoi Overview, Datasets and Inference

Harrison Cook

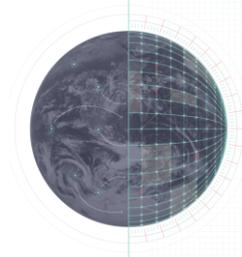


On behalf of the Anemoi team and contributors



Goals

- Develop an open-source framework to facilitate:
 - the development of data-driven weather forecasts...
 - ... and their running in operations
- In collaboration with the Centre's Member States and others
 - Support both global and limited area models



Anemoi

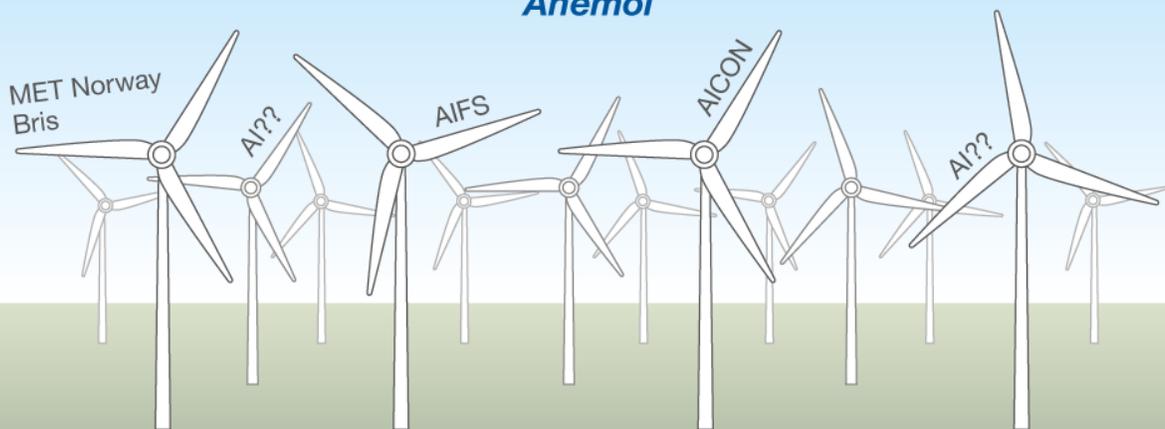
MET Norway
Bris

AI??

AIFS

AICON

AI??

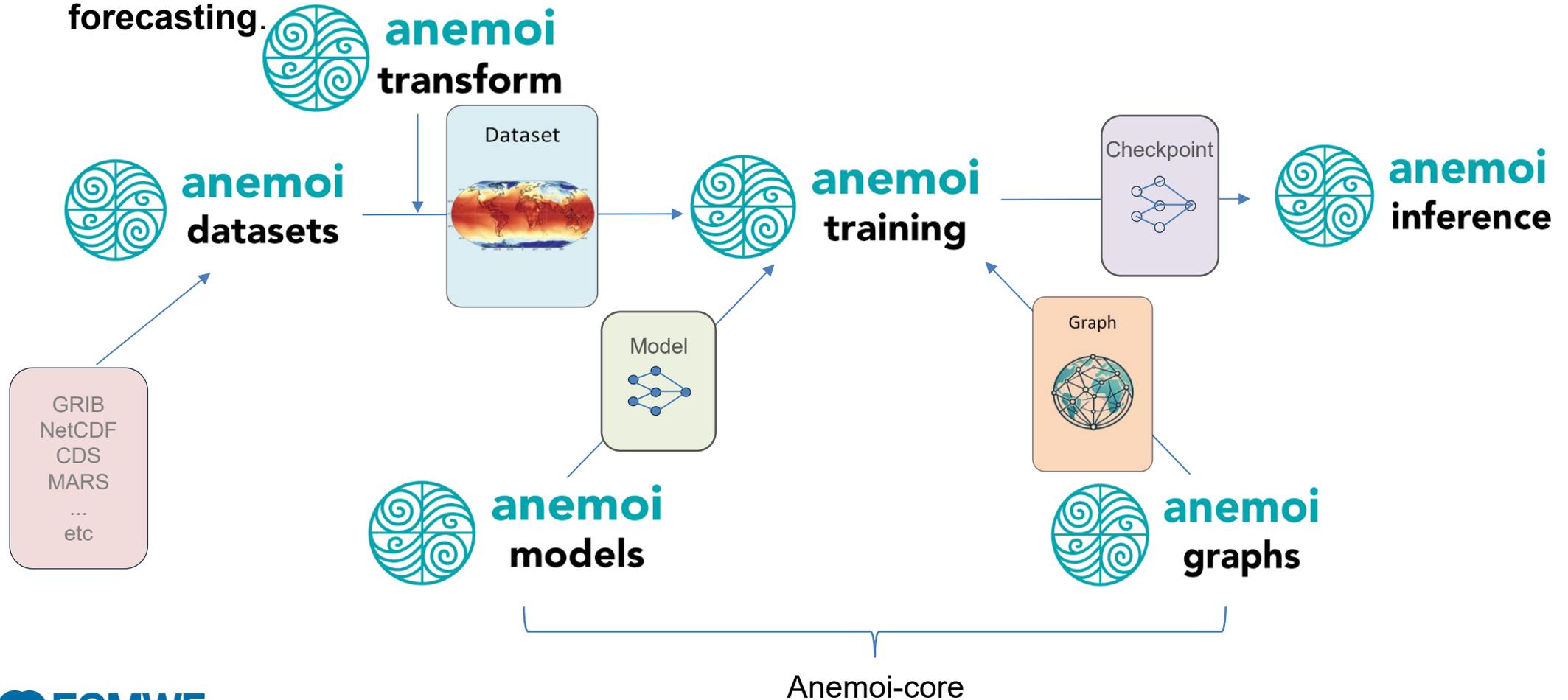


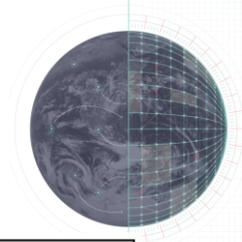


Anemoi in a Nutshell

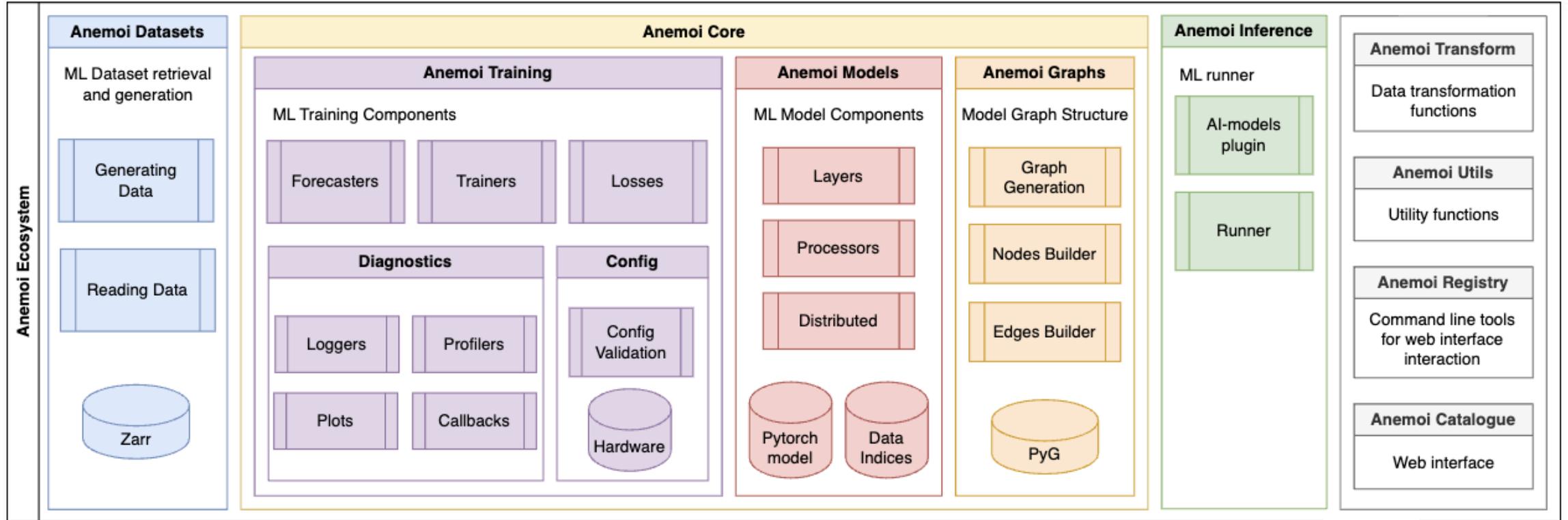
- Anemoi is an open-source framework to develop **data-driven meteorological forecasting**.

- Anemoi is developed through a **collaborative European initiative**





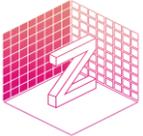
Anemoi's modular structure and Python packages



Dependencies: (selection)



earthkit



Zarr



PyTorch



PyG



Numpy



HYDRA



Pydantic



PyTorch Lightning



mlflow™



User Base

Researchers

Developers

Operators

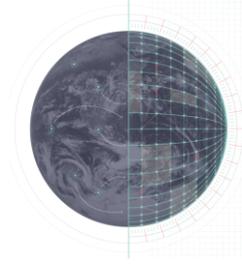


User Base

Modifies Configs for
Experimentation and
Implementation of
Operational Models

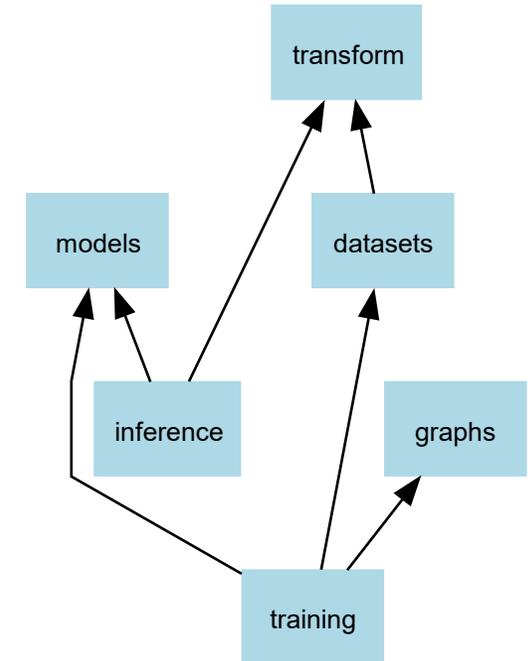
Modifies Codebase
to implement new
Features and
Augment Anemoi
Libraries

Runs the Anemoi
Model in a common
interface on reliable
infrastructure

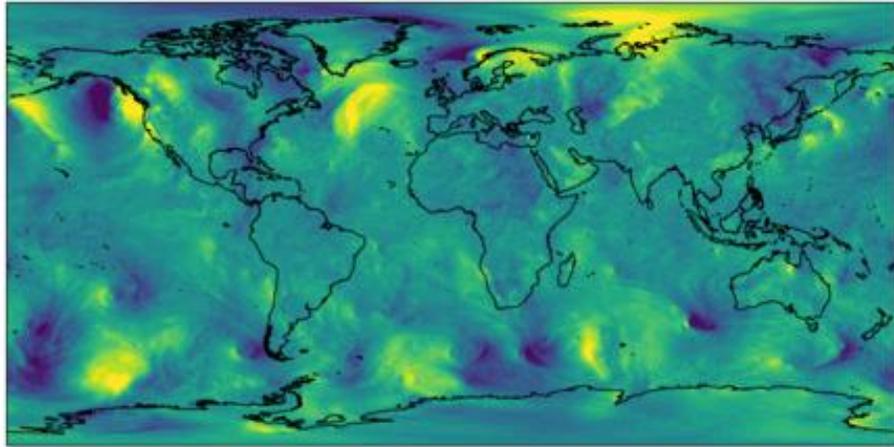


Main components - Design decisions

- OO design, heavy use of factories to instantiate object from configs
- Each component provides one or more command line tools
 - anemoi-datasets create data-config.yaml out.zarr
 - anemoi-training train train-config.yaml
 - anemoi-inference run inference-config.yaml
- Minimise software dependencies to facilitate R2O
 - Inference and training are independent
- Each component collects “metadata” that can be used by the others



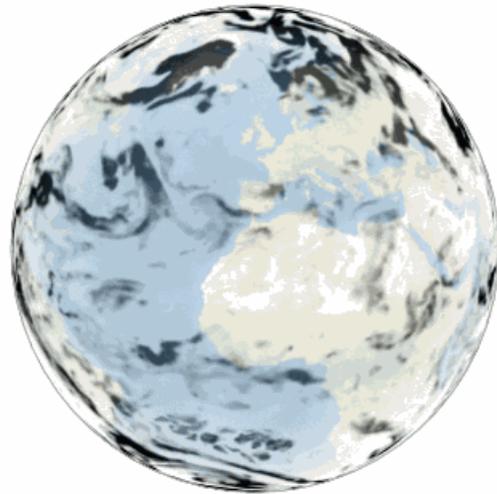
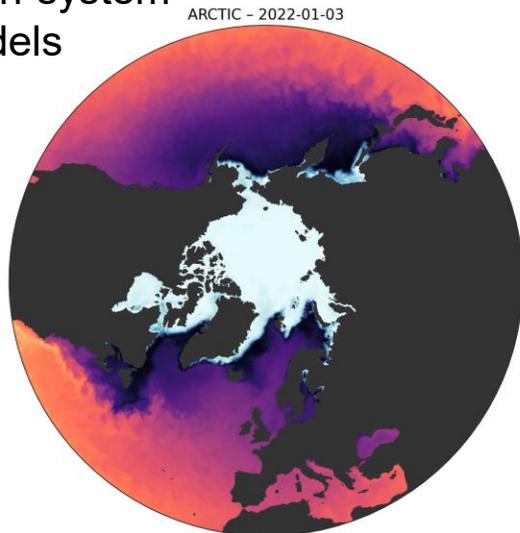
What Anemoui enables?



Training deterministic & ensemble models

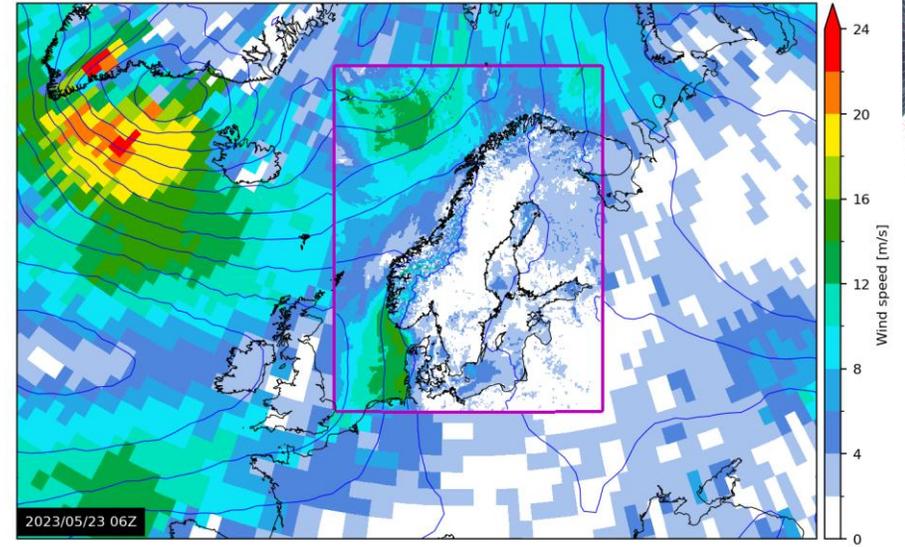
Lang et al. 2024a & b

Building coupled earth-system models



Regional high-resolution modelling:

Nipen et al. 2024



Real-time running at operation centres
Anemoui powers ECMWF's operationalisation of AIFS in 2025.

Latest point-based forecast

AIFS ENS Meteograms

AIFS ENS Meteograms show a probabilistic interpretation of the AIFS ENS forecasts for specific locations using a box and whisker plot. It shows the time evolution of the distribution of several meteorological parameters on a single diagram...

Latest forecast

AIFS Single: Total cloud cover

This display helps with the recognition of clouds of different layers, even when they overlap. Brighter colouring represents greater cloud cover. Cloud-free areas appear white while areas of full cloud cover at all levels appear dark grey (e.g. active fronts)...



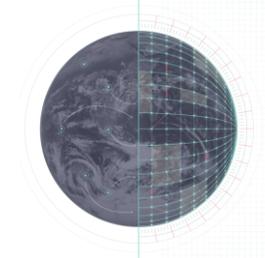
Anemoi - Datasets

Machine Learning for Weather Prediction - 2026



Datasets - Goals

- Create “machine-learning ready” datasets for training data-driven weather forecasts
- Make the loading of data samples as efficient as possible
- Provide rich metadata that can be used in training and inference



Datasets - Non-goals

- Is not a replacement for the original source of data
- Consider the datasets created by anemoi-datasets is pre-processed data that are as close as possible to the needs of training



Training data-driven forecasts

- A “sample” is the state of the atmosphere a time T
- We want to train a model that, given the state of the atmosphere a time T returns the state of the atmosphere a $T+1$
- The state of the atmosphere is a collections of meteorological fields (variables)
 - 2m temperature, surface pressure,...
 - geopotential, winds, temperature on 1000 hPa, 500 hPa, ...



Training a weather forecast

```
x, y = ds[n : n + 1]
y_hat = model.predict(x)
loss = model.loss(y, y_hat)
```

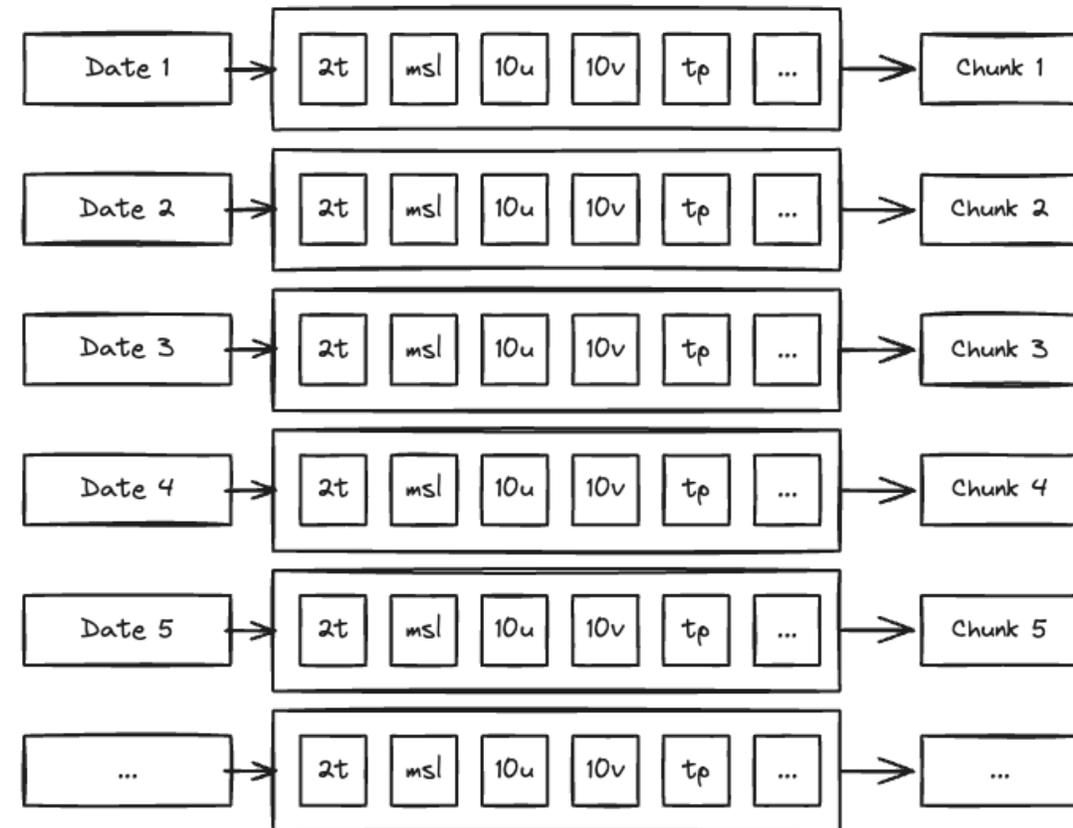
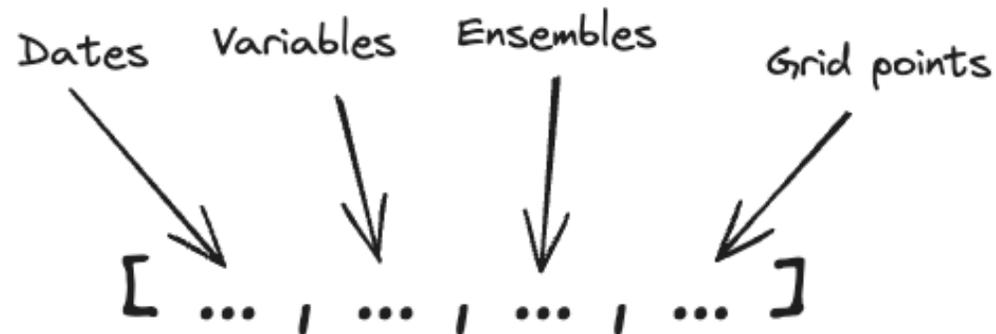


- A dataset like ERA5 is too large to fit in memory
- We need to store the data on a filesystem
- We want to create a dataset where each sample is one retrieval operation



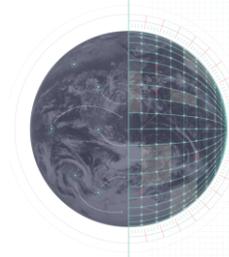
Using Zarr

- The Zarr format is a good fit for the requirements
- It offers an array-like view on a collection of files (chunks)
- We use it to define an array that match exactly a sample
- Each file is a single date

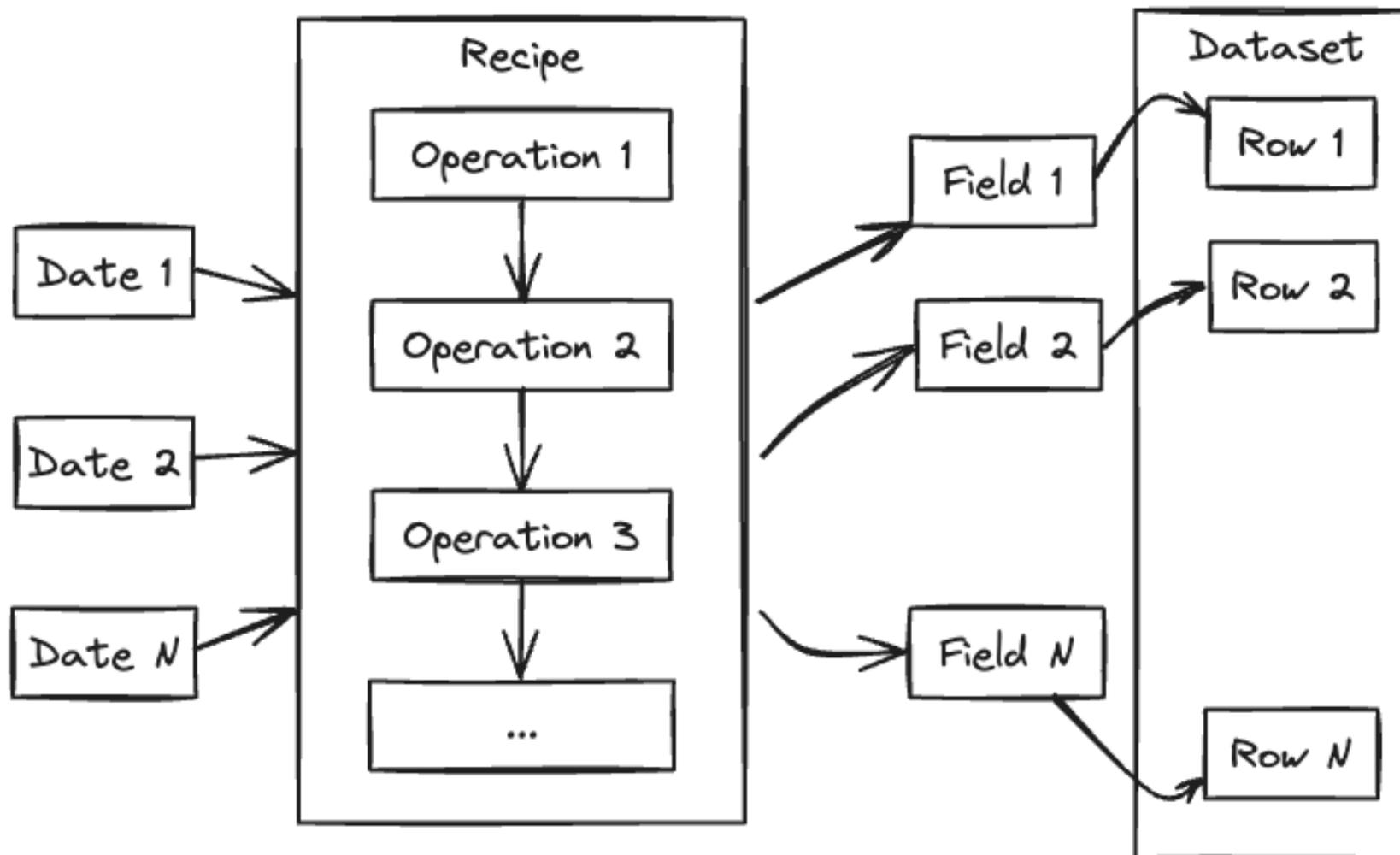




Building datasets



Datasets are built from a recipe





Recipes are YAML files

dates:

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

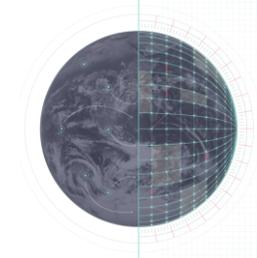
input:

grib:

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

Then:

```
% anemoui-datasets create recipe.yaml dataset.zarr
```



Join

The join is the process of combining several sources data. Each source is expected to provide different variables at the same dates.

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



Pipe

The pipe is the process of transforming fields using filters. The first step of a pipe is typically a source. The following steps are filters.

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



Concat

The concatenation is the process of combining different sets of operation that handle different dates.

```
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:
        ...
```



Statistics gathering

- By defaults, the statistics are not computed on the whole dataset, but on a subset of dates.
- The dates subset used to compute the statistics is defined using the following algorithm:
 - If the dataset covers 20 years or more, the last 3 years are excluded.
 - If the dataset covers 10 years or more, the last year is excluded.
 - Otherwise, 80% of the dataset is used.
 - Using user provided dates

```
statistics:  
  end: 2020
```

- By default, statistics will fail if a NaN is found. You can change that:

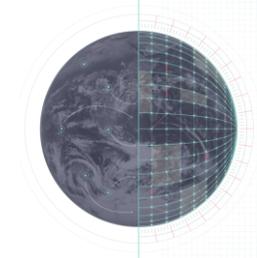
```
statistics:  
  allow_nans: true
```

- Statistics of increments can also be computed:
 - 1h, 3h, 6h 12h and 24h increments



General purposes sources

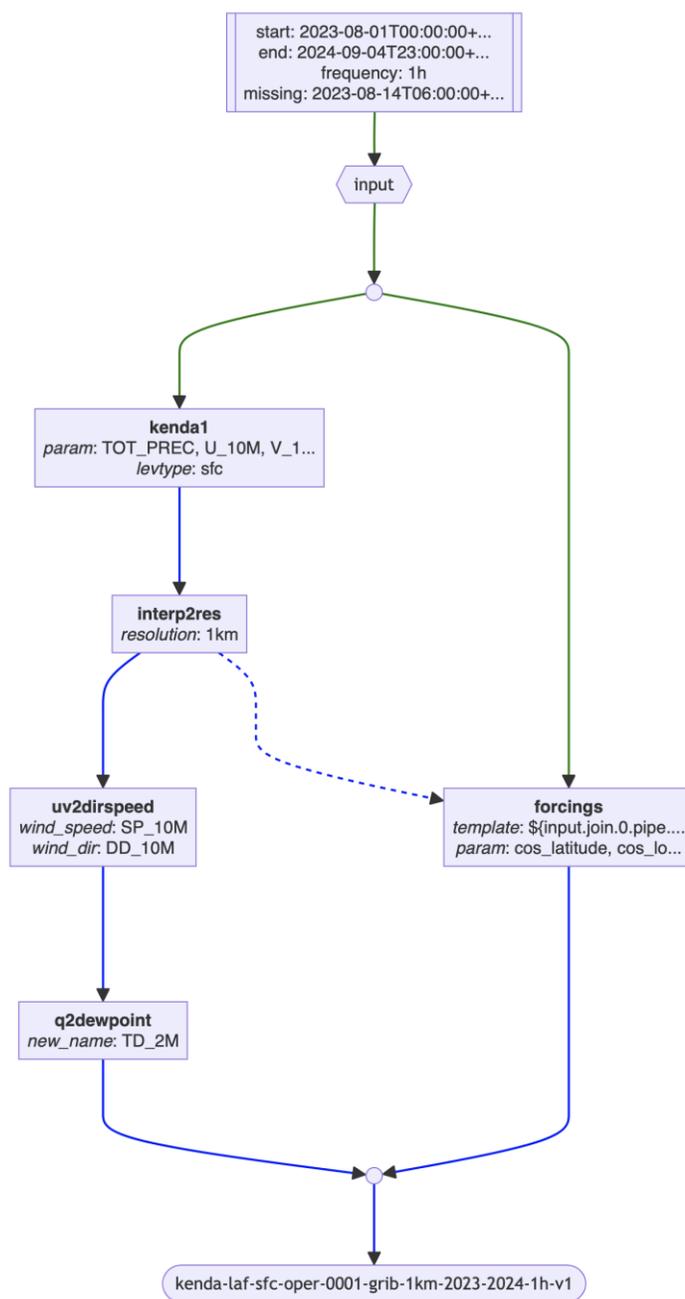
Name	Description
<code>grib</code>	Access grib files on disk <pre>input: grib: param:[msl, 2t] path:/path/to/file.grib</pre>
<code>netcdf</code>	Access NetCDF files on disk (xarray-based) <pre>input: netcdf: path:/path/to/file.nc</pre>
<code>opendap</code>	Access OpenDAP servers (xarray-based) <pre>input: opendap: url: http://</pre>
<code>xarray-zarr</code>	Access file-based or cloud-based Zarr (xarray-based) <pre>input: xarray-zarr: url: http://</pre>
<code>xarray-kerchunk</code>	Experimental: Access kerchunk datasets (xarray-based)
<code>zenodo</code>	Experimental: access a dataset given a Zenodo DOI



Filters

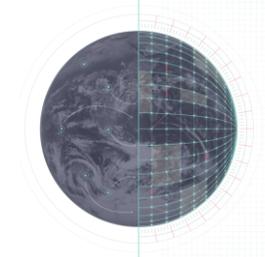
Name	Description
apply-mask	Apply a mask to the fields (set values to NaN according to a mask)
rotate_wind/unrotate_wind	Rotate wind vectors
rename	Rename variables
regrid	Interpolate input fields to a different grid
rescale	Apply unit conversions
orog_to_z	Convert orography (m) to z (geopotential height)
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)

- This is only a small selection, work in progress
- Some sites implement their own filters





Using datasets



Introduction

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

```
from anemoi.datasets import open_dataset  
  
ds = open_dataset("/path/to/dataset.zarr")
```

You can then access the data in the dataset using the `ds` object as if it was a **NumPy** array:

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



Introduction (cont.)

One of the main features is the ability to subset:

```
from anemoi.datasets import open_dataset
```

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

or combine datasets:

```
from anemoi.datasets import open_dataset
```

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



Opening a dataset

```
from anemoi.datasets import open_dataset
```

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

or

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

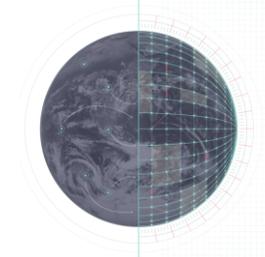


Selecting variables

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

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

```
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,  
    },  
)
```

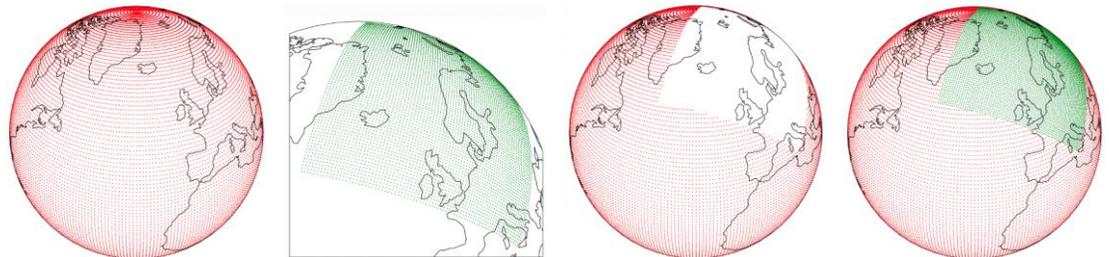
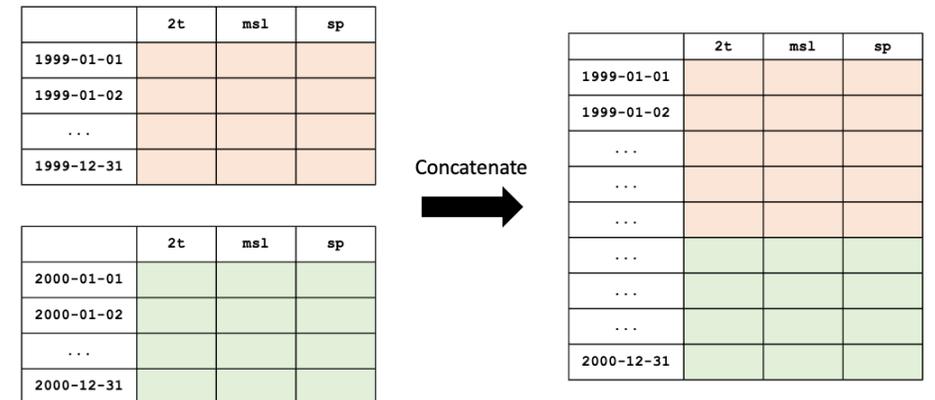
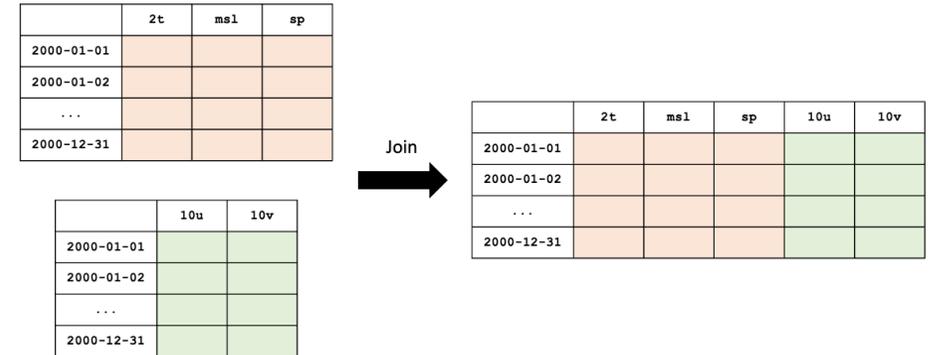
anemoi-datasets (usage)

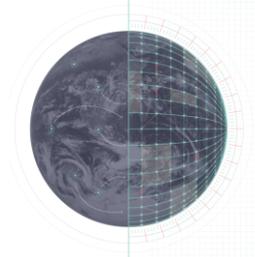
- When used, datasets can be lazily combined into “virtual” datasets
 - Allowing researchers to find the best combinations of variables for their training needs

```

from anemoi.datasets import open_dataset

ds = open_dataset(
    [
        {
            "dataset": "aifs-od-an-oper-0001-mars-o96-2016-2023-6h-v6",
            "drop": ["sp", "2d", "skt", "tcw", "cp"],
            "end": 2023,
            "frequency": "6h",
        },
        {
            "dataset": "aifs-od-an-oper-0001-mars-o96-2016-2023-6h-v1-precipitations",
            "end": 2023,
            "frequency": "6h",
            "rename": {"tp_0h_12h": "tp"},
            "select": ["tp_0h_12h"],
        },
    ],
    end=202311,
)
    
```

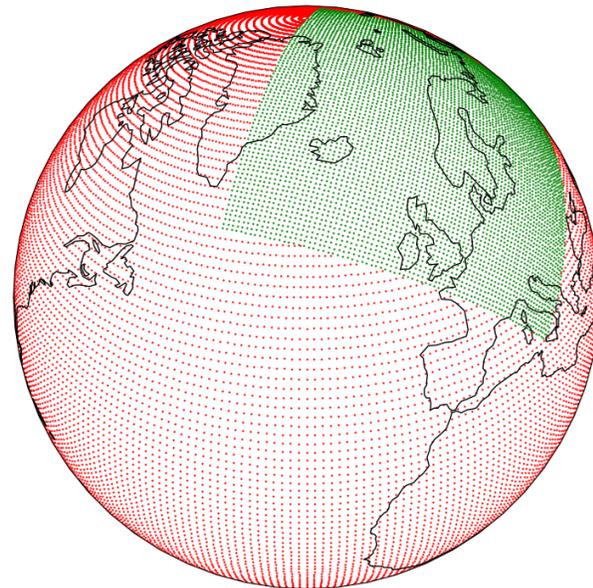
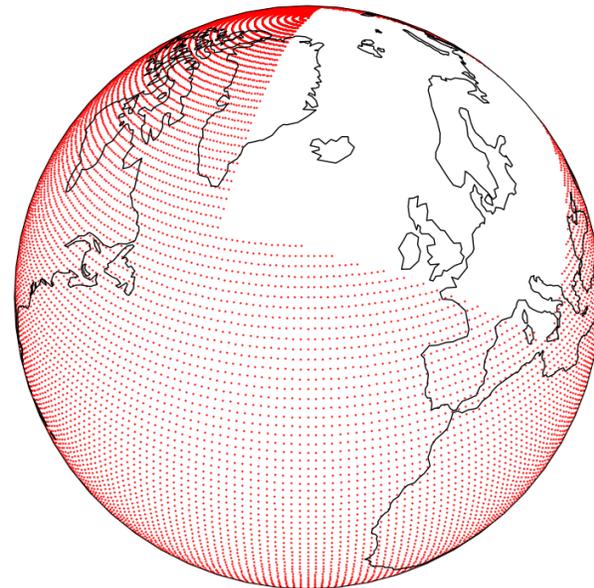
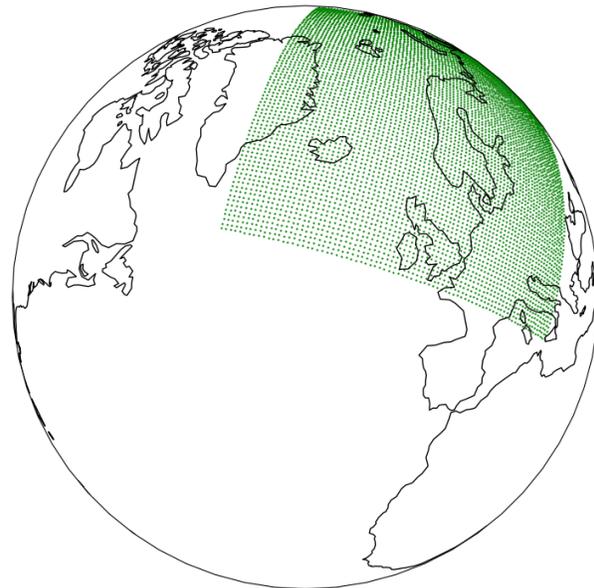
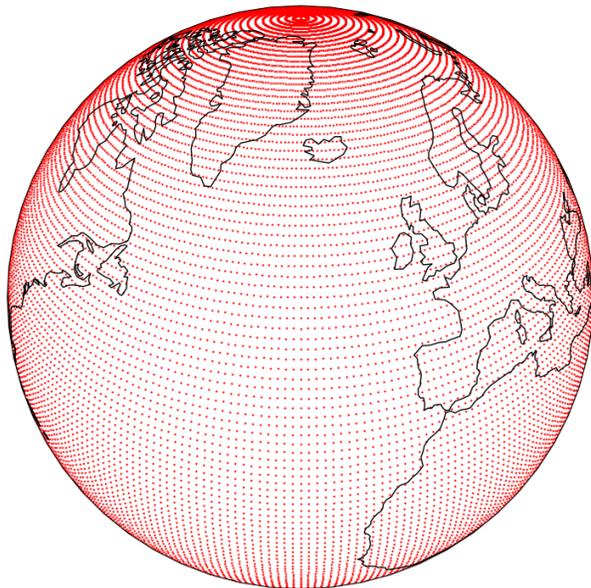


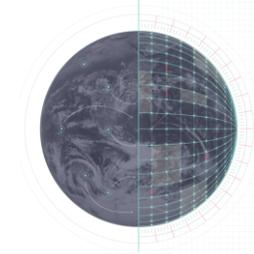


Cutout - Use to combine limited area models and global models

```
from anemoi.datasets import open_dataset
```

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





Attributes

Attribute	Description
<code>shape</code>	A tuple of the dataset's dimensions.
<code>field_shape</code>	The original shape of a single field, either 1D or 2D. When building datasets, the fields are flattened to 1D.
<code>dtype</code>	The dataset's NumPy data type.
<code>dates</code>	The dataset's dates, as a NumPy vector of datetime64 objects.
<code>frequency</code>	The dataset's frequency (i.e the delta between two consecutive dates)
<code>latitudes</code>	The dataset's latitudes as a NumPy vector.
<code>longitudes</code>	The dataset's longitudes as a NumPy vector.
<code>statistics</code>	(Next slide)
<code>resolution</code>	The dataset's resolution.
<code>name_to_index</code>	A dictionary mapping variable names to their indices. <pre>print(dataset.name_to_index["2t"])</pre>
<code>variables</code>	A list of the dataset's variable names, in the order they appear in the dataset.
<code>missing</code>	The set of indices of the missing dates.



Statistics

```
ds = open_dataset(...)  
stats = ds.statistics  
  
stats["mean"]  
stats["stdev"]  
stats["minimum"]  
stats["maximum"]  
  
idx = ds.name_to_index("msl")  
  
print(stats["mean"][idx], stats["stdev"][idx])
```

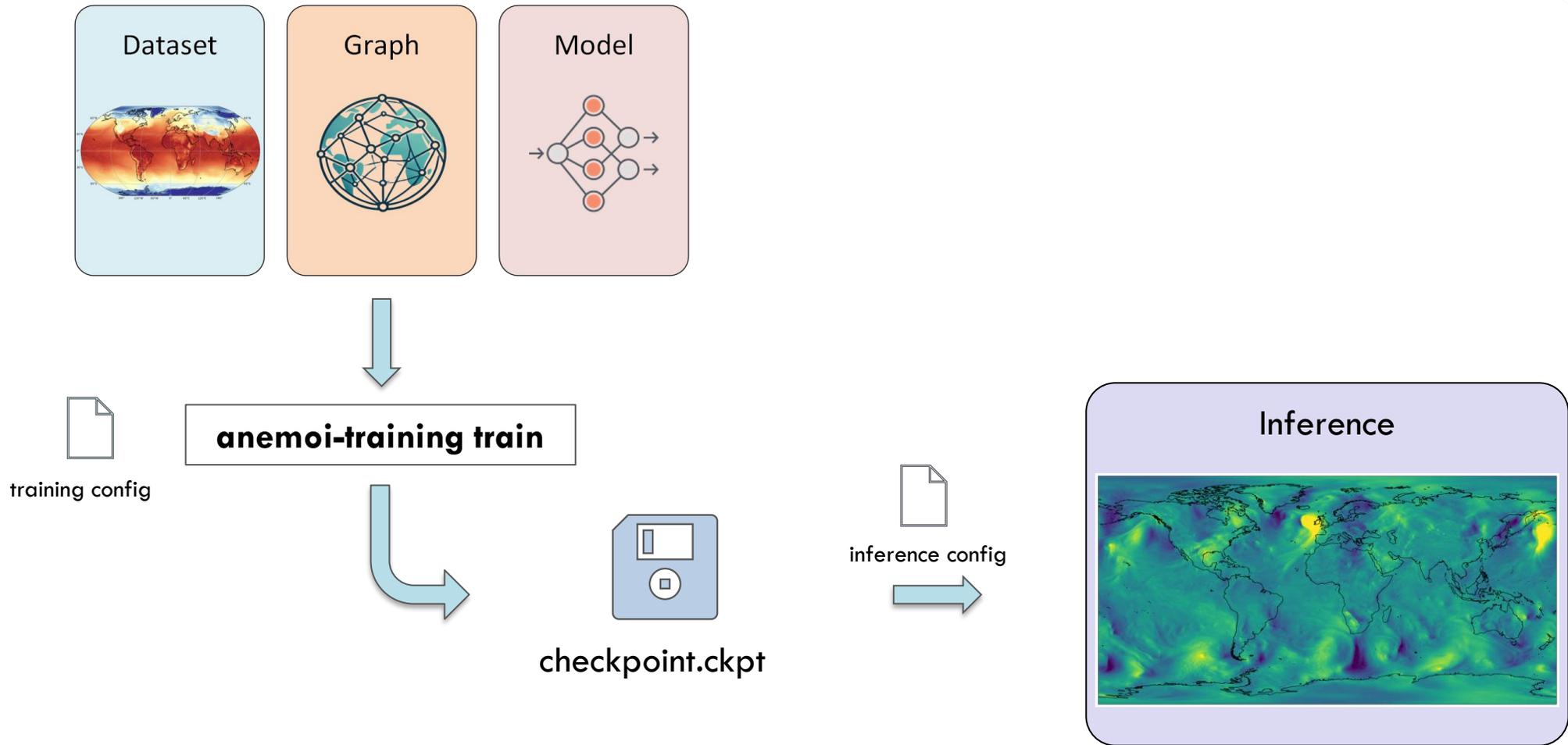
When combining datasets, the statistics of the first encountered dataset is used

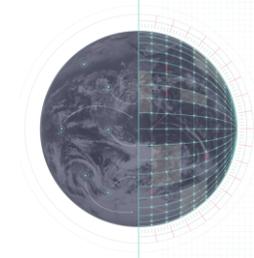


Anemoi - Inference

Machine Learning for Weather Prediction - 2026



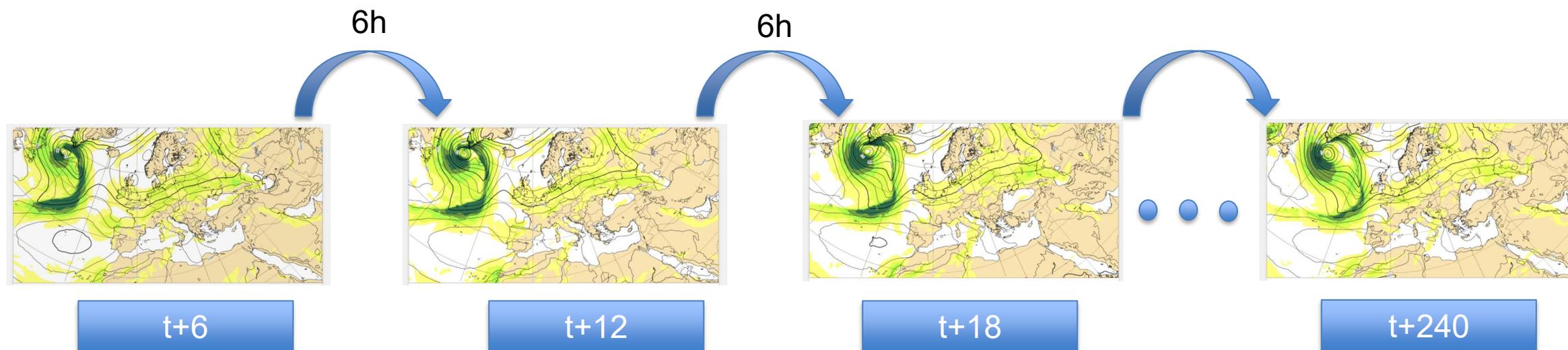




Inference for the forecasting task

- If a model predicts $t+6$
 - Any time a multiple of 6 is possible

- Autoregressive rollout



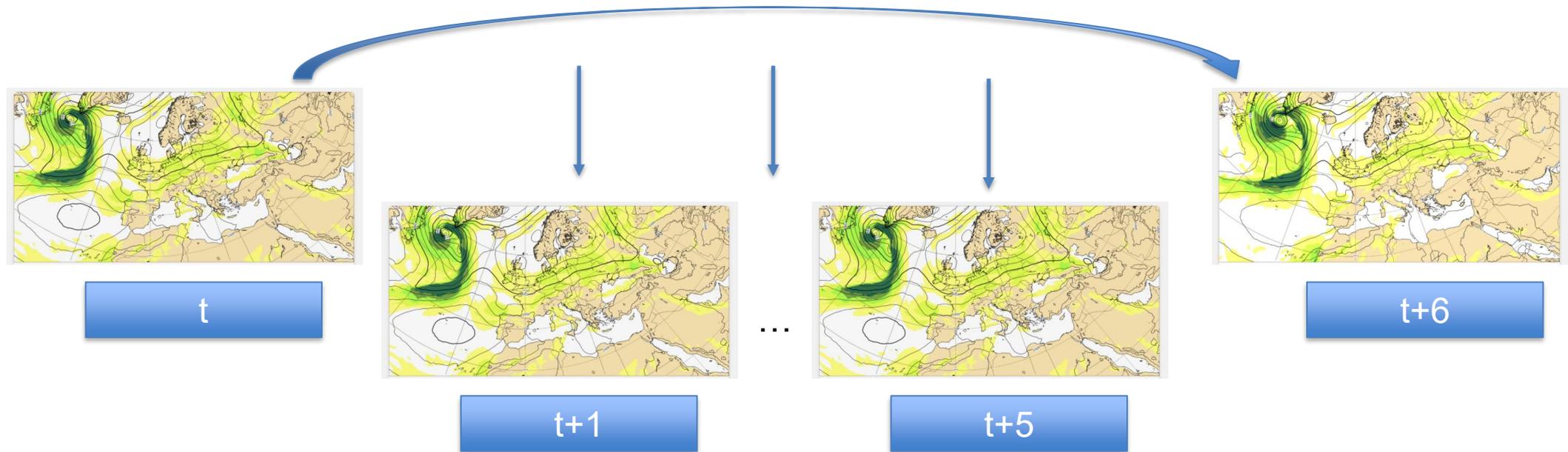


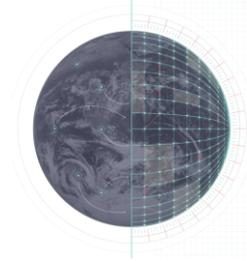
Inference for temporal interpolation

- With t & $t+6$

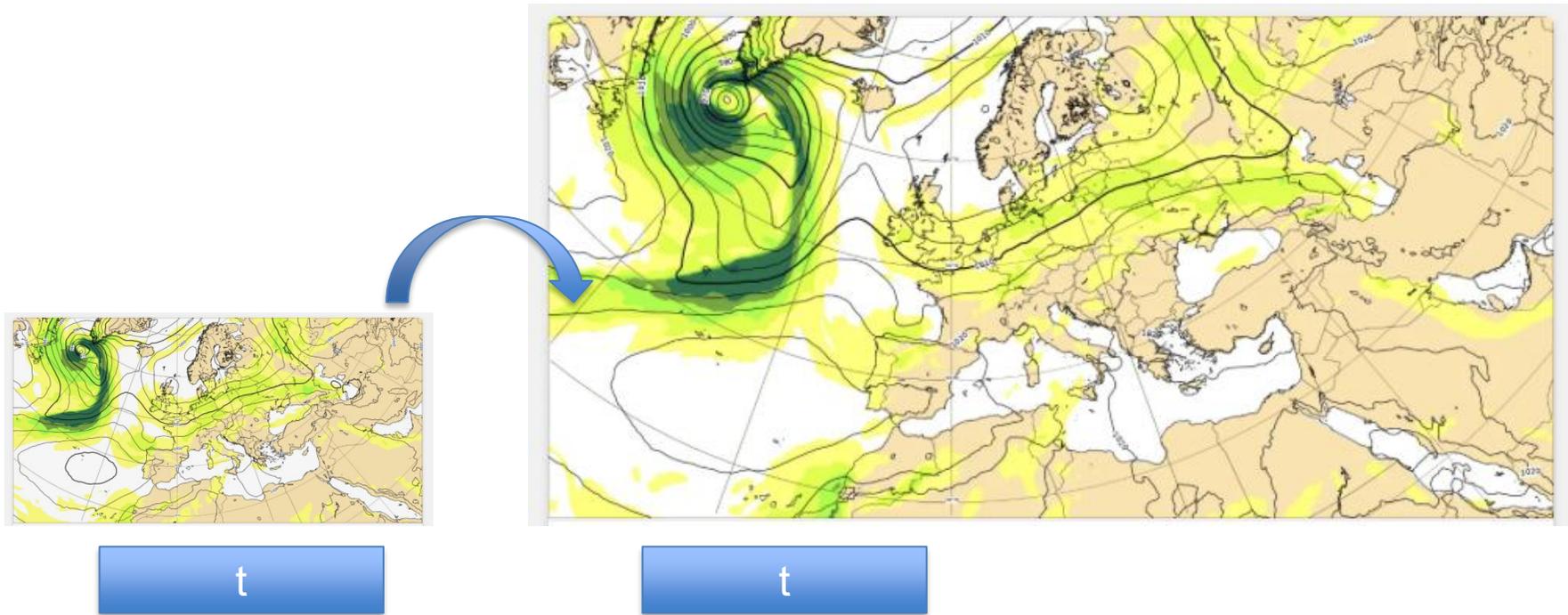
- Predict $t+1 \rightarrow 5$

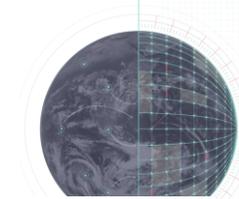
6h





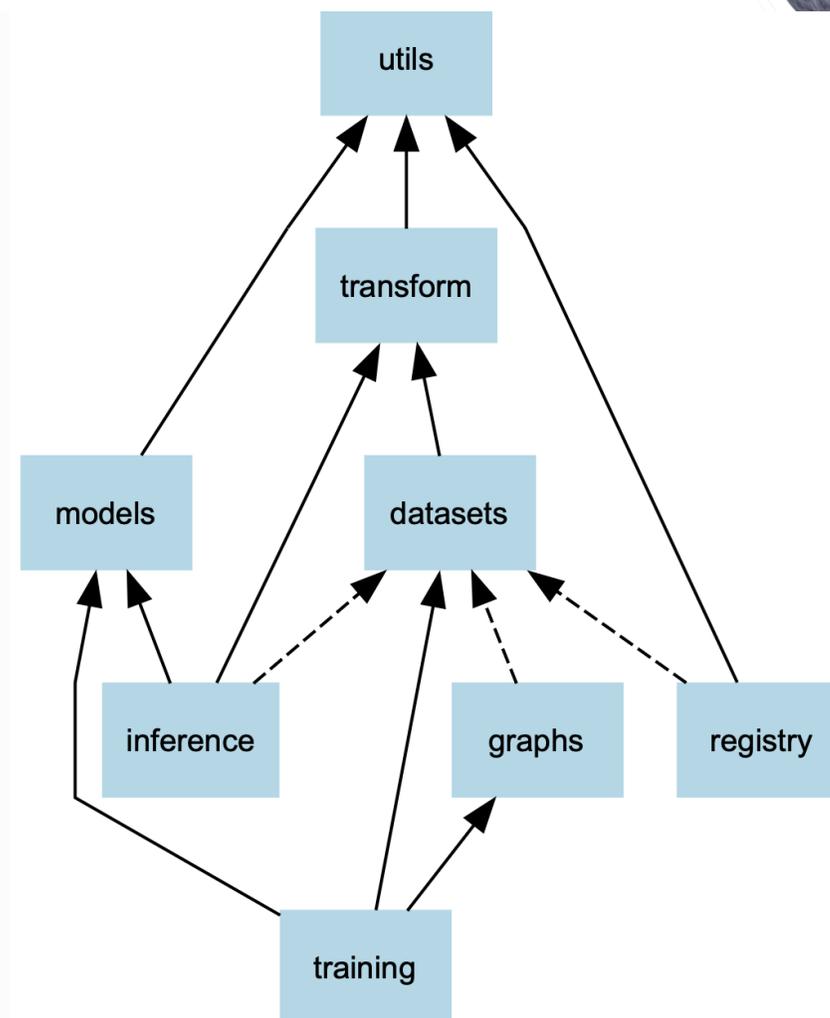
Inference for downscaling





Rationale

- Run trained models in inference mode
- Community project
 - "Any" model can be run
 - But Operationalisation in mind
- Focus on the inference problem
 - Minimise unnecessary dependencies
 - Different data sources
 - Limited compute
 - Integration with external tooling





Inference - Three different API levels

- NumPy to NumPy low level API
- Object-oriented high level API

- Command-line API

Command Line Execution

Note:

Other API's for programmatic usage are also available



```
> anemoi-inference run --help  
usage: anemoi-inference run [-h] [--defaults DEFAULTS] config [overrides ...]
```

Run inference from a config yaml file.

positional arguments:

config	Path to config file. Can be omitted to pass config with overrides and defaults.
overrides	Overrides as key=value

options:

-h, --help	show this help message and exit
--defaults DEFAULTS	Sources of default values



Command Line Configuration

```
checkpoint: '/path/to/checkpoint.ckpt'

date: 20251030T00
lead_time: 240

input:
  mars:
    log: false

pre_processors:
  - no_missing_values
post_processors:
  - accumulate_from_start_of_forecast

output:
  tee:
    - grib: 'output.grib'
    - netcdf: 'output.nc'
    - plot:
        dir: plots/go/here
        variables: [2t, msl]
        method: overlay
        domain: Europe
```

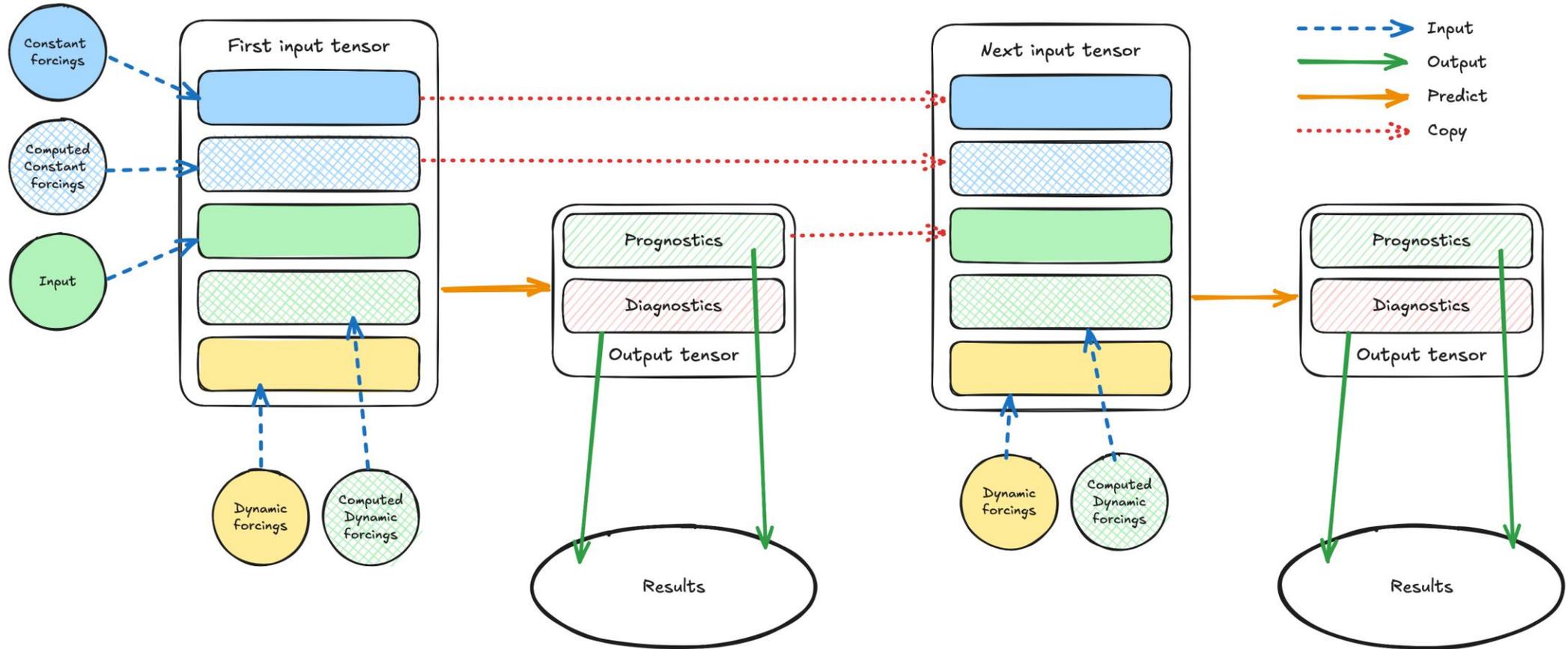
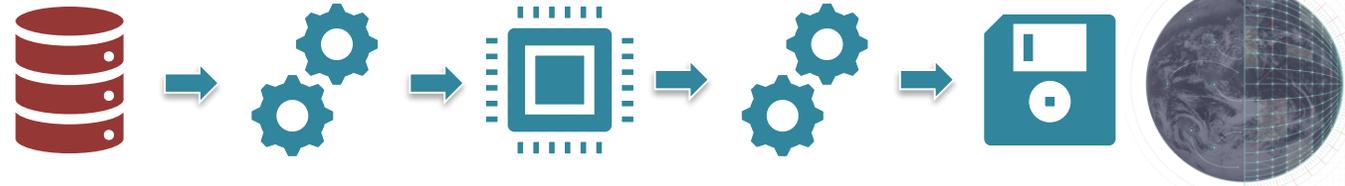
Checkpoints

- Trained models stored in an “atomic” file with metadata
- Metadata:
 - Datasets used for training
 - Operations that took place on the data (e.g.: cutout)
 - Grid & Area
 - Mapping of variables to position in the input tensor
- Automatic setup of inference pipeline
 - List of prognostic, diagnostic and forcing variables
 - Minimal to no configuration required to retrieve initial conditions

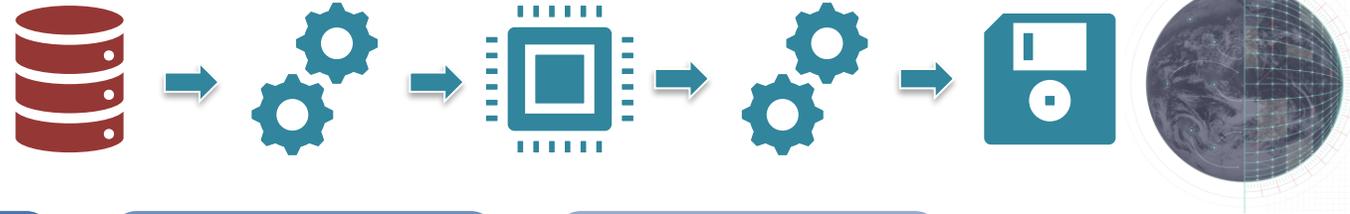


Hugging Face

Inputs



Inputs - Available



- Direct Access to the ECMWF Archive

MARS

- Utilise publicly available ERA5

CDS

- Freely and openly available IFS analysis

Opendata *

- Regional extractions and custom collections

Polytope *

- Directly run from anemoi-datasets

Datasets

- Any local grib file

GRIB

- Merge multiple sources for LAM / Nested Grid models

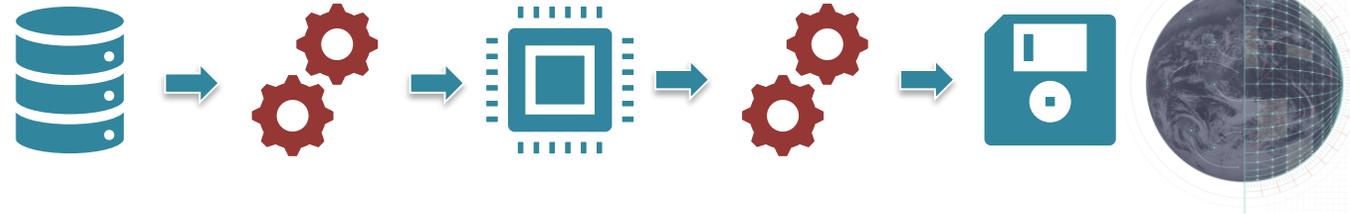
Cutout

- Quickly test your setup works with fake input

Dummy

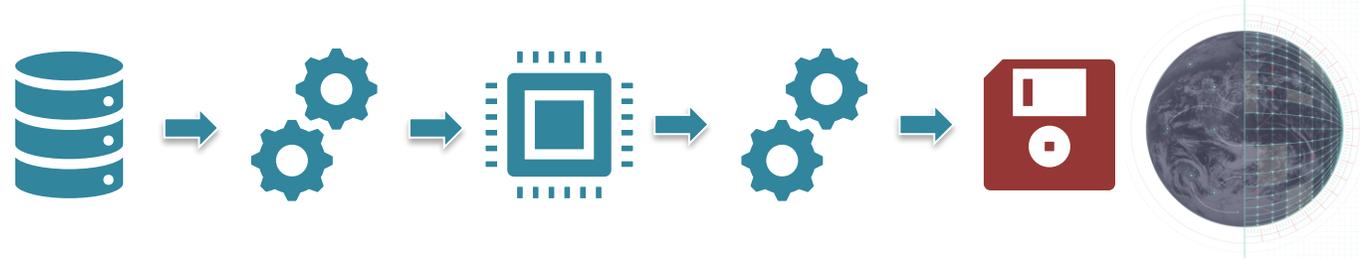
* Available as a plugin

Processors



- Data is weird
 - Comes in many shapes and sizes
- For further modification of the data
 - Prepare the initial conditions
 - Finalise output state

Outputs

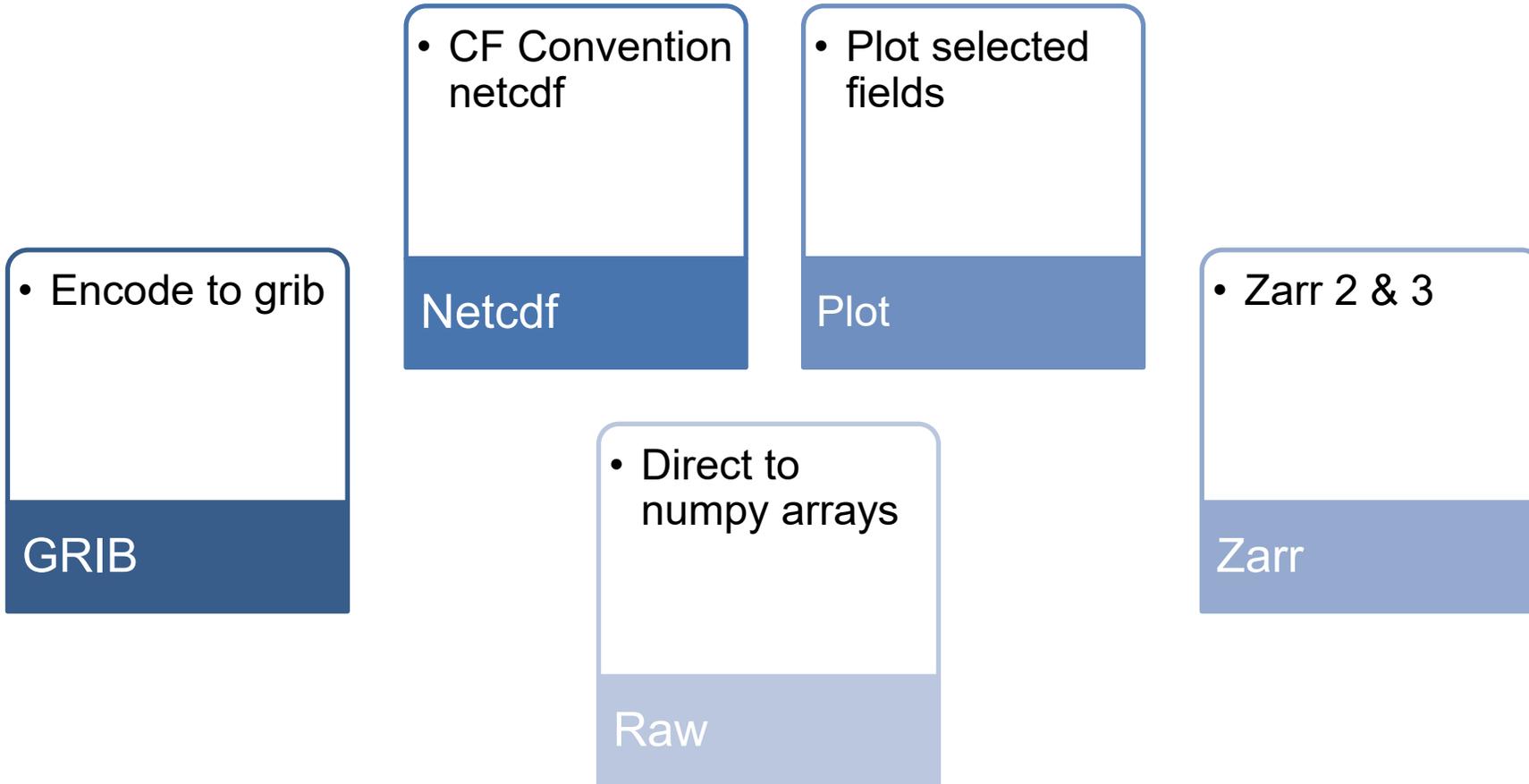
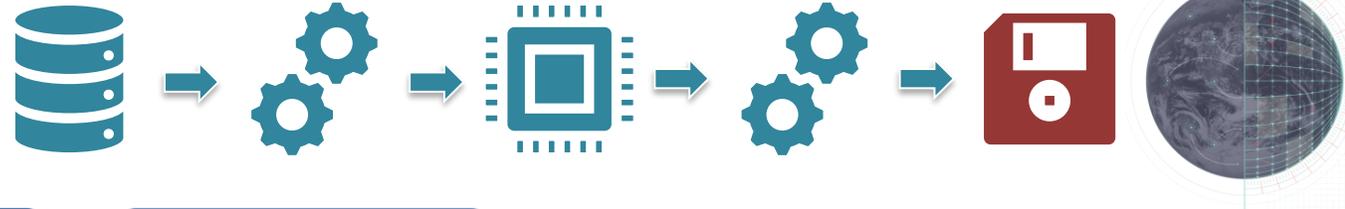


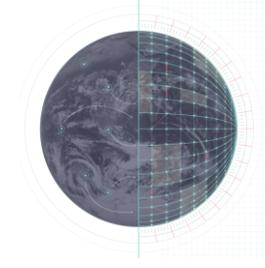
- Write the data to disk
- Major bottleneck in operations
- Save into various formats

Todo:

- Parallel IO
- Streaming to post processing systems

Outputs - Available





Hands On Session

- Run an AIFS yourself
- Remove the training wheels, all configurable
 - Initial conditions
 - Running
 - Outputs