How we build a new AIFS & get it into operations*

Machine Learning Training Course

Harrison Cook and Mariana Clare



This forecast started as a NaN problem

ecmwf/anemoi-models · pull request #56

Feature/mask NaNs in training loss function

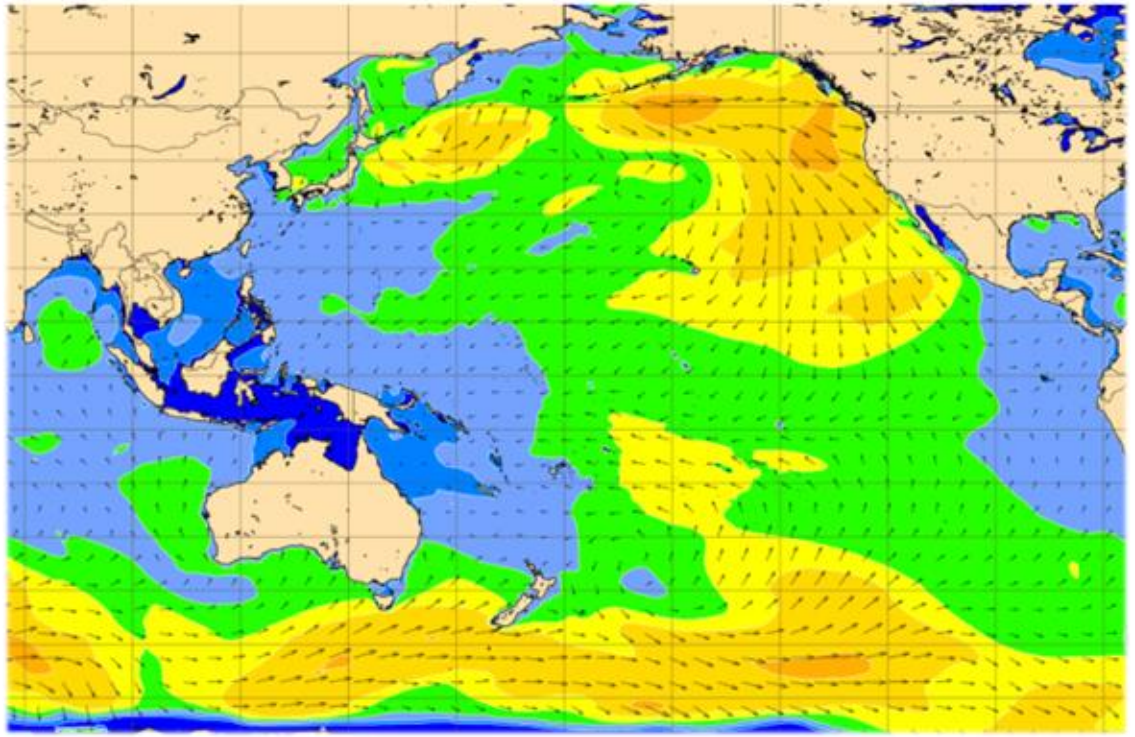
Merged 22 Nov 2024 · shipped in anemoi-models 0.4.1

> *"Variables with missing values that are imputed by the imputer should not be considered in the loss."*

Wave fields are undefined over land: NaNs the network must not learn from.

Companion PR: anemoi-training #72 · both now part of anemoi-core

Where it started



What users download today

Everything between these two pictures is this lecture.

The journey from pull request to forecast

PART 1 · Making an AIFS

1
Contribute
Code into Anemoi

2
Research
Isolate & evaluate

3
Choose
Prioritise & combine

4
Train
At scale on EuroHPC

5
Evaluate
Candidate, 6 months

PART 2 · Putting it into operations

6
Hand over
Research / Development
→ Production

7
Run
Operational suite

8
Package
The module

9
Operations
Real-world challenges

The technical work needs to work hand-in-hand with the scientific development and the operations.

Part 1 · Making an AIFS

From a contribution to a trained, evaluated candidate

Where do contributions come from?

Mostly internal research: scientists who want the model to do something new, or to do something better.

Ocean waves

Add wave variables the AIFS did not forecast before. Our running example.

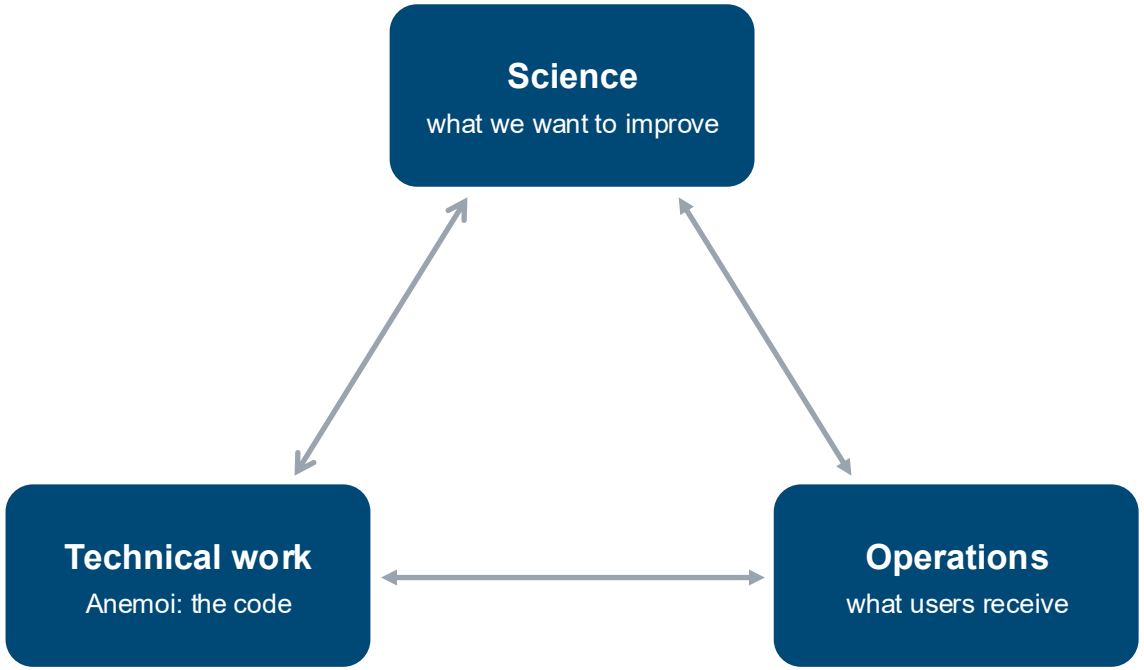
New variables

Tropical precipitation

Reduce a known precipitation bias in the tropics, inherited from the IFS.

Bias reduction

Every contribution touches all three



WAVES THREAD

Needs a new feature in anemoi-core before it can even be trained.

Why every contribution goes to main

WHAT CAN GO WRONG

1. **Built on an old version**

   A contribution was developed on a version that was no longer the baseline.

2. **Compared against the current baseline**

   Differences came from the version gap, not only from the contribution.

3. **Evaluation invalid → retrain**

   ~Weeks of GPU time might need to be repeated on the right version.

WHAT MAIN GIVES US

- **A stable, reproducible baseline**

  When things go wrong (they do), we know what we started from.

- **We can retrain on demand**

  Any operational model can be rebuilt from a tagged version.

- **One environment for everyone**

  Training is easier when nobody runs a private fork.

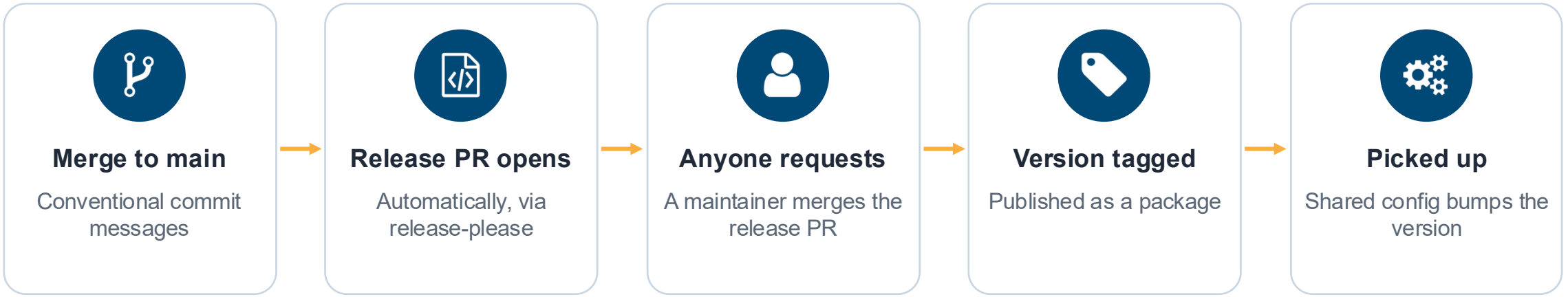
- **Reviewed and tested**

  Every change is checked before others build on it.

WAVES THREAD

Land-NaN masking merged to main, reviewed and tested like any other change.

Making a release



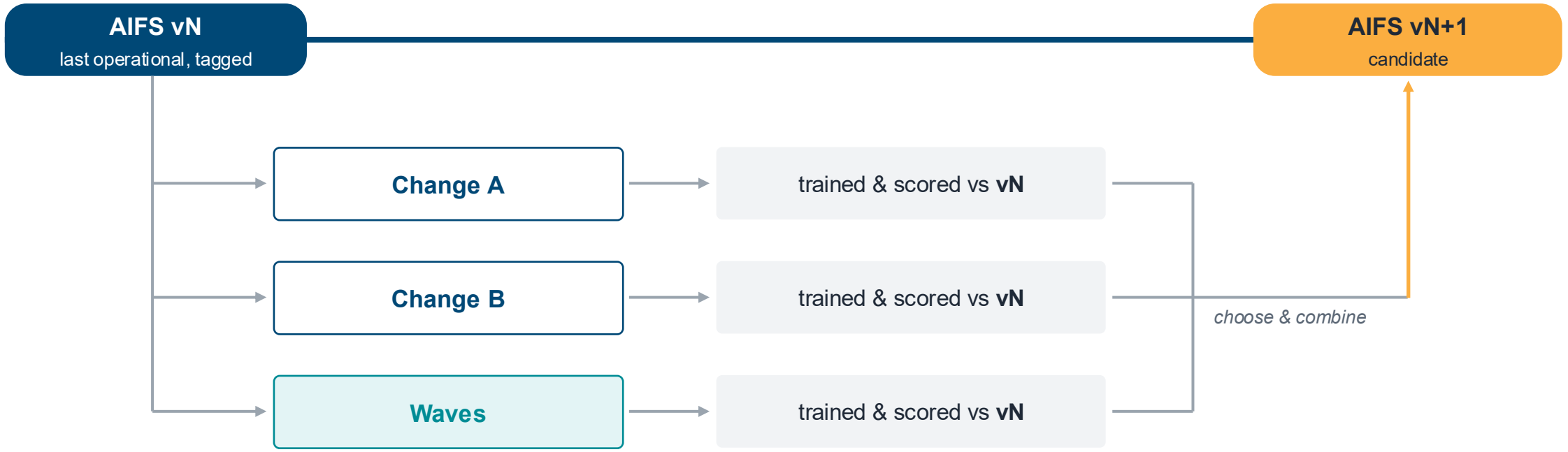
≈ monthly

release cadence at the moment, anyone can ask for one

WAVES THREAD

The NaN-masking feature ships in anemoi-models 0.4.1

The ideal research process



One change per experiment so any difference in the scores belongs to that change.

Mostly at full scale occasionally reduced resolution first; each run costs days of GPU time.

Note distinctly serial

WAVES THREAD

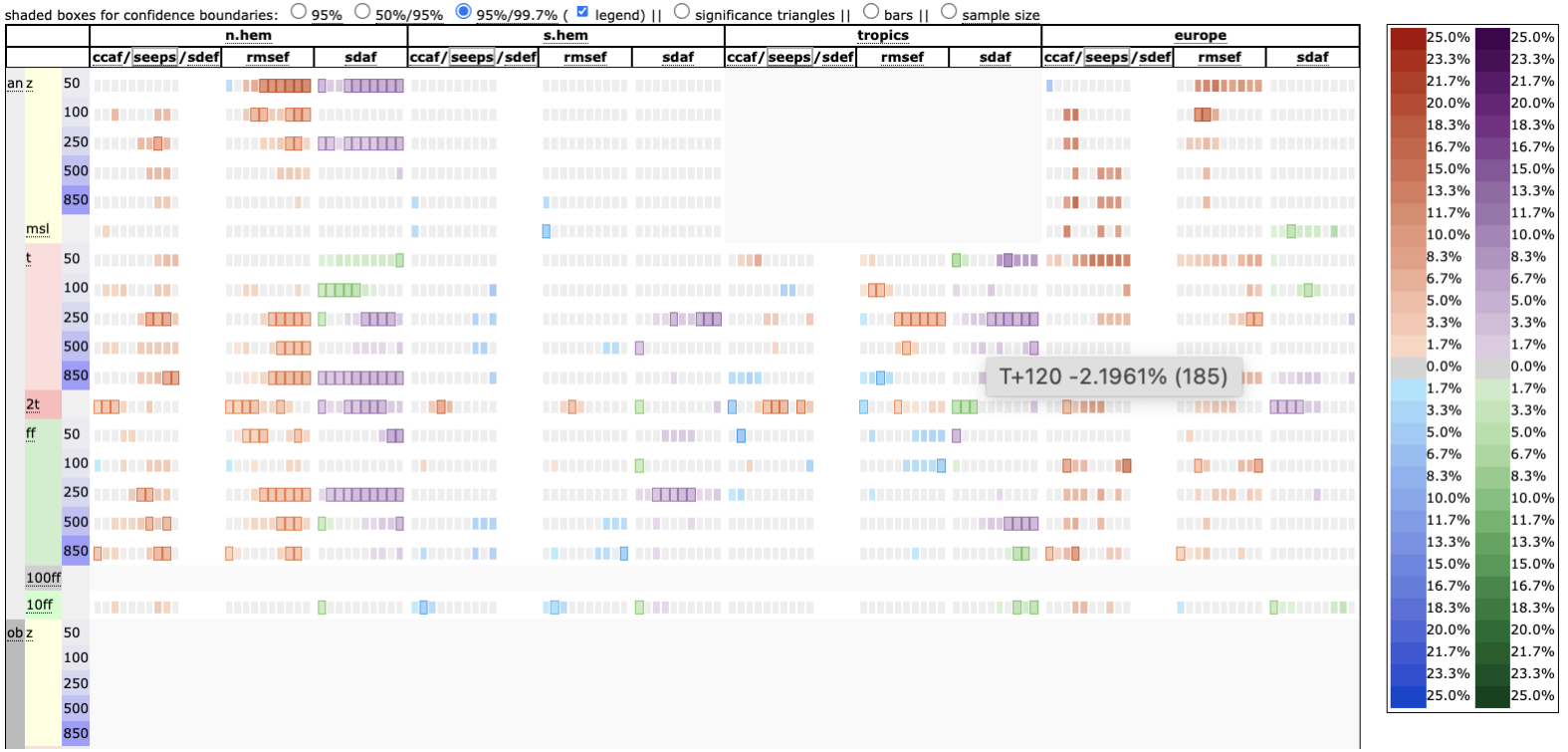
Tested alone on vN: only the imputation and wave variables differ from the baseline.

Is the release fit for purpose? - Reproducibility

Same model configuration, different anemoi version, different machine: is the difference real, or just noise?

Run-to-run variation

e.g. GPU non-determinism, seeds



Is the release fit for purpose? - Reproducibility

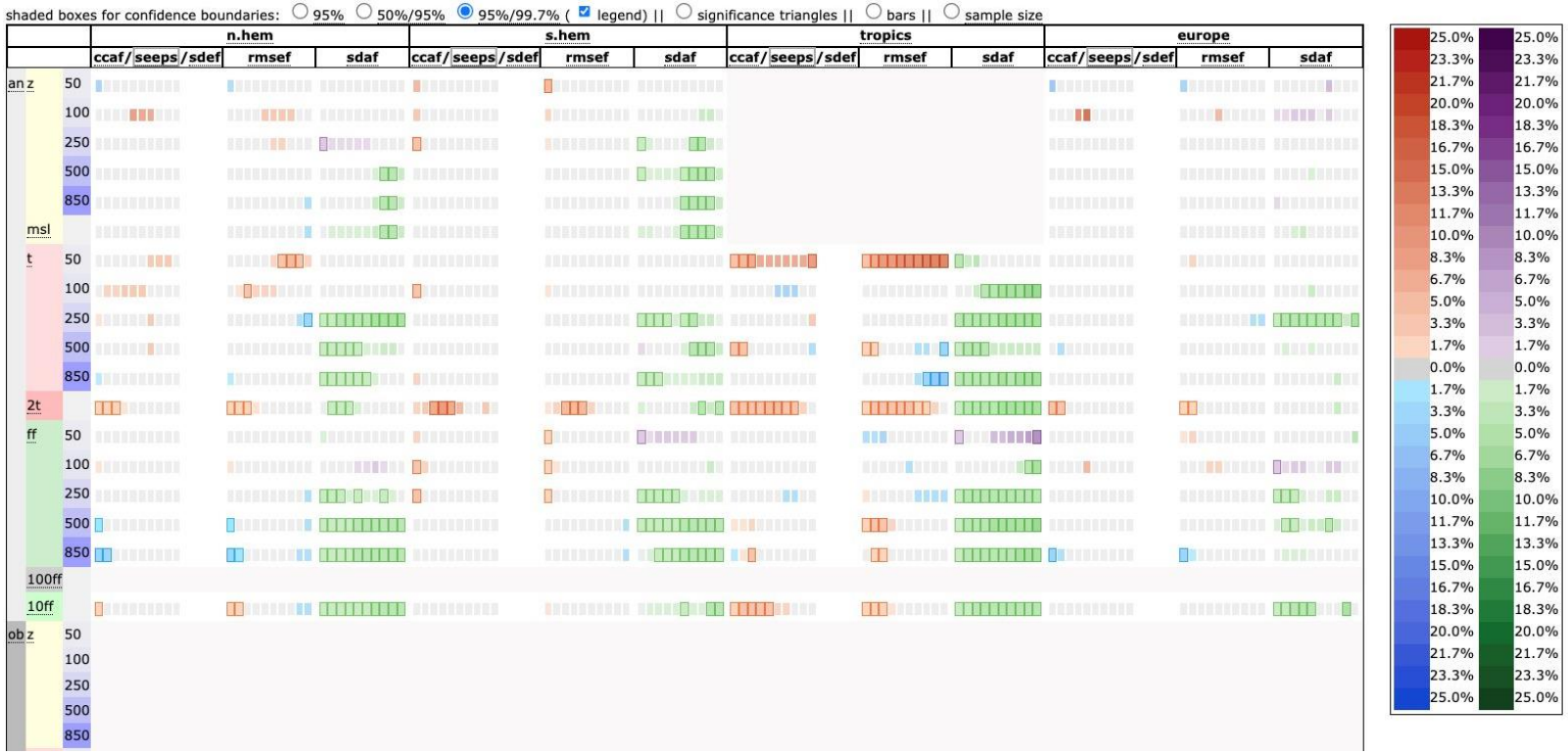
Same configurations, different anemoi version, different machine: is the difference real, or just noise?

Run-to-run variation

e.g. GPU non-determinism, seeds

Different anemoi versions

Refactors and new features can cause changes to default behaviour and breaking config changes



Is the release fit for purpose? - Reproducibility

Same configurations, different anemoi version, different machine: is the difference real, or just noise?

Run-to-run variation

e.g. GPU non-determinism, seeds

Different anemoi versions

Refactors and new features can cause changes to default behaviour and breaking config changes

How do we determine what is noise and what is a real issue?

How can we guard against this?

- Accuracy tests in anemoi now implemented to try and catch when a PR unexpectedly changes scores
- Training models with a fixed seed to limit run-to-run variation
- ….

Evaluating one contribution



WHAT MAKES THE CASE

Scorecards
against the same baseline

Bias reduction
maps and zonal means

Skill in new variables
against observations or analyses

Case studies
events experts recognise

WAVES THREAD
Skill in the new wave variables, without hurting the atmosphere.

Choosing what goes in



Example: land forcings

Urban heat islands are not well represented in prior versions of the AIFS. Introducing a urban forcing allows the model to better represent them and supports communities that are poorly served by forecasts today.

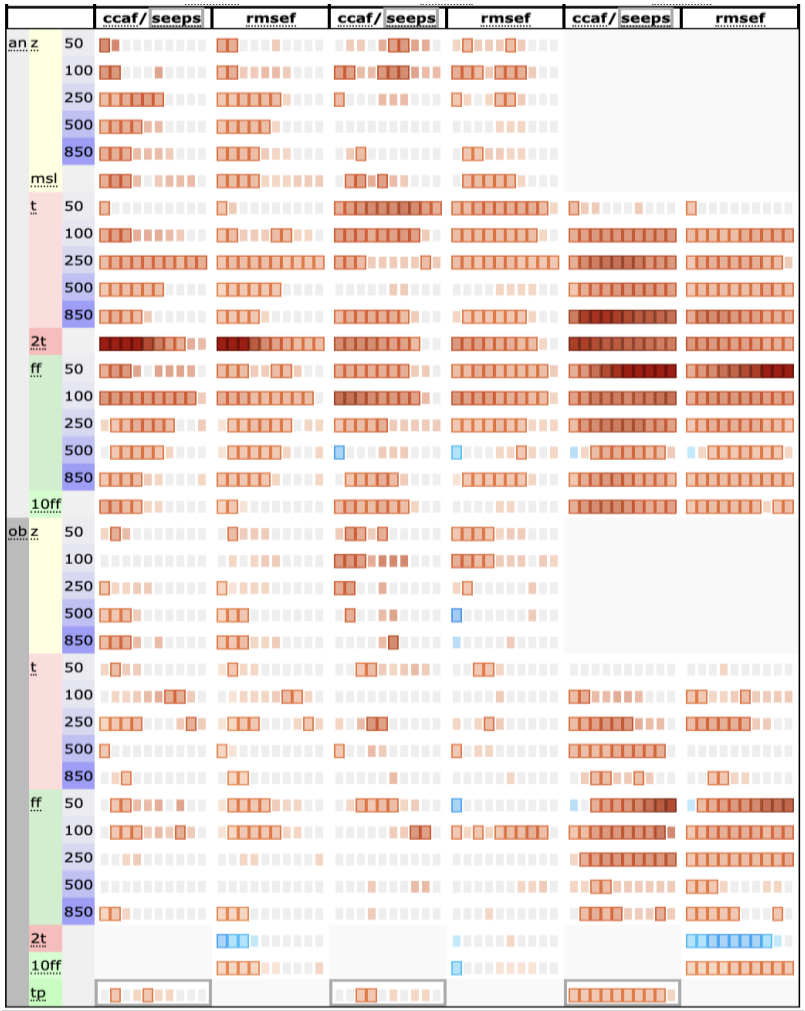
Ask early:

whose need does this meet?

When upstream moves: IFS cycle 50r1

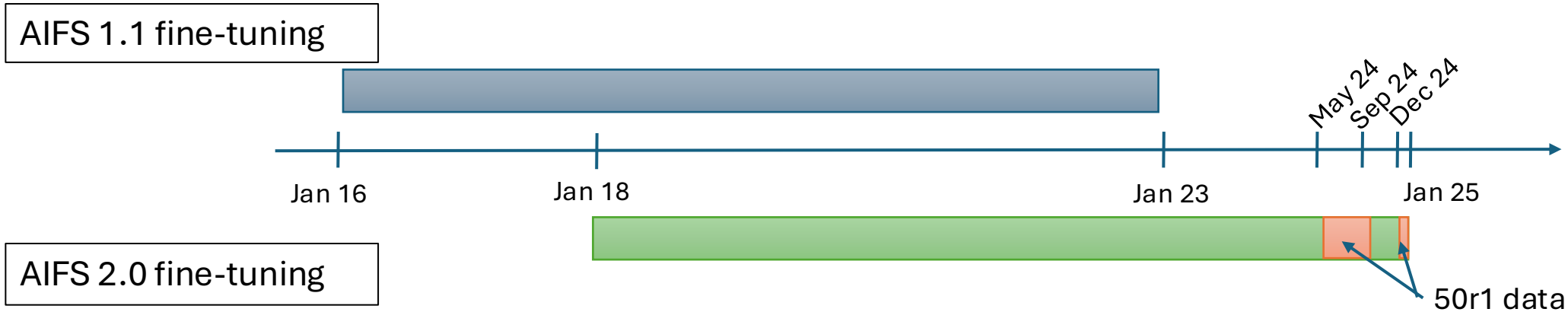
The AIFS is downstream of the IFS analysis. A new IFS cycle changed the distribution of the analysis particularly in the sea ice and led to degradations in AIFS

Comparing AIFS Single 1.1 from 50r1 vs from oper (49r1) - DJF 24/25



Fine-Tuning including data from 50r1 IFS experimental data

- Move 7 years of data for rollout/fine-tuning on operational data towards December 2024
- Substitute when available with experimental 50r1 data



Impact of fine-tuning: AIFS v2 vs AIFS v1 both initialised on 50r1



Good alone, not always good together



Group like changes

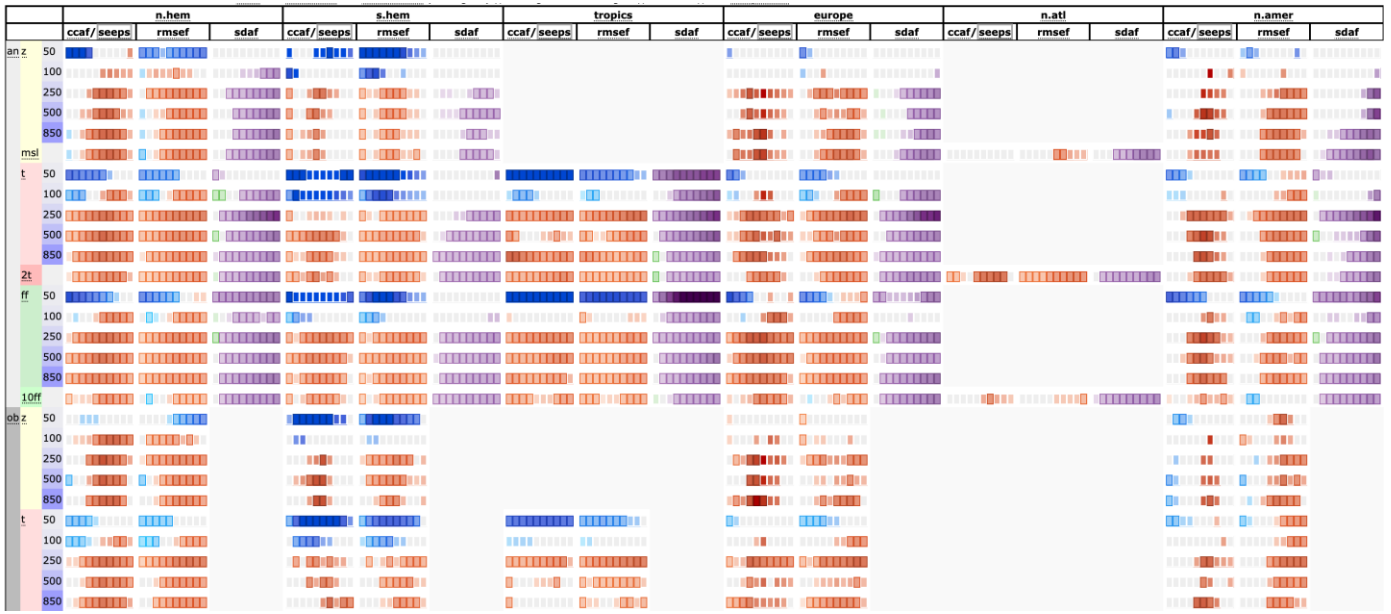
Changes that touch the same part of the model are tested together.

Re-run the full evaluation

on the combination, not just the parts.

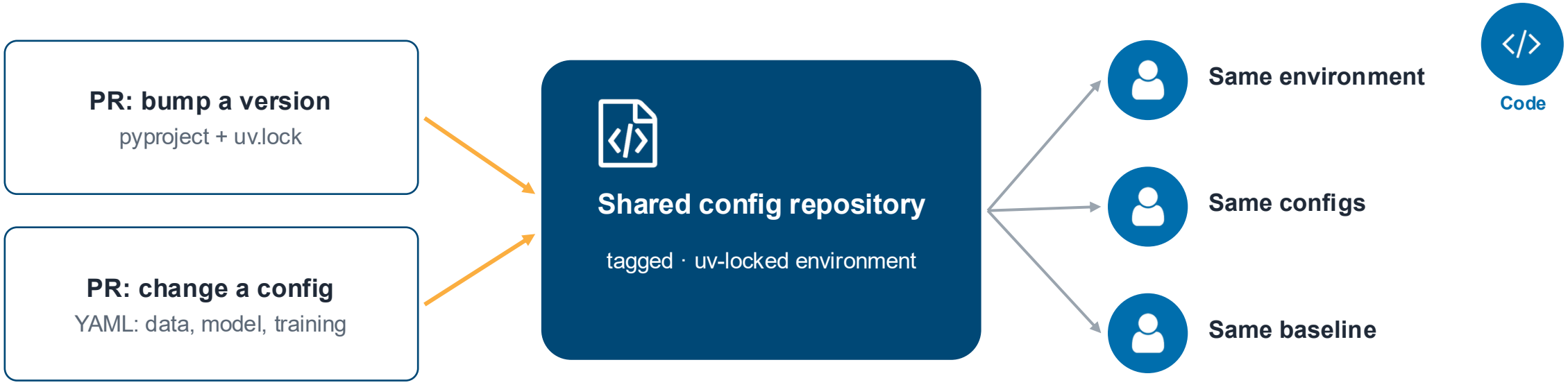
If they clash, separate

and find out which interaction caused it.



Note: AIFS Single v2 trained 20+ variants

Shared configs, shared environments



Result: people can pick up each other's work without rebuilding it.

WAVES THREAD

One PR: bump anemoi-core, switch on imputation and add the wave variables in the config.

Cheap to run, expensive to train

TRAINING AN AIFS ENSEMBLE

≈ 10,250

GPU-hours: 64 GPUs (16 nodes × 4) for one week

RUNNING ONE FORECAST MEMBER

2 minutes

on a single GPU

One training run costs about as much as 300,000 forecasts.

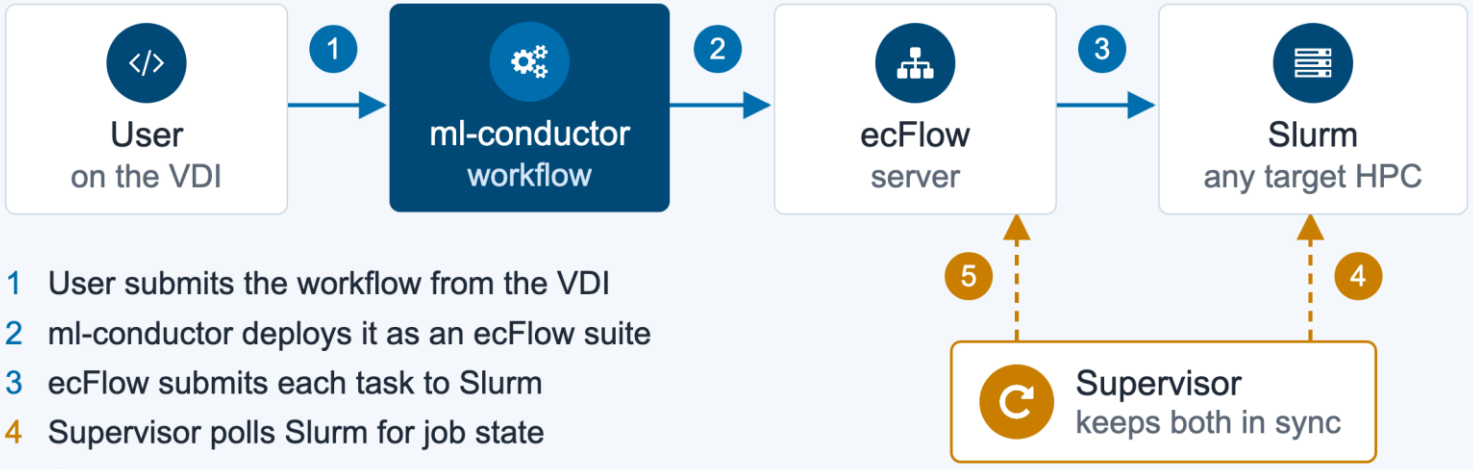
Orchestrating training across EuroHPC

Chain jobs
across walltime limits

Resubmit
automatically on failure

Span machines
across EuroHPC systems

How ml-conductor drives a run



1 User submits the workflow from the VDI
2 ml-conductor deploys it as an ecFlow suite
3 ecFlow submits each task to Slurm
4 Supervisor polls Slurm for job state
5 Supervisor updates ecFlow

EuroHPC Joint Undertaking

Same workflow, four different systems

JUPITER
NVIDIA GH200
Slurm: 12 h walltime

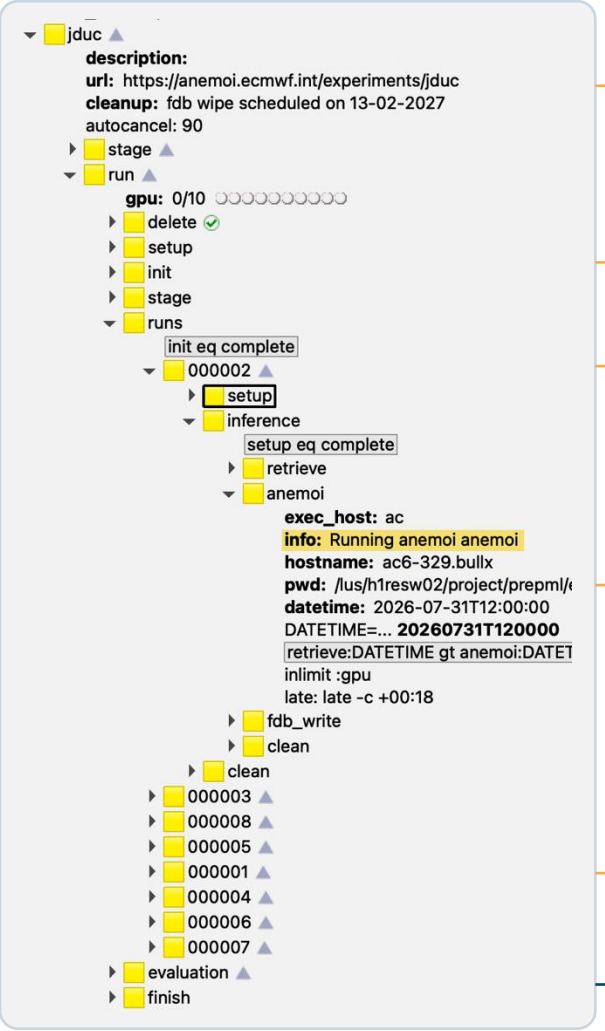
Leonardo
NVIDIA A100
Security: two-factor

MareNostrum 5
NVIDIA H100
Network: air-gapped

ECMWF HPC
NVIDIA GH200 and A100
Home: ecFlow and VDI

Evaluating the candidate

INSIDE ONE ecFLOW SUITE



Registered experiment
catalogue page and a scheduled FDB clean-up

delete · setup · init · stage
clear old scores, get checkpoint, stage input data from tape

runs: one family per member

inference: loops over start dates
`retrieve` - get initial conditions
`anemoi -` run inference
`fdb_write` - write to the FDB for eval tool read

000001 … 000008
the other members run the same loop in parallel

EVALUATION, IN THE SAME SUITE

1. **`compute`**
   Quaver scores each start date against IFS analysis once all members have archived it, on parallel workers.

2. **`plot_quaver`**
   Once every date is scored, one task draws the plots into a single quaver.pdf.

3. **`finish`**
   The plots go to the experiment's Anemoi catalogue page, next to its checkpoint and metadata.

Same shared scripts and environment for every candidate. The config sets the depth: basic, advanced or scorecard.

Bringing in the experts

Anyone with relevant expertise reviews the candidate

Scores show whether it is better on average; experts see whether it behaves like the real atmosphere.

WHO LOOKS, AND AT WHAT

Tropical cyclones
Tracks and intensity against observed best tracks.

Precipitation
Biases against station data, including the tropical bias we set out to fix.

Extremes
Do heat, wind and heavy rain stay intense, or get smoothed away?

Waves
Wave height and period against altimeter data.

Forecasters
Case studies of high-impact events they know well.

Diagnostics
Spectra and smoothness: is fine detail lost at longer lead times?

WHEN AN EXPERT FINDS SOMETHING

1. **Spot it**
   an expert flags an issue in their area
2. **Trace it**
   each change was tested alone, so the finding points to one contribution
3. **Fix or drop it**
   that change is fixed, or it leaves the combination
4. **Re-evaluate**
   the new combination goes through the full evaluation again

WAVES THREAD
We had initialised from the wrong wave IC's, which caused a subtle difference, an expert identified it

Part 2 · Putting it into operations

From a chosen candidate to a forecast that arrives on time, every time

From candidate to operations



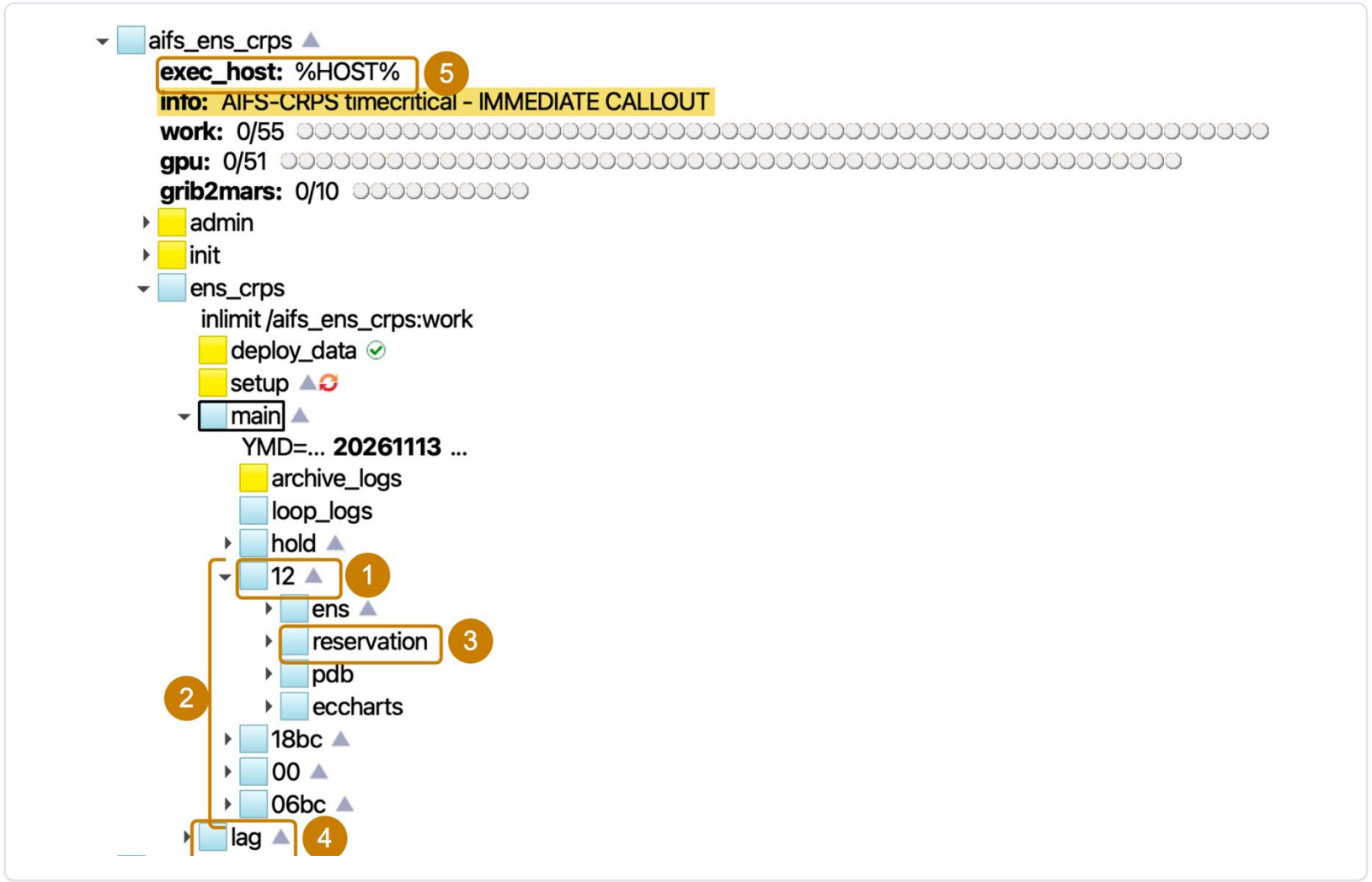
· working across departments so context is not lost at each hand-over

Research	Freezes the candidate and provides the trained checkpoint	Checkpoint
Development	Packages the checkpoint and its Python environment into a module	AIFS module
Production	Updates the operational software stack and workflows, and tests integration with FDB MARS ecCharts dissemination ecFlow	Operational suite

Standardisation pays off here

The same configs and environment used in research are what production receives. Less to re-learn, less to break.

The operational suite



KEY POINTS

1. **Triggered by the IFS**
   Starts as soon as the IFS analysis is computed

2. **One family per run**
   00, 06, 12 and 18 run as separate families

3. **Reserved for operations**
   Dedicated Slurm reservation for each run

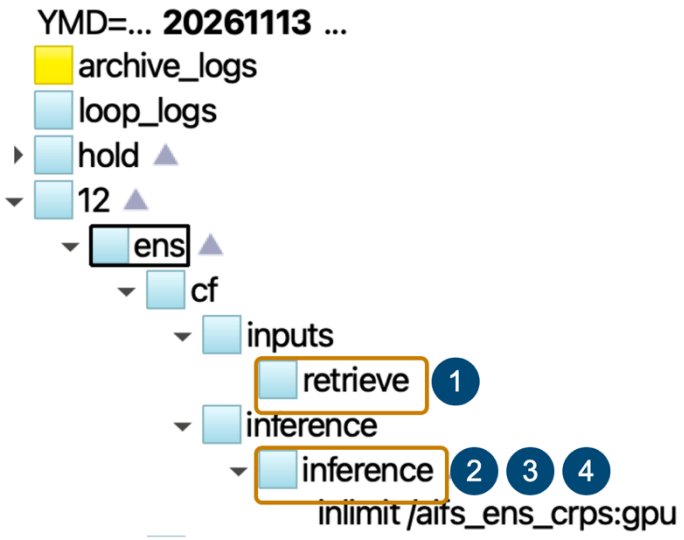
4. **Archival off the critical path**
   MARS archiving runs in the lag family

5. **Failover**
   Filesystem ws1 ↔ ws2, GPU cluster ac ↔ ag

How the scripts tie together

1. Retrieve ICs
2. Check inputs
3. Run inference
4. Encode GRIB2
5. Archive & disseminate

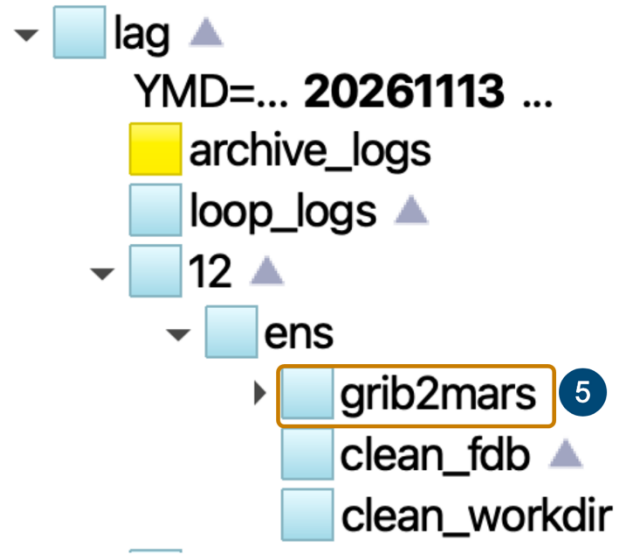
main / 12 / ens / cf



ecFlow task
retrieve
step 1

ecFlow task
inference
steps 2–4, in one GPU job

lag / 12 / ens



Packaging

AIFS module vN+1

Environment
uv-locked Python stack

Checkpoint
the trained weights

IC retrieval
scripts to fetch inputs

Run scripts
config and entry points

Built as a standalone module on the HPC, versioned alongside the AIFS

Continuous integration and delivery

Build
The module is built on each ECMWF HPC cluster

Test
Every command and artefact is tested automatically

Install
Installed and set as the default version

Roll back
Previous versions stay installed: reset the default

Manual installation remains possible if CI systems are unavailable

AIFS is Anemoi plus a configuration

Anemoi
the framework

+

Configuration
data · model · training

+

Checkpoint
trained weights

=

AIFS
the operational model

Making sure it runs as it was trained

- Same variables, from the same source
- Same units and grids
- Same initial conditions as in training

Waves: the IC source changed

Training	wave fields from the coupled IFS
Operations	wave fields from a standalone wave stream

Common challenges in operations

Failover AC ↔ AG

Two Atos complexes with different architectures: x86 and arm64. Anemoi and its environment must run on both.

Initial-condition sources

Operations may not get inputs from where training did. Check every field's source.

Encoding

GRIB2 templates, paramIds and the C++ stack must produce what users expect.

Delays

Upstream IFS delays, GPU queues and dissemination deadlines all eat into the time budget.

Takeaways: what to think about early

USERS

Engage with users

Understand their needs before you build. A contribution that meets no need will not be chosen.

DATA

Training data ≠ operational data

Check early how each input will be available in operations: source, timing, units.
What are the constraints of a realtime system?

CODE

Make your science useful with shared code

Build on main and the shared configs, so others can reproduce and operate your work.

NaN fix PR → Release → Isolated test → Scorecard → Prioritised → Trained → Evaluated → IC source fixed → **Operational**

Questions?

