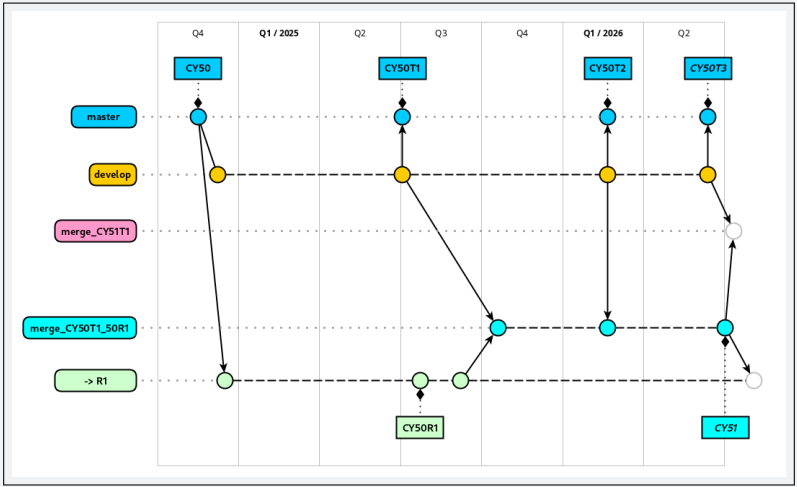
- Cycles and Integration -

A.Mary

April 13th, 2026

Latest and current cycles



CY50T1

- `https://github.com/ACCORD-NWP/IAL/releases/tag/CY50T1`
  - Rephasings of **MF e-suite** `49T1_op1` devs
  - **Observations** : MWTS3, WIGOS, CRIS FSR, STRATEOLE2, CALISOL10M, direct radar assimilation, Mode-S AW, AMV errors & QC, OceanSat-3 OSCAT, snow cover
  - **DA algo** : Arome 4DEnVar, hydrometeors
  - **Dynamics** : `LNHHY/LNHQE/LNHEE & LSPNHX, LSI_UPDATE_FULL/CHEAP, NTPVAR, LBIGW, LSLONDEM_SPLIT`, cleanings
  - **EPS** : SPG, RPP, SPP update
  - **Physics & diags** : PHYEX update, Harmonie-Arome phasing, θ_s, CIN, Arpege/Tiedtke, DDH@tstep, aero diags
  - **System** : porting/benchmark fixes, optimizations, Cmake compilation, various cleanings
  - **Refactoring** : SI, SL, LAM/NH GP dynamics, extension of `field_api` encapsulation, YDMODEL quasi-constant
  - and miscellaneous fixes...

CY50T2

- Externalisation of FA/LFI/LFA
- Externalisation of PHYEX
- Externalisation of SURFEX
- Internalisation of DAVAI (in directory `davai/`)
- Internalisation of the **bundle** and integration of cmake/ecbuild compilation for LAM (in directory `bundle/`, cf. doc therein)
- Migration of `POSNAM*` to `fiat` and use of `POSNAMEF` to make empty blocks optional (env. var)
- Miscellaneous corrections

Bundled Contributions

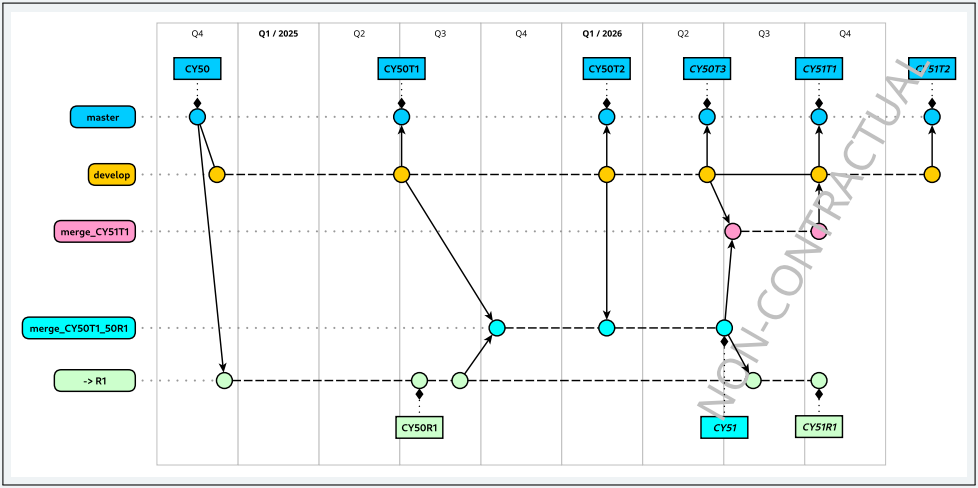
Example : SURFEX

- Developments in a branch of `https://github.com/ACCORD-NWP/SURFEX` → PR
- Merged in a staging branch for the next CY release (`50T2_develop`)
- The staging branch is bundled in an IAL PR to the `develop` branch (`bundle/bundle.yml`)
- Potential companion IAL PRs for interface changes

CY51

- Initial merge of CY50T1 + CY50R1 (fall 2025)
- + Merge of CY50T2 (March 2026)
- Synchronization on external packages (ecTrans, oops, ...)
- Validation work just resumed after a few months on hold
- Pending : reintroduction of FFT992 in ecTrans
- Target date for declaration : Sept. 2026

From CY51 onwards : continued cycling ?

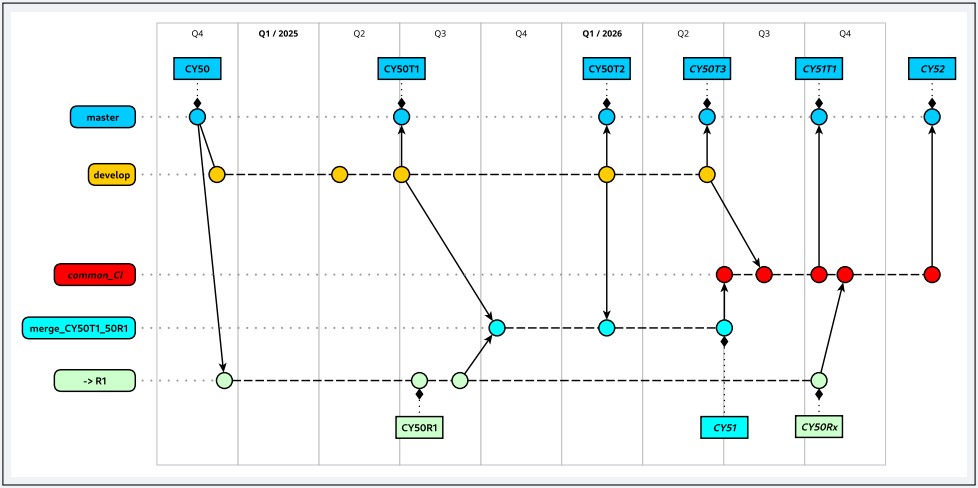


The issue with common cycles

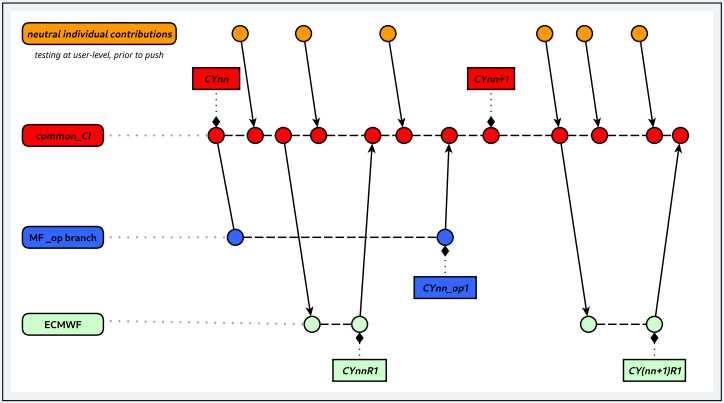
A long-standing, proven-to-work practice (over 30 years of IFS-Arpege collaboration !), but :

- Difficult merges
- Difficult validation
- A bottleneck
- Prevents agile collaboration with ECMWF, esp. on code modernization

From CY51 onwards : or a more linear and continuous process ?



From CY51 onwards : a Common Continuous Integration (CCI) ?



CCI : what would it look like ?

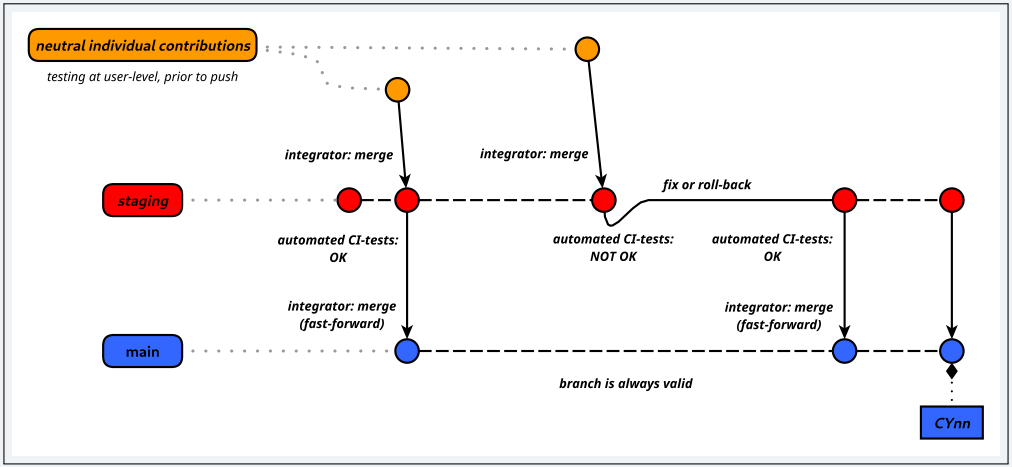
- **Neutral** [a] contributions to be submitted and merged *"continuously"* (i.e. not as part of a call for contributions to a cycle)
  ↪ based on latest release or up-to-date main branch
  ↪ into a "staging" branch
  ↪ more visibility and cross-review
- Contributions with **meteorological impact** [b] (or groups of) cannot follow the same path (need for further statistical validation and case-studies → longer timescale)
  ↪ thorough evaluation done in side-branches before re-entering the main branch
- Declaration of new **releases** on main branch is flexible :
  ↪ after integration of meteorological-impact branches, or upon demand
  ↪ less bound to the notion of *"cycling"*, more linear : semantic versioning ?

a. seeafter
b. less than 1/3 of PRs

CCI : Neutrality of contributions ?

- bit-identical (for which compilers ?) ?
- bit-reproducible (i.e. whatever the // profile) ?
- non-repro but no meteorological impact ?

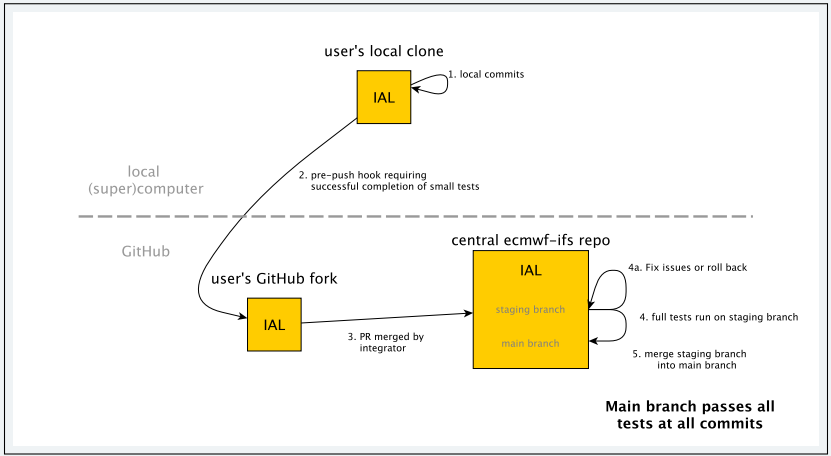
CCI : testing


neutral individual contributions
testing at user-level, prior to push
integrator: merge
integrator: merge
staging
fix or roll-back
automated CI-tests: OK
automated CI-tests: NOT OK
automated CI-tests: OK
integrator: merge (fast-forward)
integrator: merge (fast-forward)
main
branch is always valid
CYnn

CCI : testing

- Testing at user-level : contributions need to be tested locally **before** dropping a PR
  - machine-specific test suites
  - ifs-tests + davai
- Testing at CI-level : automated on **pushes to the staging branch** on github
  - using a GitHub workflow (automation)
  - triggering tests on several machines
  - triggered by integrators only

CCI : testing


user's local clone
IAL
1. local commits
local
(super)computer
2. pre-push hook requiring
successful completion of small tests
GitHub
central ecmwf-ifs repo
IAL
4a. Fix issues or roll back
user's GitHub fork
staging branch
4. full tests run on staging branch
IAL
3. PR merged by
integrator
main branch
5. merge staging branch
into main branch
Main branch passes all
tests at all commits

Comments, questions ?

PS : `git push != Ctrl-S`