Testing overview

Why do we want to do this?

We're building an *infrastructure*.

- long-term sustainability and maintainability
- many people involved, many components involved

➡️ Need (automated) processes to see breakage *before* it happens.

➡️ Need good practices to reduce error rate.

Previous work

- HBP/EBRAINS Software Quality Guidelines “Best Practices”
  - fill the SQ-Checklist.pdf → passing, silver, gold ❓
  - https://gitlab.ebrains.eu/esq-wg/sq-guidelines ❗
- Handbook: Software Deployment and Delivery
- ESD meetings for EBRAINS Software (science-to-esd)
- ESD-on-HPC meetings for HPC deployments (esd-to-machine)
- Merge-request template: ☑ Software Follows SQ-Guidelines…
- ESD Hackathon: Test Matrix slides

What / How / When

What do we want to test?

- Scientific model
- Tool
- HPC site
- Services
- Workflow, Workshop materials, …

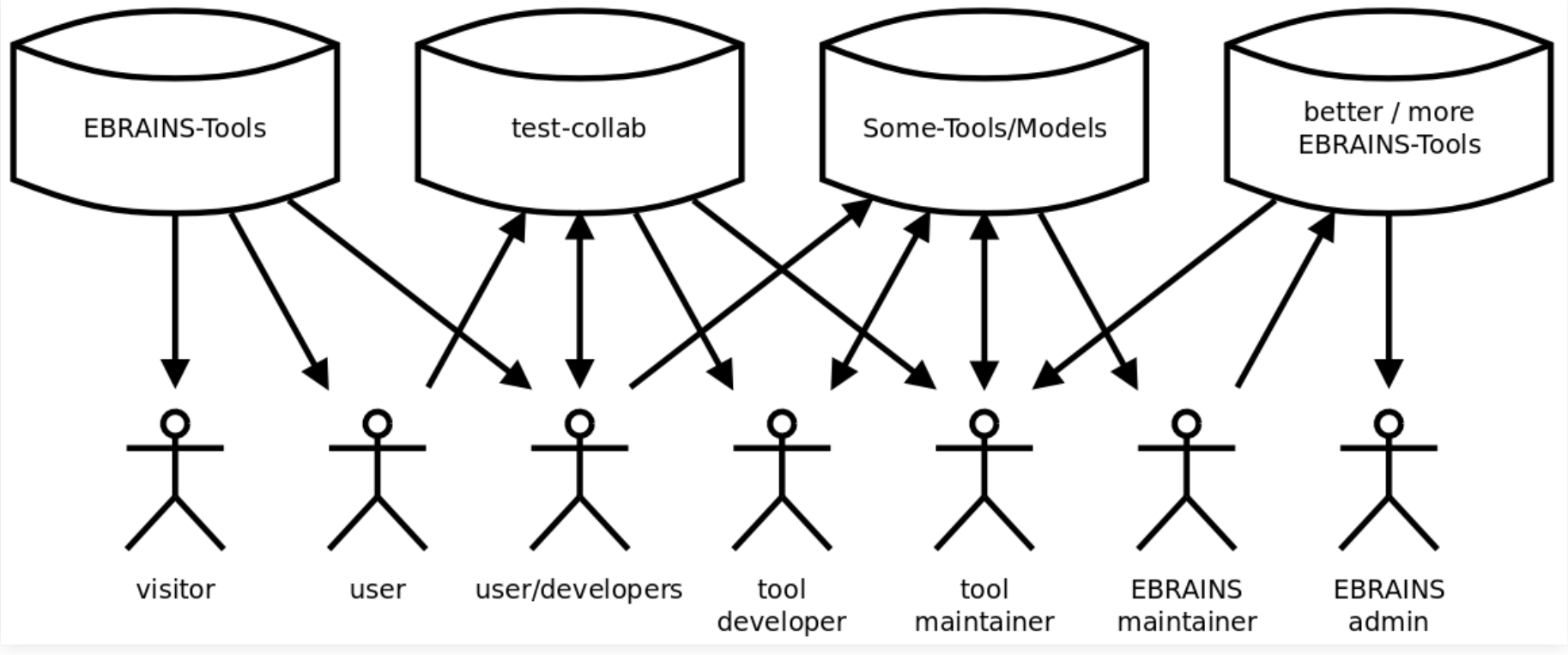
How to test

- Static checks
- Unit tests
- Functional / regression tests
- Benchmark tests
- Local integration tests
- Packaging & deployment
- Framework integration
- “Full-stack” testing

What to consider

- Result
- Process Location
- Underlying Resources
- Environment (local, CI, HPC, …)
- Process / Framework(s)
- Development Level
    - functionality
    - integration
    - interoperability
    - service & operation
    - validation

Guiding idea



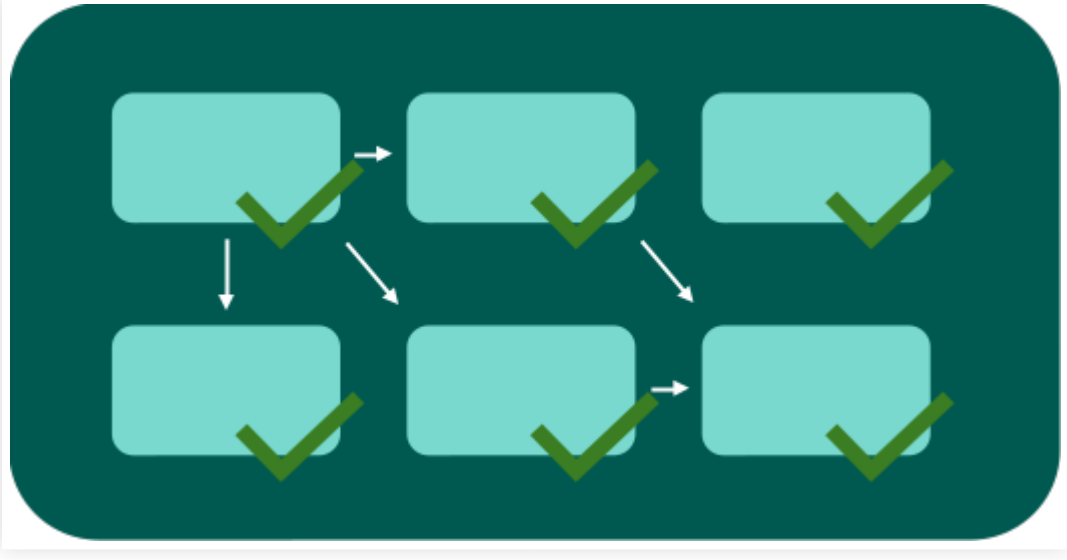
💡 Test where the knowledge is.

Status & Goals

What we have in place

Type	Environment
Static checks	Local
Unit tests	local
Functional / regression tests	local
Benchmark tests	local/HPC
Local integration tests	local+CI
Packaging & deployment	CI / CI+HPC
Framework integration	CI+`int`
“Full-stack” testing	(CI+x) 

Per-tool post-installation tests



Purpose: ensure every tool is functional and correctly installed across CI, Lab, HPC etc.

- defined for each ESD package inside Spack recipe
- executed **automatically** on every package **update or rebuild**
- typically a lightweight version of the tool's own test suite
    - no heavy data, long runtimes, large resource usage

➡️ **Unit tests**

validate individual modules of each tool

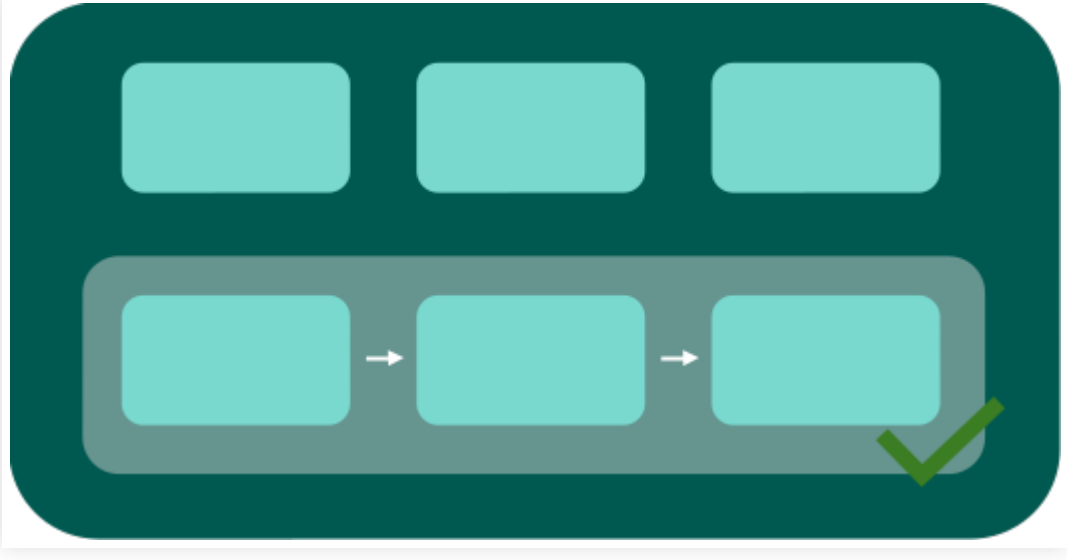
➡️ **Functional / regression tests**

ensure tools behave correctly as features evolve

➡️ **Packaging & deployment tests**

ensure proper installation on any target systems

Workflow tests



Purpose: validate interoperability between tools across the ESD ecosystem

Current coverage

- represent realistic multi-tool usage
- executed **automatically** on every workflow component **update or rebuild**
- provide a way to introduce necessary, but not EBRAINS-developed (external) tools in the ESD
- ensure that EBRAINS workflows/showcases can be executed out-of-the-box anywhere the ESD is deployed (EBRAINS Lab, HPC systems)

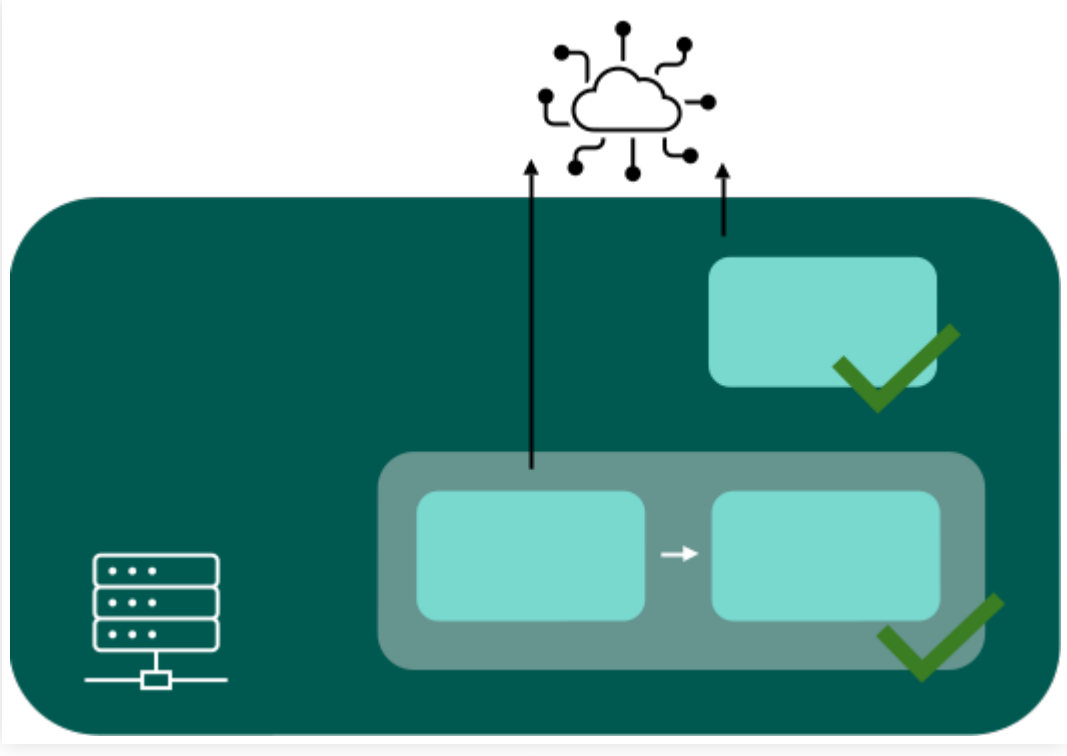
➡️ **Local integration tests**

validate interoperability: dependency compatibility, API consistency, cross-tool data formats

➡️ **Functional / regression tests (multi-tool context)**

ensure workflows behave correctly as tools evolve

Stand-alone tests



Purpose: testing interactions with infrastructure beyond the ESD

- stand-alone tests that run independently of package updates/ deployments
- executed in a scheduled manner on each environment
- catching issues caused by external system changes: services, underlying system changes
- can be executed on CI, on HPC, on production systems (e.g., Lab)

➡️ **Framework integration tests**

validate integration with external services and infrastructure

➡️ **Full-stack testing**

end-to-end execution of realistic multi-tool workflows in real deployment environments

What is missing

- multi-site workflows
  - HPC connection from Jupyter and CI, pyUnicore, …
- multi-site benchmarks
  - `bm-` packages, EuroHPC quota & procedure
- Dummy service containers in tests
  - service API tests, Lab-2-Service compatibility
- Scientific validation
  - comparison of test results with expected values
- Test execution on HPC from CI
  - authentication mechanism: Jacamar, UNICORE, …
  - resource management, frequency
- Different test suites for each package
  - smoke tests, benchmark tests, validation tests, full tests

 *LET's START!* 

Thank you.

EBRAINS ESD Hackathon

Athens 2025/11

Other links…

- These slides (src)
- Test Matrix slides from previous ESD Hackathon (Heidelberg, 2024/11)
- ESD slide template

tags: `slides` `hackathon`