EBRAINS

Co-funded by the European Union

Testing the ESD
"Unit" Testing - EBRAINS tools

1st ESD Hackathon

Eleni Mathioulaki (on behalf of the ESD team)

EBRAINS 2.0 has received funding from the European Union's Research and Innovation Program Horizon Europe under Grant Agreement No. 101147319.

ESD testing - Motivation

- **Reliability**
  - guarantee that tools function as expected
- **Consistency**
  - ensure updates or changes do not introduce conflicts/instability
- **Interoperability**
  - confirm that tools and dependencies work seamlessly together in the ecosystem
- **Future-Proofing**
  - identify and address issues proactively, sustain the ecosystem over time
- **User Confidence**
  - provide researchers with a verified, ready-to-use system that “just works.”

ESD testing

What?

- **tools**: verify functionality of individual tools, defined by tool maintainers
- **workflows**: verify integration and consistency between tools

When?

- **post-installation** tests
  - immediately after installation
  - confirm proper setup and reproducibility in each environment/deployment
- **periodic** tests
  - regular, scheduled tests
  - ensure stability and compatibility over time (including external system interactions)

ESD unit post-install tests

- validate individual tools
- **automated** in EBRAINS GitLab CI: catch issues early
- **cross-platform**: ensure tools work consistently across local, Lab, and HPC environments

Implementation:

- Spack build-time tests
- pre-defined tests per build system (e.g. python import tests, make installcheck)
- executed when `spack install --test root`
- run in the package’s build environment
- **limitation**: build environment is not always the same as runtime environment!

Co-funded by the European Union

ESD unit post-install tests

```
@run_after('install')
@on_package_attributes(run_tests=True)
def install_test(self):
    # run tests here:
    pytest = which('pytest')
    pytest()
```

```
@run_after('install')
@on_package_attributes(run_tests=True)
def check_install(self):
    ppu_gcc = which('powerpc-ppu-gcc')
    ppu_gcc('--version')
```

```
@run_after("install", when="+python")
@on_package_attributes(run_tests=True)
def install_test(self):
    python("-c", "import arbor")
```

```
@run_after('install')
@on_package_attributes(run_tests=True)
def check_install(self):
    make("test.serial")
```

```
@run_after('install')
@on_package_attributes(run_tests=True)
def install_test(self):
    python('-c', 'import neuron; neuron.test(); quit()')
```

Co-funded by the European Union

ESD workflow packages

- Spack "meta-packages", named **"wf-{workflow name}"**
- Represent **multi-tool EBRAINS workflows**
  - e.g., notebooks, scripts, multi-site/UNICORE/CWL workflows etc
- Workflow package definitions include:
  - all the **software dependencies** of the workflow (may include EBRAINS and external tools)
  - well-defined **tests** (may include validation/benchmarking aspects)
- Motivation:
  - **structured representation** of tool interdependencies
  - facilitates **deployment** of workflows
  - facilitates **testing** of workflows (incl. possible service dependencies)

Testing the ESD

post-installation tests

periodic (stand-alone) tests

individual tool tests

- test **usability** and **correct installation** of tools
- defined by developers

workflow tests

- test correct execution of **workflow** and **interoperability** of tools
- defined by workflow maintainer

scheduled tests

- additional **scheduled** execution of workflow tests
- weekly (daily if needed?)
- may also include testing of **external services, underlying systems**

triggered by any update in the package/workflow
or one of its dependencies

triggered periodically
(e.g. weekly or more often)

EBRAINS

Thank you!

Co-funded by
the European Union

EBRAINS 2.0 has received funding from the European Union's Research and Innovation Program Horizon Europe under Grant Agreement No. 101147319.