EBRAINS

Co-funded by
the European Union

EBRAINS Software Distribution

EBRAINS developer's day

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.

Ambition - a common software ecosystem

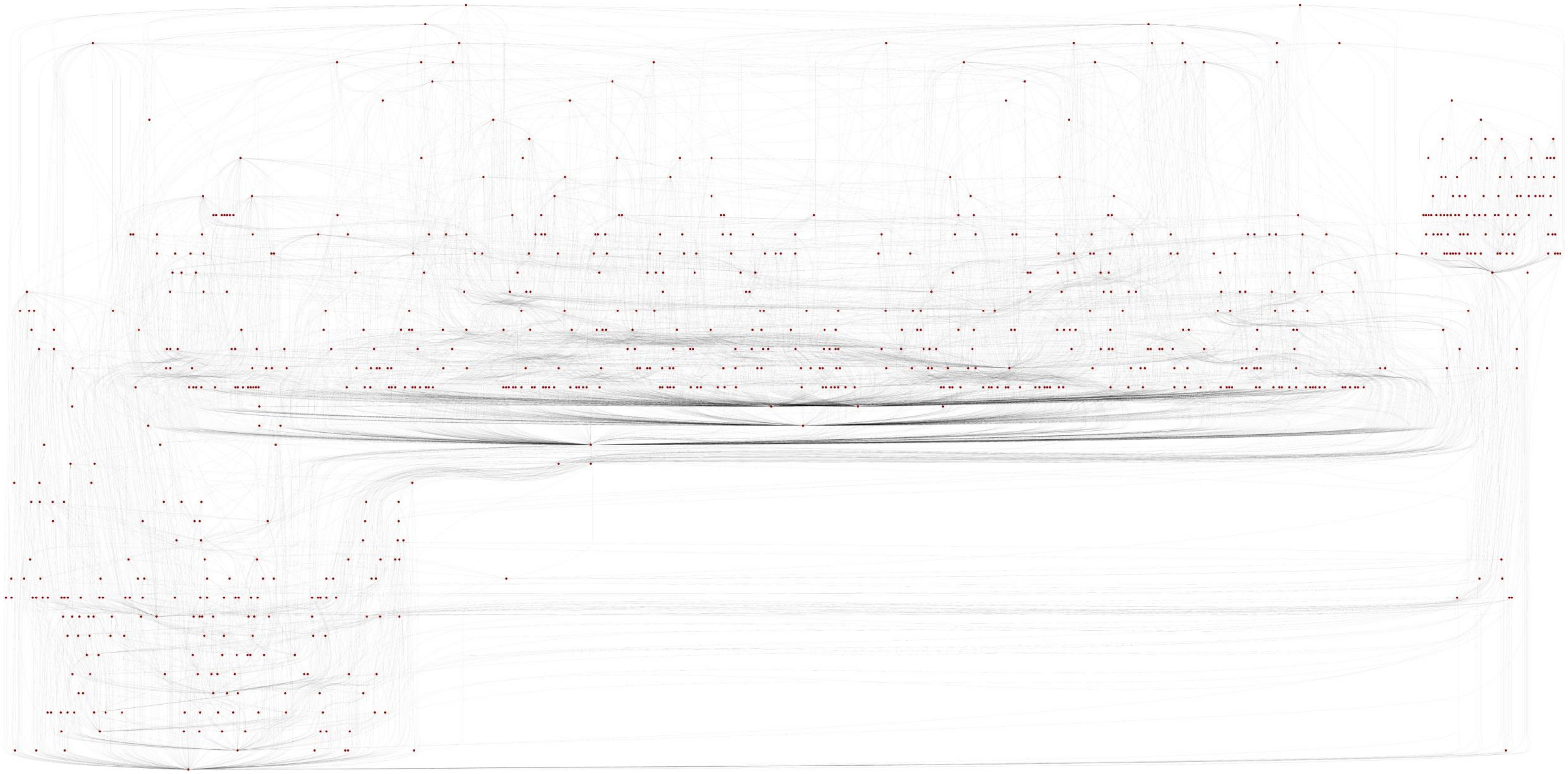
- modern scientific tools: numerous **dependencies on external libraries**
  - **code reuse** - reduces duplication, increases efficiency
  - BUT increases **complexity** of managing sw environments
  - maintaining **interoperability**: integration effort
  - updates create **constant** compatibility challenges – **ongoing effort**
  - **technical dept**
  - **non-reproducible** environments

Co-funded by the European Union

Ambition - a common software ecosystem

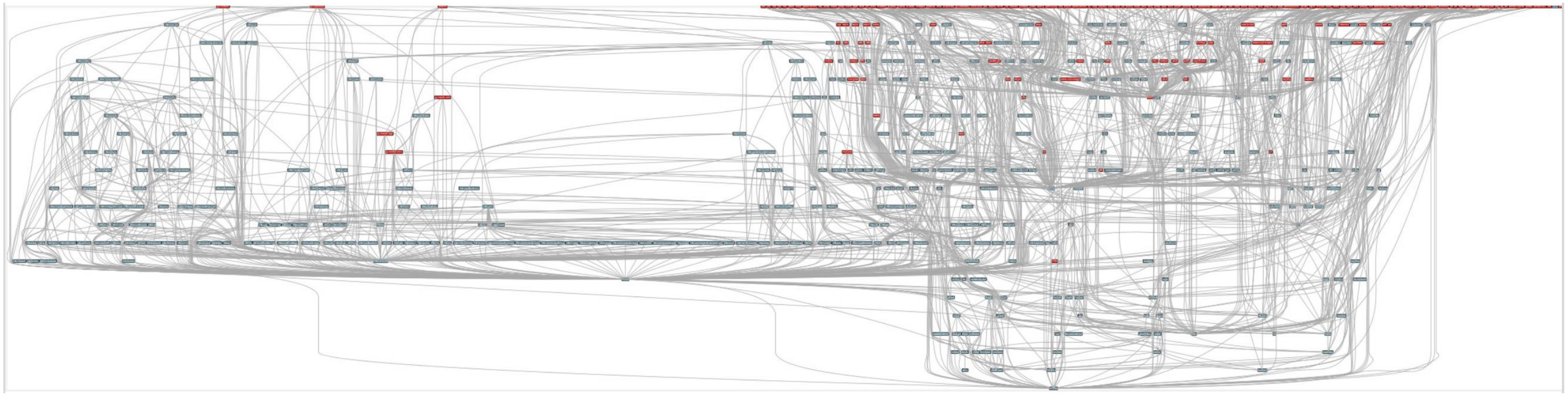
```
$ apt-get install python3-pynn
# ...
The following NEW packages will be installed:
  binutils binutils-common binutils-x86-64-linux-gnu bzip2 cpp cpp-12 file fontconfig-config fonts-dejavu-core g++ g++-12 gcc gcc-12
  ibverbs-providers javascript-common krb5-locales libabsl20220623 libaec0 libaom3 libasan8 libatomic1 libavif15 libbinutils
  libblas3 libblosc1 libboost-dev libboost1.74-dev libbrotli1 libbsd0 libc-dev-bin libc-devtools libc6-dev libcc1-0 libcrypt-dev
  libctf-nobfd0 libctf0 libcurl4 libdav1d6 libde265-0 libdeflate0 libevent-core-2.1-7 libevent-pthreads-2.1-7 libexpat1
  libexpat1-dev libfabric1 libfontconfig1 libfreetype6 libfribidi0 libgav1-1 libgcc-12-dev libgd3 libgdbm-compat4 libgdbm6
  libgfortran5 libglib2.0-0 libglib2.0-data libgomp1 libgprofng0 libgraphite2-3 libgssapi-krb5-2 libharfbuzz0b libhdf5-103-1
  libheif1 libhwloc-plugins libhwloc15 libibverbs1 libicu72 libimagequant0 libisl23 libitm1 libjansson4 libjbig0 libjpeg62-turbo
  libjs-jquery libjs-sphinxdoc libjs-underscore libk5crypto3 libkeyutils1 libkrb5-3 libkrb5support0 liblapack3 liblbfgsb0 liblcms2-2
  liblerc4 liblsan0 liblzo2-2 libmagic-mgc libmagic1 libmpc3 libmpfr6 libmunge2 libnghttp2-14 libnl-3-200 libnl-route-3-200
  libnsl-dev libnsl2 libnuma1 libopenblas-dev libopenblas-pthread-dev libopenblas0 libopenblas0-pthread libopenjp2-7 libopenmpi3
  libpciaccess0 libperl5.36 libpmix2 libpng16-16 libpsm-infinipath1 libpsm2-2 libpython3-all-dev libpython3-dev libpython3-stdlib
  libpython3.11 libpython3.11-dev libpython3.11-minimal libpython3.11-stdlib libquadmath0 libraqm0 librav1e0 librdmacm1 librtmp1
  libsnappy1v5 libssh2-1 libstdc++-12-dev libsvtav1enc1 libsz2 libtiff6 libtirpc-common libtirpc-dev libtirpc3 libtsan2 libubsan1
  libucx0 libwebp7 libwebpdemux2 libwebpmux3 libx11-6 libx11-data libx265-199 libxau6 libxcb1 libxdmcp6 libxext6 libxml2 libxnvctrl0
  libxpm4 libxsimd-dev libyuv0 linux-libc-dev mailcap manpages manpages-dev media-types mime-support netbase neuron
  ocl-icd-libopencl1 perl perl-modules-5.36 python-babel-localedata python-tables-data python3 python3-all python3-all-dev
  python3-babel python3-beniget python3-cheetah python3-decorator python3-dev python3-distutils python3-gast python3-jinja2
  python3-lazyarray python3-lib2to3 python3-markupsafe python3-minimal python3-neo python3-neuron python3-numexpr python3-numpy
  python3-olefile python3-packaging python3-pil python3-pkg-resources python3-ply python3-pynn python3-pythran python3-quantities
  python3-scipy python3-tables python3-tables-lib python3-tz python3.11 python3.11-dev python3.11-minimal rpcsvc-proto
  shared-mime-info xdg-user-dirs xz-utils zlib1g-dev
0 upgraded, 200 newly installed, 0 to remove and 0 not upgraded.
Need to get 187 MB of archives.
After this operation, 941 MB of additional disk space will be used.
Do you want to continue? [Y/n]
```

Current ESD Dependency Graph



Co-funded by the European Union

Dependency Graph



Co-funded by the European Union

The EBRAINS Software Distribution

EBRAINS

- simulator engines, data analysis and visualisation tools, client libraries of EBRAINS services
- **60+ science tools** need to be available to users
- **~800 dependencies** in total
- different target environments need **different configurations**: EBRAINS Lab, optimised installations on different HPC sites

py-neo
py-pynn
py-nestml
py-bsb
wf-biobb
neuron
py-frites
sda
py-hippounit
py-neuror
py-tvb-gdist
py-lfpy
wf-protein-association-rates
py-hbp-validation-client
wf-multi-area-model
py-tvb-multiscale
py-tvb-storage
py-hbp-archive
wf-uq-akar4
wf-brainscales2-demos
py-hbp-neuromorphic-platform
py-quantities-scidash
py-snudda
py-netpyne
r-rgsl
jaxsnn
py-libsonata
py-tvb-contrib
py-spynnaker
arbor
py-bluepyefe
py-silbra
r-sbtabvfgen
py-tvb-data
py-efel
py-quantities
py-tvb-library
py-tvb-framework
py-tvb-widgets
py-ebrains-kg-core
py-elephant
r-uqsa
py-viziphant
py-bluepymm
py-libneuroml
py-pyunicore
py-bluepyopt
py-ebrains-drive
pynn-brainscales
py-fairgraph
py-neurom
nest
hxtorch
py-lfpykit
biobb

Unified, **consistent** EBRAINS software ecosystem containing:

- all EBRAINS tools
- the optimal tree of all their (transient) **dependencies**
- **EBRAINS workflows** (software dependencies & tests)
- soon possibly services

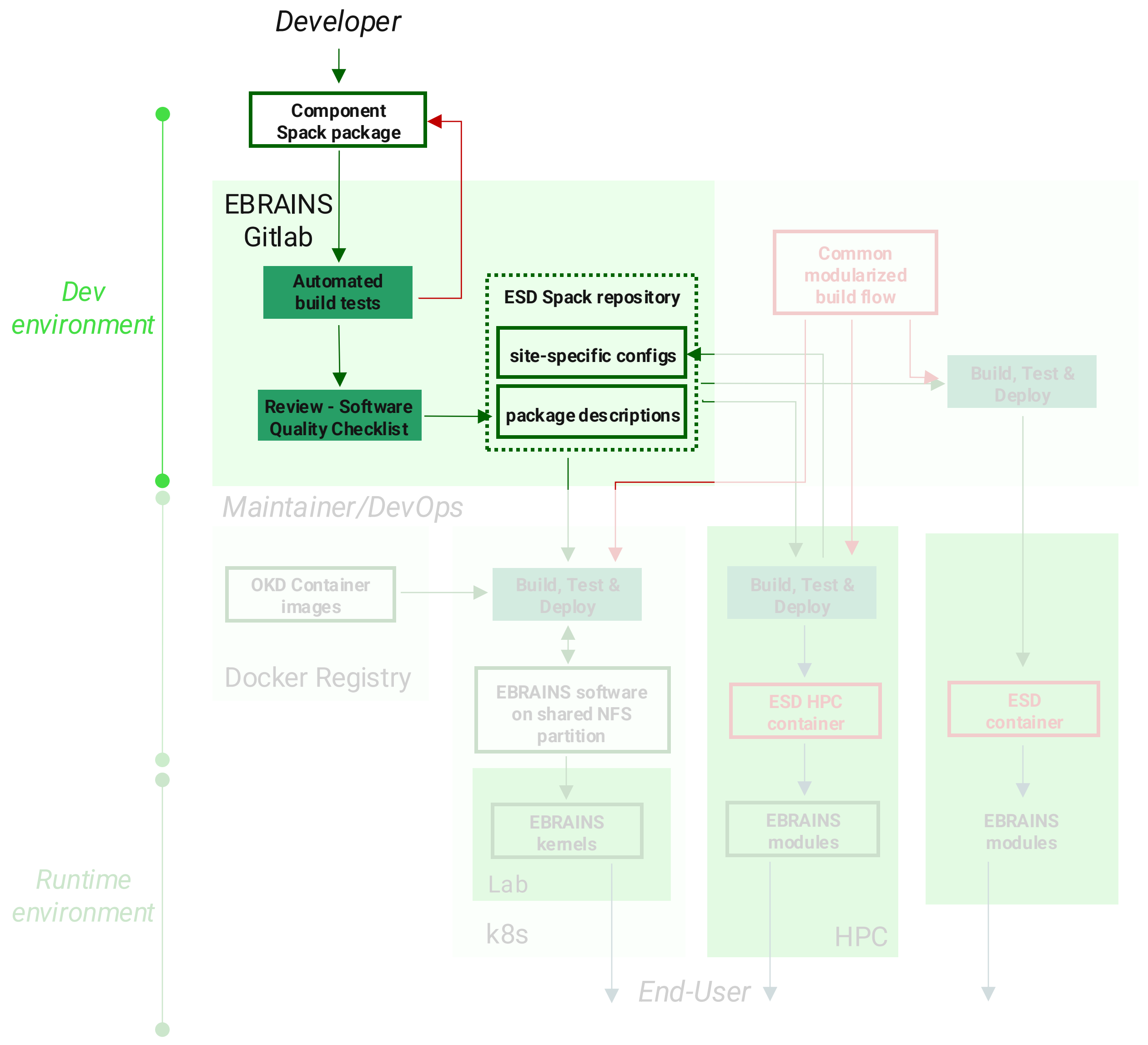
The EBRAINS Software Distribution

Goals:

- automated **dependency management**
- ensuring **consistency** (no conflicts)
- **reproducible** software environment
- tool unit/interoperability **testing**
- versioned, tested, validated **releases** on structured schedule
- **transparency** to users: EBRAINS kernels in the Lab, EBRAINS modules on HPC systems

Co-funded by the European Union

Development and Release Flow

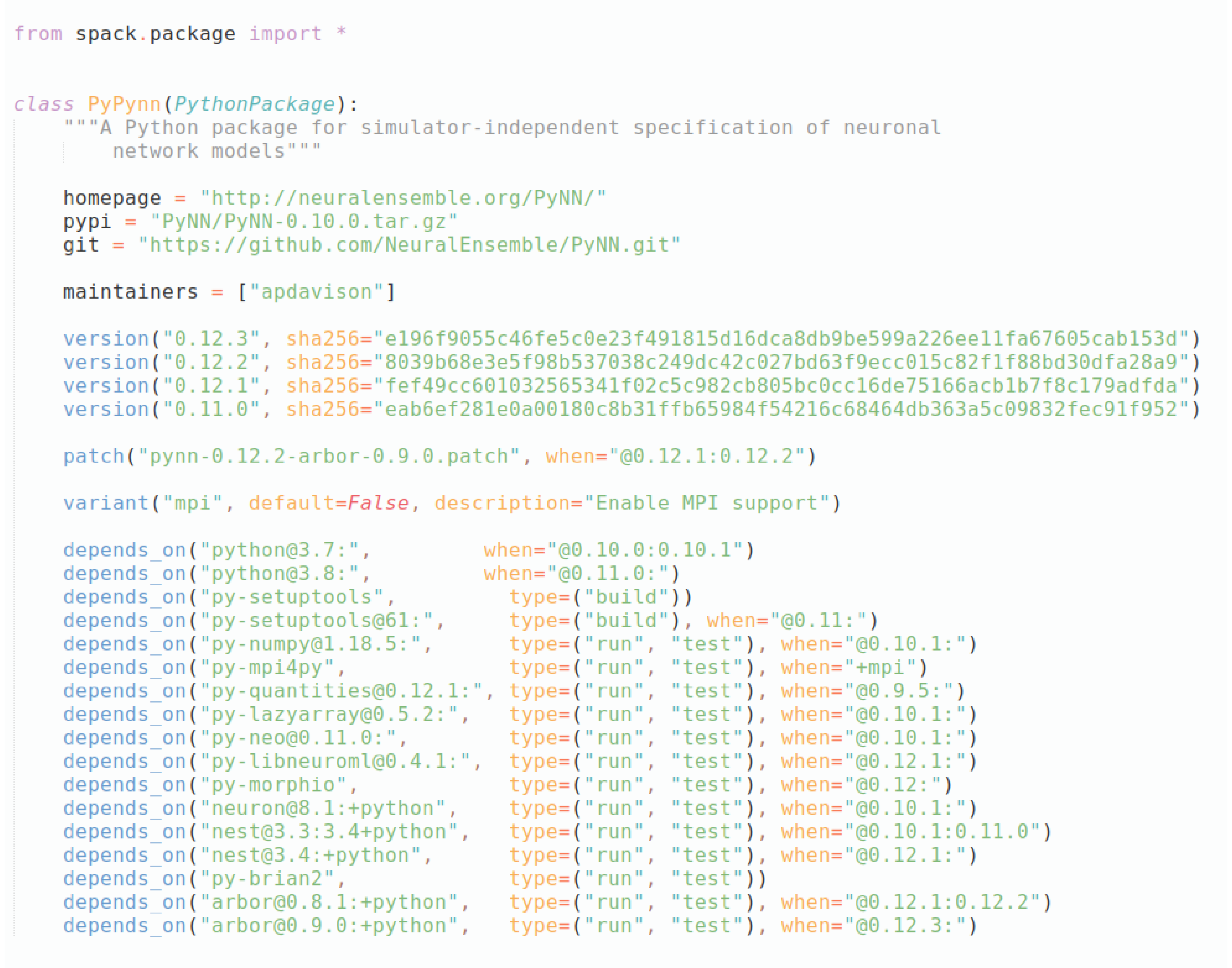


- **Official ESD repository:** https://gitlab.ebrains.eu/ri/tech-hub/platform/esd/ebrains-spack-builds
- **Spack** used to define the software stack, dependencies and build instructions: `spack create <url>`
- **MR** in official ESD repository
- **automated build tests** triggered on each commit and MR
- acceptance criteria:
  - passing **build test pipeline**
  - passing **Software Quality Checklist**

Co-funded by the European Union

Getting software into the ESD

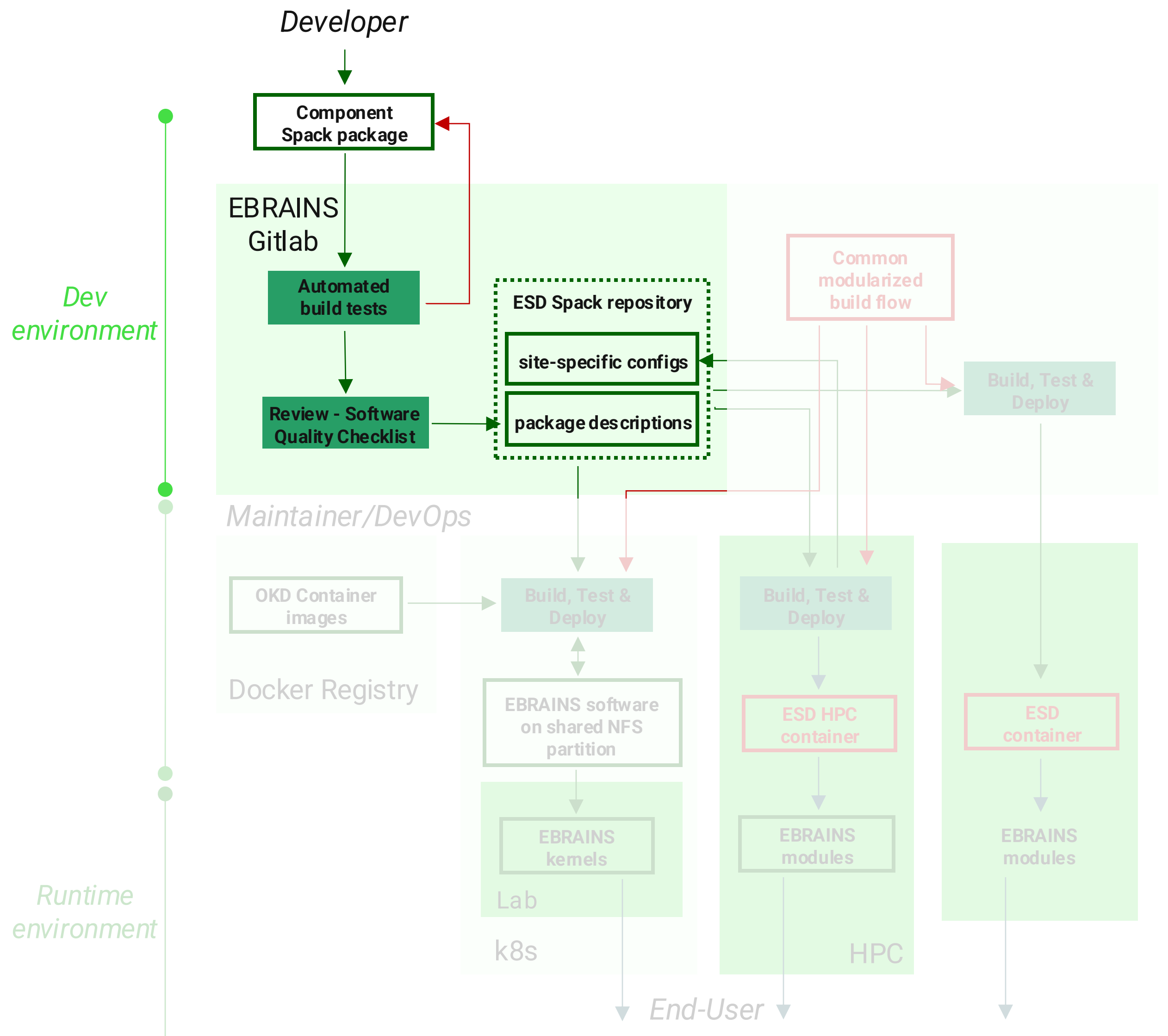
- **Spack package**
  - versions
  - versioned dependencies
  - variants
  - patches
  - build logic



```python
from spack.package import *


class PyPynn(PythonPackage):
    """A Python package for simulator-independent specification of neuronal
    network models"""

    homepage = "http://neuralensemble.org/PyNN/"
    pypi = "PyNN/PyNN-0.10.0.tar.gz"
    git = "https://github.com/NeuralEnsemble/PyNN.git"

    maintainers = ["apdavison"]

    version("0.12.3", sha256="e196f9055c46fe5c0e23f491815d16dca8db9be599a226ee11fa67605cab153d")
    version("0.12.2", sha256="8039b68e3e5f98b537038c249dc42c027bd63f9ecc015c82f1f88bd30dfa28a9")
    version("0.12.1", sha256="fef49cc601032565341f02c5c982cb805bc0cc16de75166acb1b7f8c179adfda")
    version("0.11.0", sha256="eab6ef281e0a00180c8b31ffb65984f54216c68464db363a5c09832fec91f952")

    patch("pynn-0.12.2-arbor-0.9.0.patch", when="@0.12.1:0.12.2")

    variant("mpi", default=False, description="Enable MPI support")

    depends_on("python@3.7:",         when="@0.10.0:0.10.1")
    depends_on("python@3.8:",         when="@0.11.0:")
    depends_on("py-setuptools",         type=("build"))
    depends_on("py-setuptools@61:",     type=("build"), when="@0.11:")
    depends_on("py-numpy@1.18.5:",      type=("run", "test"), when="@0.10.1:")
    depends_on("py-mpi4py",             type=("run", "test"), when="+mpi")
    depends_on("py-quantities@0.12.1:", type=("run", "test"), when="@0.9.5:")
    depends_on("py-lazyarray@0.5.2:",   type=("run", "test"), when="@0.10.1:")
    depends_on("py-neo@0.11.0:",        type=("run", "test"), when="@0.10.1:")
    depends_on("py-libneuroml@0.4.1:",  type=("run", "test"), when="@0.12.1:")
    depends_on("py-morphio",            type=("run", "test"), when="@0.12:")
    depends_on("neuron@8.1:+python",    type=("run", "test"), when="@0.10.1:")
    depends_on("nest@3.3:3.4+python",   type=("run", "test"), when="@0.10.1:0.11.0")
    depends_on("nest@3.4:+python",      type=("run", "test"), when="@0.12.1:")
    depends_on("py-brian2",             type=("run", "test"))
    depends_on("arbor@0.8.1:+python",   type=("run", "test"), when="@0.12.1:0.12.2")
    depends_on("arbor@0.9.0:+python",   type=("run", "test"), when="@0.12.3:")
```

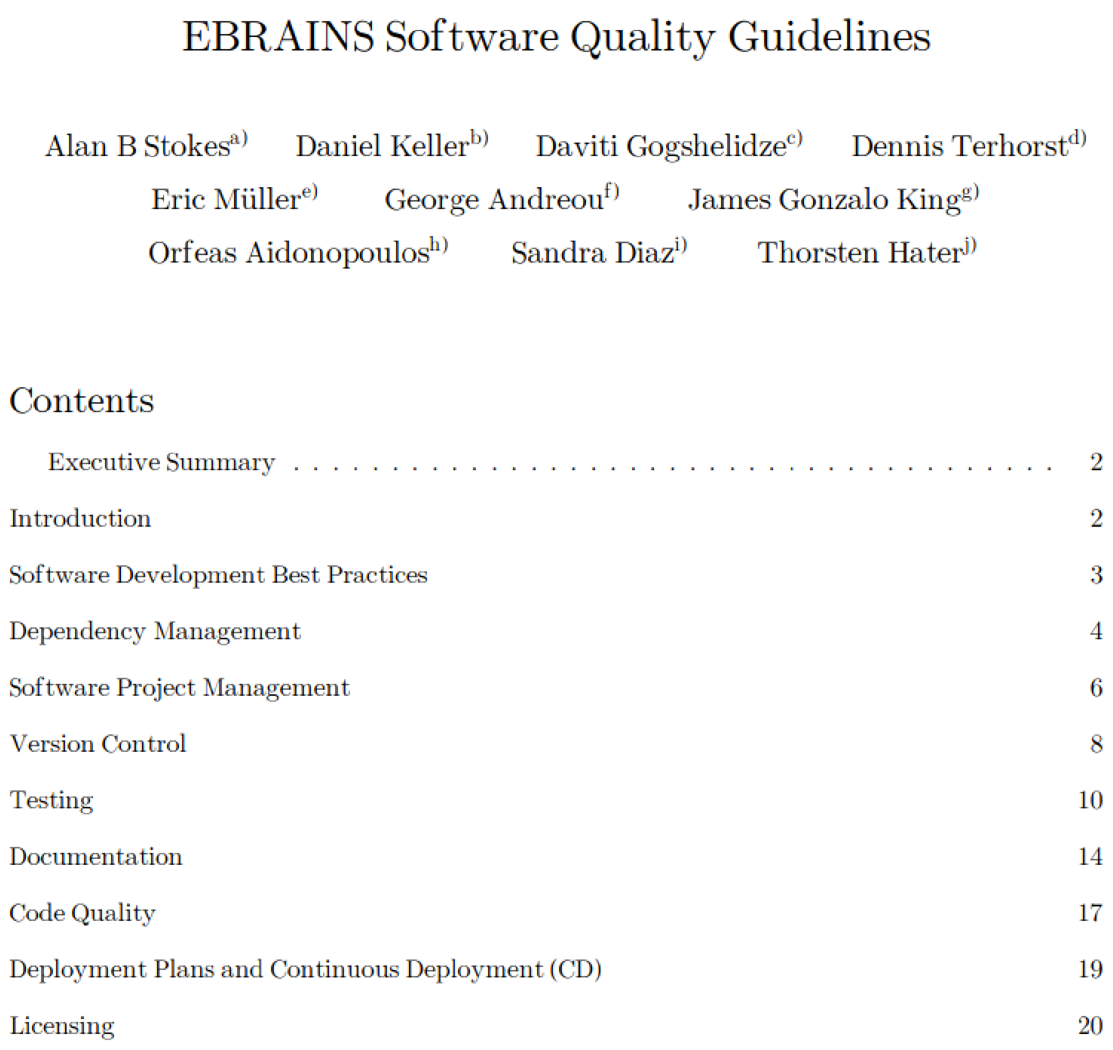
Co-funded by the European Union

Development and Release Flow



- **Official ESD repository:** https://gitlab.ebrains.eu/ri/tech-hub/platform/esd/ebrains-spack-builds
- **Spack** used to define the software stack, dependencies and build instructions: `spack create <url>`
- **MR** in official ESD repository
- **automated build tests** triggered on each commit and MR
- acceptance criteria:
  - passing **build test pipeline**
  - passing **Software Quality Checklist**

Software Quality Checklist



EBRAINS Software Quality Guidelines

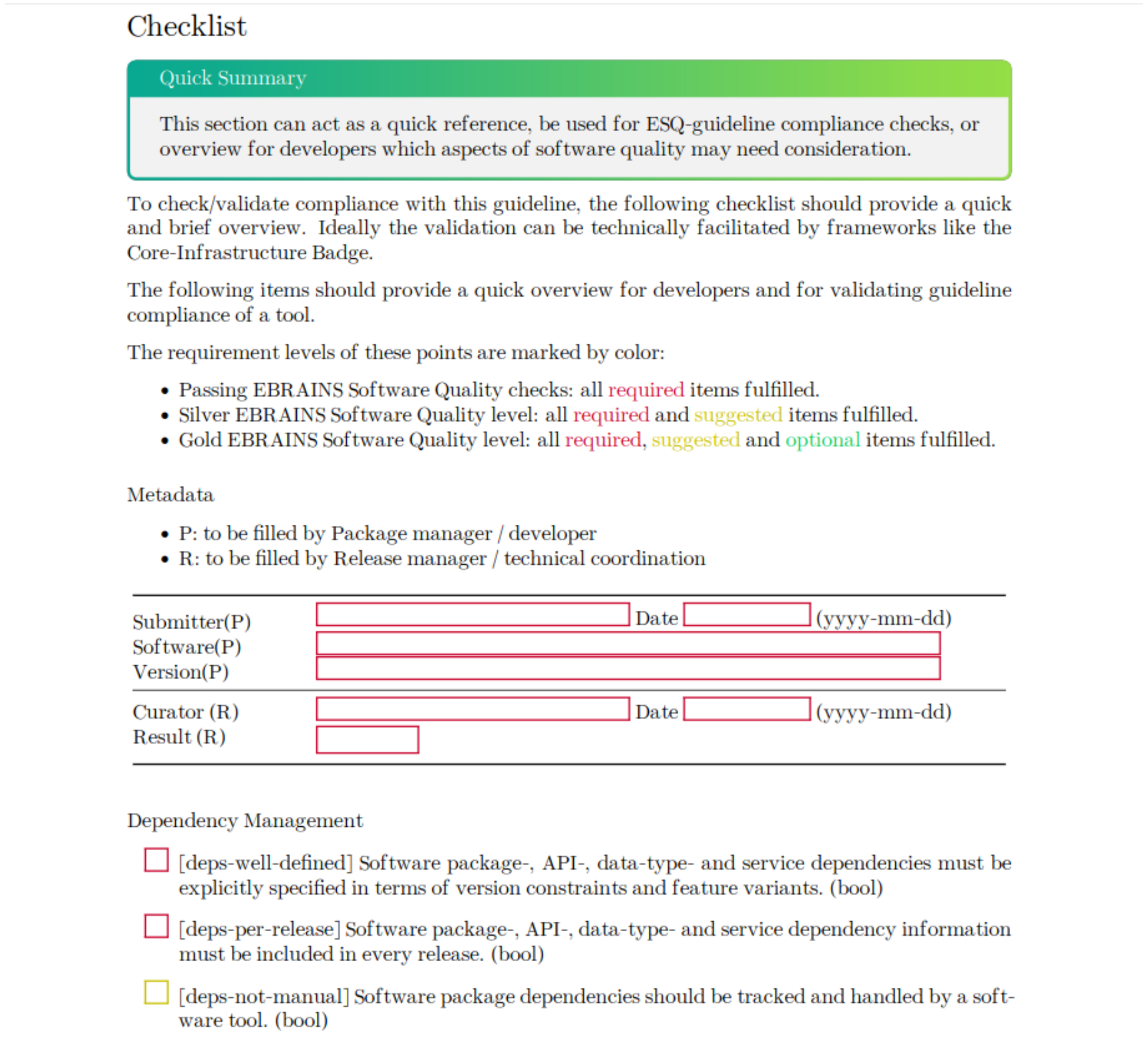
Alan B Stokes[a)] Daniel Keller[b)] Daviti Gogshelidze[c)] Dennis Terhorst[d)]
Eric Müller[e)] George Andreou[f)] James Gonzalo King[g)]
Orfeas Aidonopoulos[h)] Sandra Diaz[i)] Thorsten Hater[j)]

Contents

Executive Summary 2
Introduction 2
Software Development Best Practices 3
Dependency Management 4
Software Project Management 6
Version Control 8
Testing 10
Documentation 14
Code Quality 17
Deployment Plans and Continuous Deployment (CD) 19
Licensing 20

https://drive.ebrains.eu/d/6061531326d048308823/

Software Quality Checklist



Checklist

Quick Summary

This section can act as a quick reference, be used for ESQ-guideline compliance checks, or overview for developers which aspects of software quality may need consideration.

To check/validate compliance with this guideline, the following checklist should provide a quick and brief overview. Ideally the validation can be technically facilitated by frameworks like the Core-Infrastructure Badge.

The following items should provide a quick overview for developers and for validating guideline compliance of a tool.

The requirement levels of these points are marked by color:

- Passing EBRAINS Software Quality checks: all required items fulfilled.
- Silver EBRAINS Software Quality level: all required and suggested items fulfilled.
- Gold EBRAINS Software Quality level: all required, suggested and optional items fulfilled.

Metadata

- P: to be filled by Package manager / developer
- R: to be filled by Release manager / technical coordination

Submitter(P)		Date		(yyyy-mm-dd)
Software(P)				
Version(P)				
Curator (R)		Date		(yyyy-mm-dd)
Result (R)				

Dependency Management

☐ [deps-well-defined] Software package-, API-, data-type- and service dependencies must be explicitly specified in terms of version constraints and feature variants. (bool)

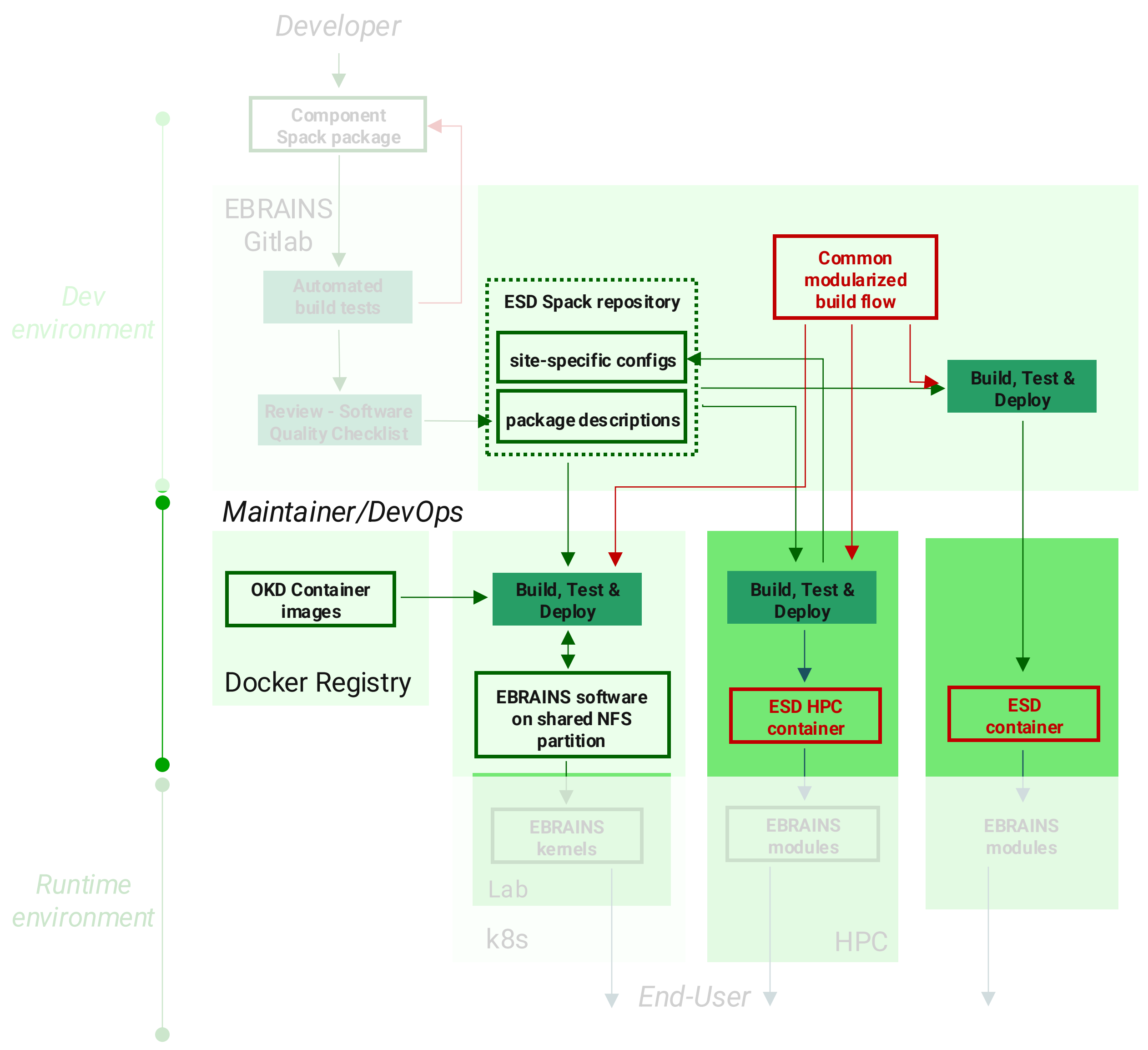
☐ [deps-per-release] Software package-, API-, data-type- and service dependency information must be included in every release. (bool)

☐ [deps-not-manual] Software package dependencies should be tracked and handled by a software tool. (bool)

https://drive.ebrains.eu/d/6061531326d048308823/

Co-funded by the European Union

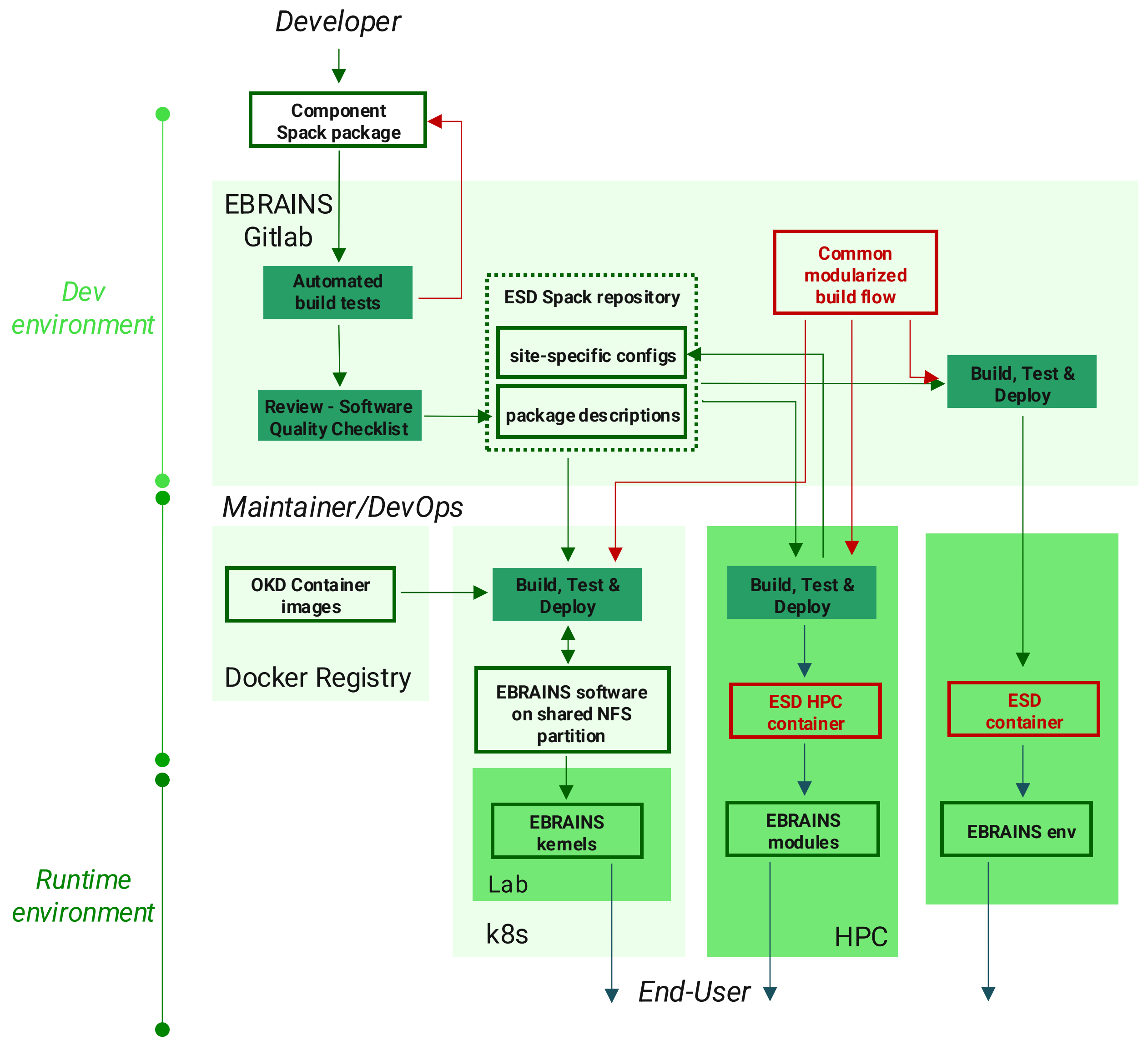
Development and Release Flow

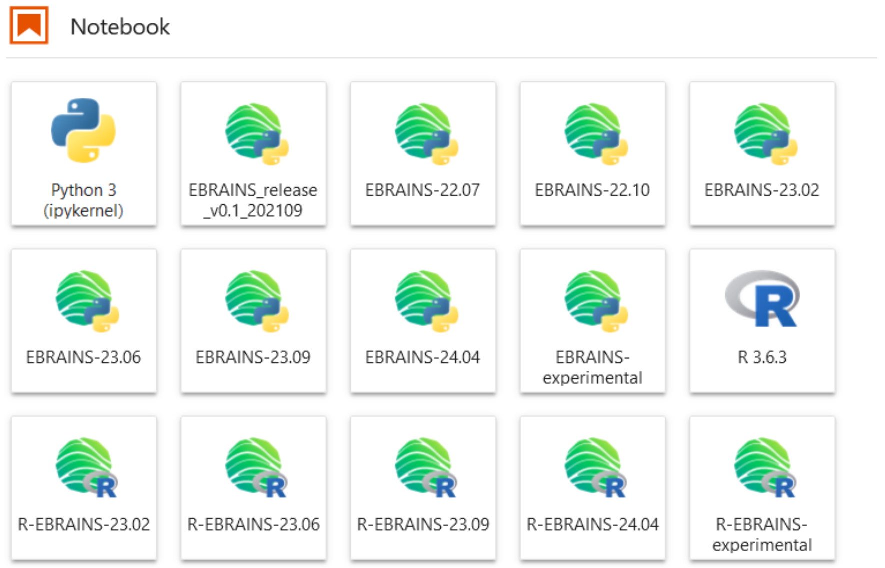


- **Official ESD repository:** https://gitlab.ebrains.eu/ri/tech-hub/platform/esd/ebrains-spack-builds
- **centralized process**, coordinated and performed in GitLab: **fully automated (testing and deployment) flow** using GitLab CI
- **site-specific configurations** decoupled from ESD definition

Co-funded by the European Union

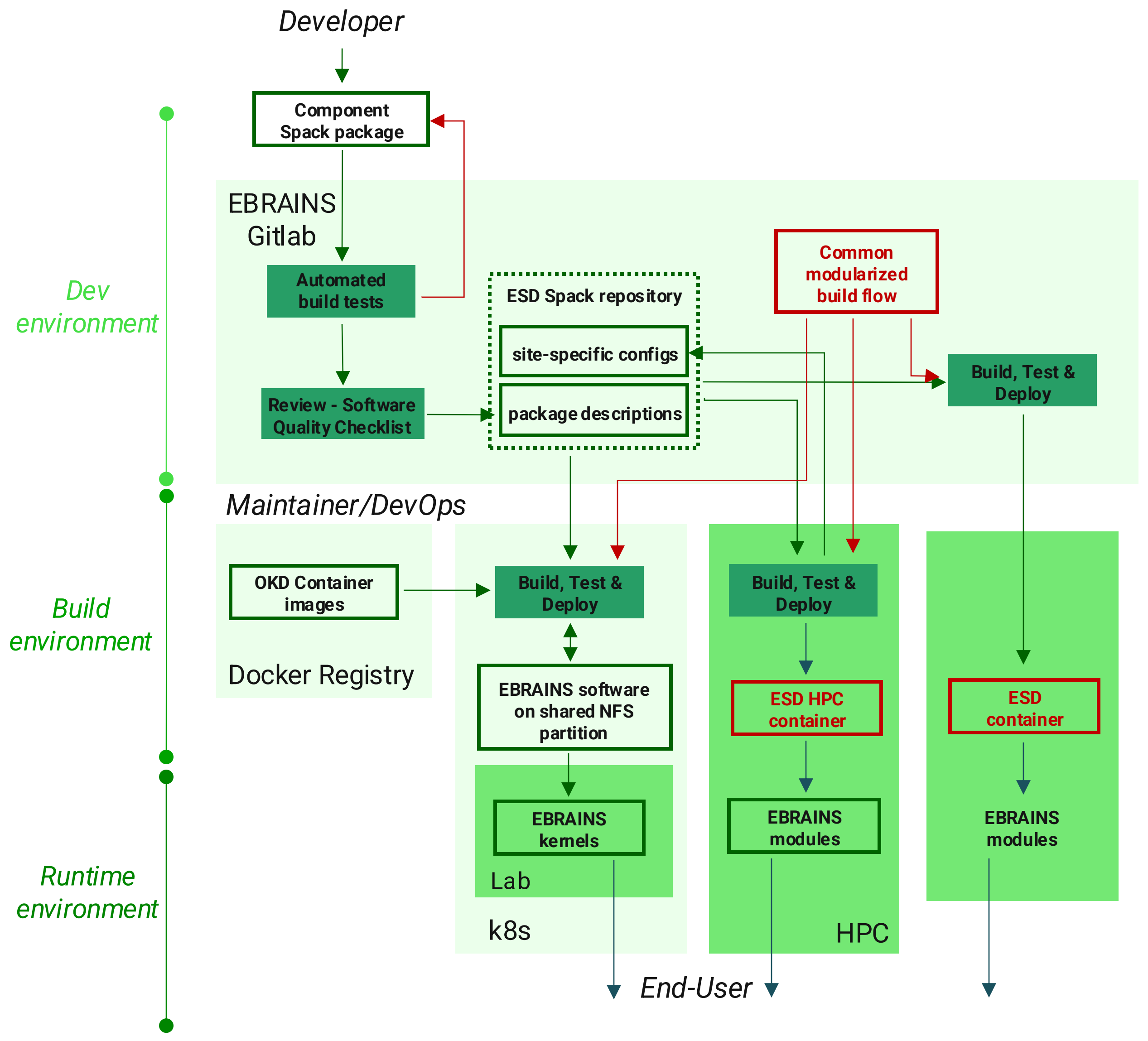
Development and Release Flow





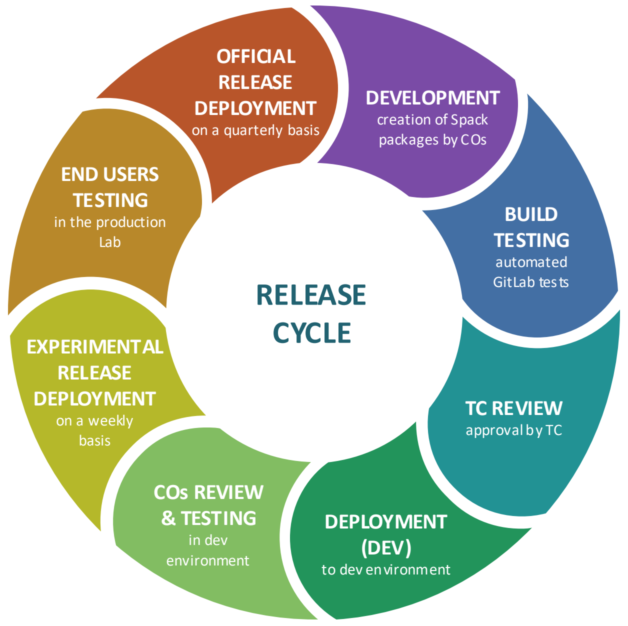
- EBRAINS Lab
  - Interactive JupyterLab environment
- HPC systems
  - high performance and scalability
- EBRAINS "laptop" containers
  - seamless user-deployed workspaces

Development and Release Flow



Official ESD Releases

- **EBRAINS official release**
  - on a **quarterly basis** (older releases remain available)
  - "release candidate" created for testing by end users before each new official release
- **EBRAINS experimental release**
  - on a **weekly basis** (replaced by the next experimental release)
  - not as verified or tested: bleeding edge delivery of new tool features



Release v0.1	EBRAINS 22.07	EBRAINS 22.10	EBRAINS 23.02	EBRAINS 23.06	EBRAINS 23.09	EBRAINS 24.04	EBRAINS 25.02	experimental release
9 EBRAINS tools	21 EBRAINS tools	26 EBRAINS tools	36 EBRAINS tools	55 EBRAINS tools	59 EBRAINS tools	61 EBRAINS tools	64 EBRAINS tools	latest versions (weekly)

available in **EBRAINS Lab** (CSCS and JSC) in **Python kernels**
automated, centralised **build and deployment process**

available also in **R kernel** in EBRAINS Lab
automated **deployment and unit testing**

deployed on ICEI **HPC sites**

Co-funded by the European Union

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)

Co-funded by the European Union

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

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()')
```

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**
- Motivation:
  - **structured representation** of tool interdependencies
  - facilitates **deployment** of workflows
  - facilitates **testing** of workflows (incl. possible service dependencies)

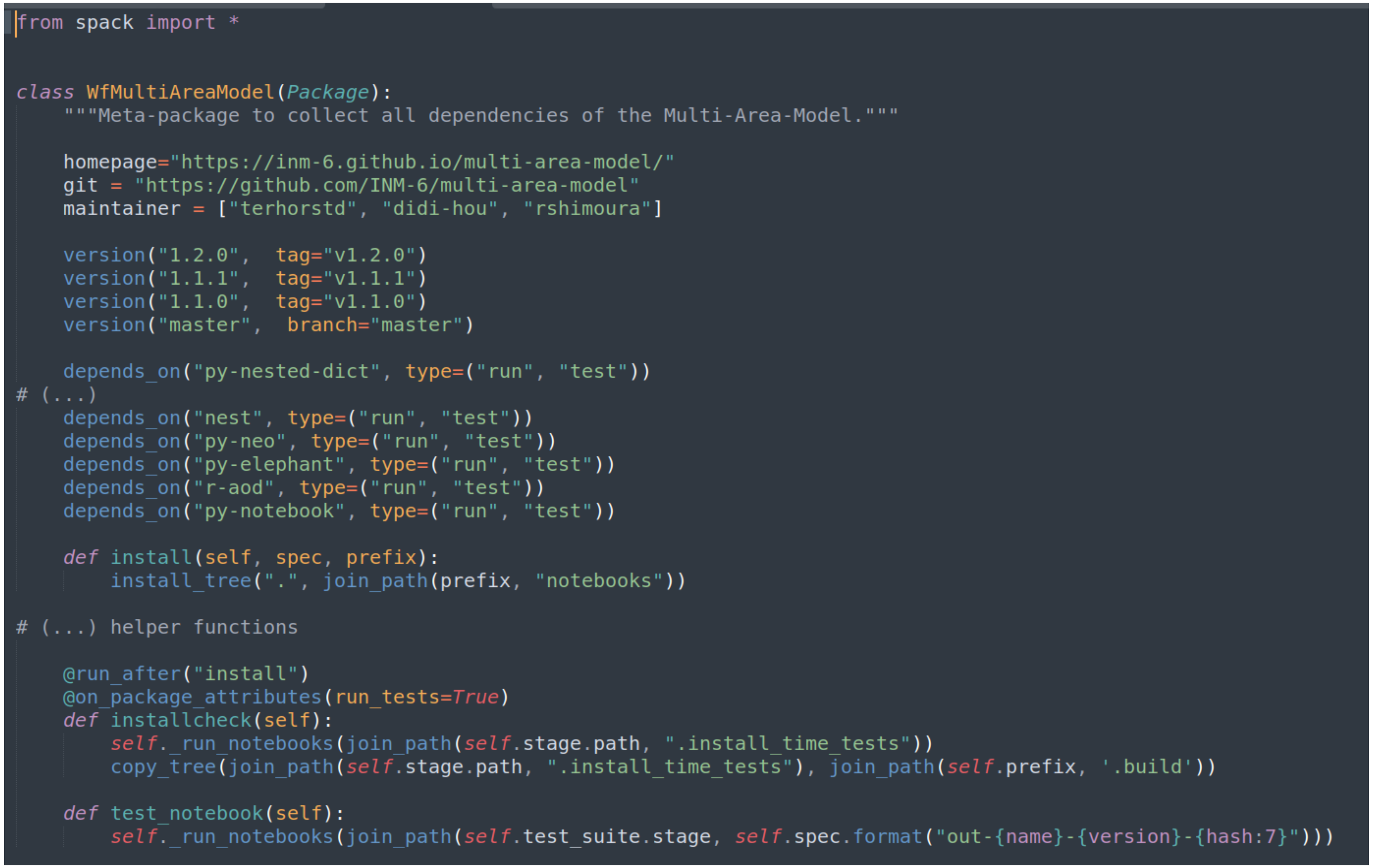
Co-funded by the European Union

ESD benchmark packages

- Named **"bm-{benchmark name}"?**
- Represent real-world EBRAINS tool benchmarks
- Benchmark package definitions include:
  - benchmark code
  - all the additional **dependencies** of the benchmark
  - (if available) expected results
  - (possibly) configurable parameters
- Motivation:
  - measure performance (esp. on HPC containers/deployments)
  - smoke tests

Co-funded by the European Union

ESD workflow packages



```python
from spack import *


class WfMultiAreaModel(Package):
    """Meta-package to collect all dependencies of the Multi-Area-Model."""

    homepage="https://inm-6.github.io/multi-area-model/"
    git = "https://github.com/INM-6/multi-area-model"
    maintainer = ["terhorstd", "didi-hou", "rshimoura"]

    version("1.2.0",  tag="v1.2.0")
    version("1.1.1",  tag="v1.1.1")
    version("1.1.0",  tag="v1.1.0")
    version("master",  branch="master")

    depends_on("py-nested-dict", type=("run", "test"))
# (...)
    depends_on("nest", type=("run", "test"))
    depends_on("py-neo", type=("run", "test"))
    depends_on("py-elephant", type=("run", "test"))
    depends_on("r-aod", type=("run", "test"))
    depends_on("py-notebook", type=("run", "test"))

    def install(self, spec, prefix):
        install_tree(".", join_path(prefix, "notebooks"))

# (...) helper functions

    @run_after("install")
    @on_package_attributes(run_tests=True)
    def installcheck(self):
        self._run_notebooks(join_path(self.stage.path, ".install_time_tests"))
        copy_tree(join_path(self.stage.path, ".install_time_tests"), join_path(self.prefix, '.build'))

    def test_notebook(self):
        self._run_notebooks(join_path(self.test_suite.stage, self.spec.format("out-{name}-{version}-{hash:7}")))
```

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)

Co-funded by the European Union

The team / Get Involved

- **EBRAINS Software Distribution: Integration and Quality WG**
  - Tuesday, 11:00 CEST
  - all ESD-related topics: integration and testing aspects, software quality, (non-HPC) container images, workflow packages, etc
- **EBRAINS Software Distribution on HPC WG**
  - Friday, 10:00 CEST
  - all ESD HPC-related aspects such as deployment, (performance) optimization and packaging
- **Rocketchat channel: https://chat.ebrains.eu/channel/ebrains-releases**

EBRAINS

EBRAINS 2.0

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.