EESSI

EUROPEAN ENVIRONMENT FOR SCIENTIFIC SOFTWARE INSTALLATIONS

Software testing in EESSI

EESSI Community Meeting @ Amsterdam

16 Sept 2022

Caspar van Leeuwen (SURF)

Single software stack, many users…

- Providing software installations is not sufficient…
- … we need to ensure they work!

Image: Wikimedia Commons

Testing the EESSI software stack

- What should be tested?
- How to test?
- When to test?
- Where to test?

Image:publicdomainpictures.net

Compatibility layer

Relatively easy…

- Mostly basic tests
- Can we find executable XYZ
- Can we find file/link ABC
- … etc
- https://github.com/EESSI/compatibility-layer/blob/main/test/compat_layer.py

Image: pix4free.org

What to test?

Software layer

- Different purposes:
  - **Smoke testing**: lightweight tests to catch major problems (blatantly broken stuff)
  - **Functional testing**: quick tests with limited resources
  - **Performance testing**: do we observe expected performance?
  - **Integration testing**: does the EESSI software stack work on my system?
  - **Monitoring**: frequent test runs to ensure things keep working
- EESSI test suite

Image: pix4free.org

How to test?

Test with ReFrame

- Designed for testing & benchmarking on HPC systems
- Interacts with batch schedulers
- Can also execute tests locally (i.e. without batch scheduler)
- Supports both functionality and performance testing

Image: Wikimedia Commons

When & where to test?

Will need to figure out & discuss what is practical & sufficient. E.g.

- **Smoke testing**
  - Checking for blatantly broken things (missing binaries, etc)
  - When installing software, updated a dependency, …
  - Build node
- **Functional testing**
  - Checking if software produces expected (scientific) results
  - When installing software
  - Build node / cluster in the cloud
- **Performance testing**
  - Checking if software performs as expected
  - When installing software
  - Cluster in the cloud

Image: Wikimedia Commons

When & where to test?

Will need to figure out & discuss what is practical & sufficient.

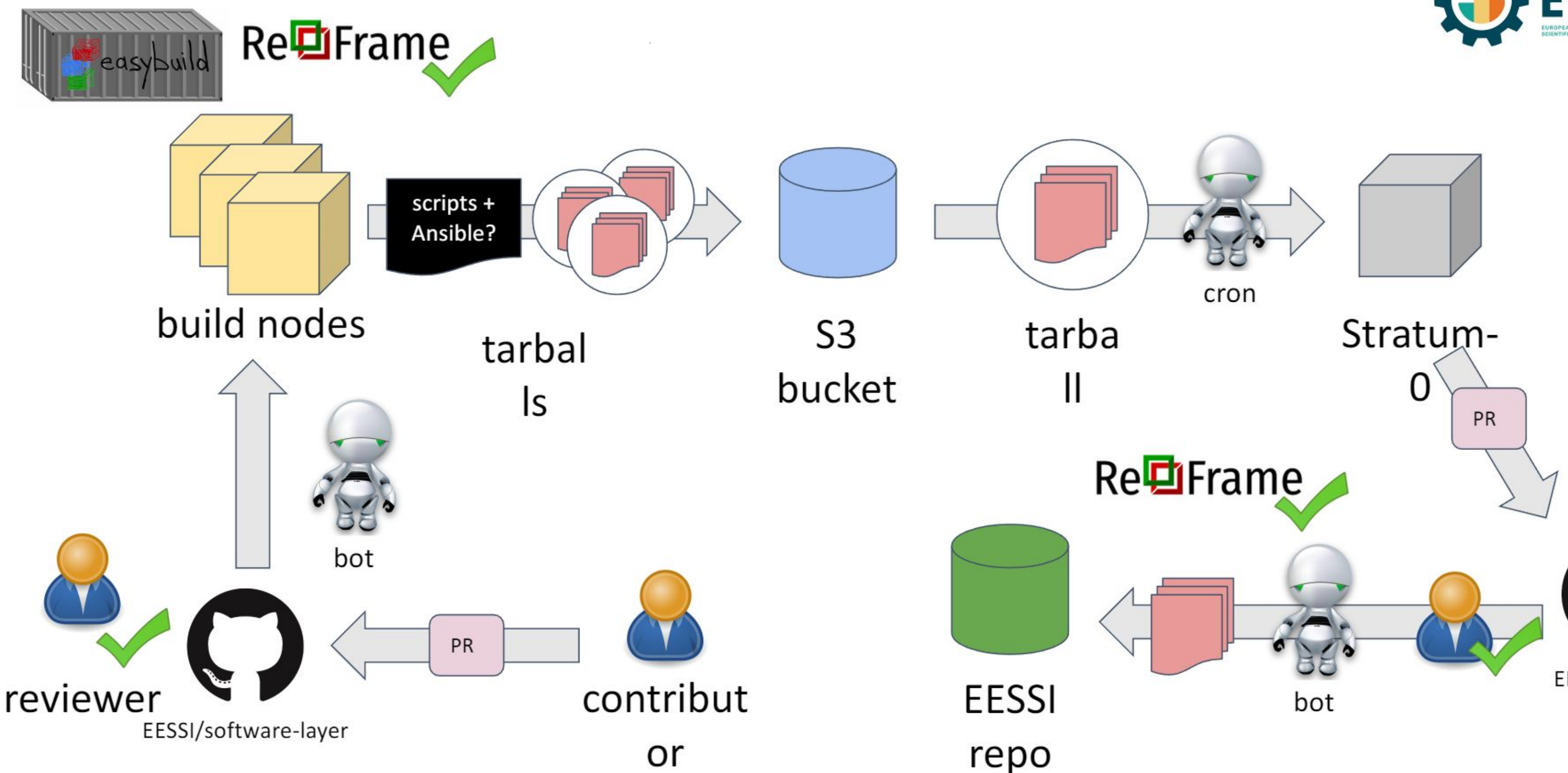
E.g.

- **Integration testing**
  - End user / HPC admin runs (part of) EESSI test suite at mount time
  - As part of functional tests: cluster in the cloud @ various architectures
- **Monitoring**
  - At fixed schedule, run functionality & performance tests
  - Cluster in the cloud, clusters of EESSI partners (?)

Image: Wikimedia Commons

Current view on automated deployment

Goal: automated, with human oversight



Current view on automated deployment

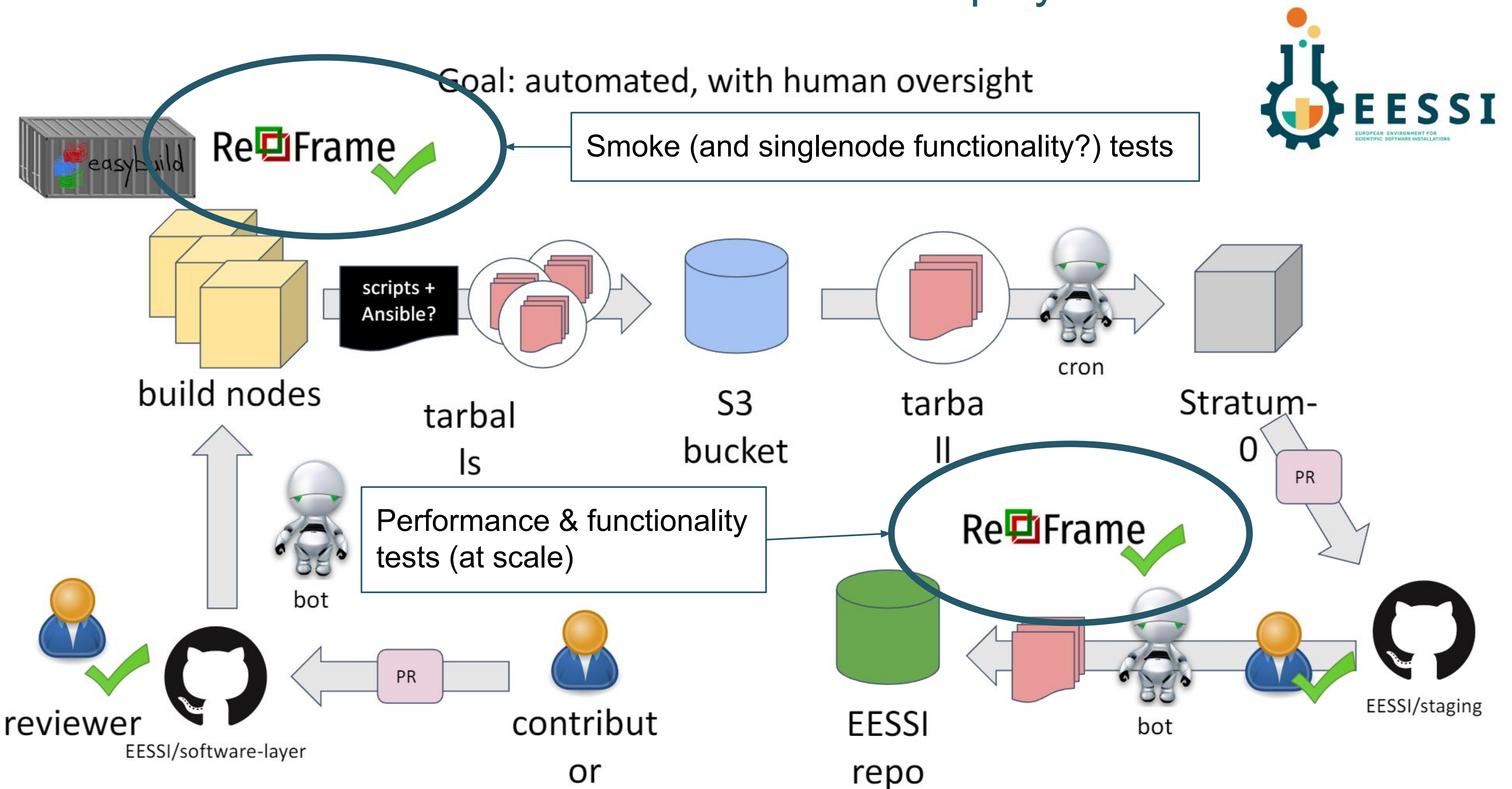

Goal: automated, with human oversight
Smoke (and singlenode functionality?) tests
ReFrame
easybuild
build nodes
scripts + Ansible?
tarbal ls
S3 bucket
tarba ll
cron
Stratum-0
PR
Performance & functionality tests (at scale)
bot
ReFrame
reviewer
EESSI/software-layer
PR
contribut or
EESSI repo
bot
EESSI/staging
EESSI
EUROPEAN ENVIRONMENT FOR SCIENTIFIC SOFTWARE INSTALLATIONS

Image: freesvg.org

Portability of test

- It should be easy to run the EESSI test suites (even non-expert users on a laptop!)
- Low bar to entry to run test across various
- Ideally: one-time, minimal configuration to specify system characteristics
  - E.g. GPU tests should not be executed on a system that does not contain GPUs
- Test logic should be decoupled from system-specific details
  - E.g. “Run with 12 threads” is not ok; “Run test with one thread per core” is ok
- Software stack is common - one less thing to worry about!
- Specific features where added in ReFrame to facilitate this
  https://github.com/reframe-hpc/reframe/pull/2479

Image: freesvg.org

Portability of test

- Standard ReFrame was designed for in house tests, not portability
- Example …

Collaboration with application experts

A test suite that will be run on a wide range of architectures is *also* valuable for application experts and developers!

- Codesign tests with application experts
  - Which aspects/features of software should be tested?
  - What are viable inputs?
  - How to verify functional correctness?
  - What is the expected performance (given a set of basic system features)?
- Application devs can leverage EESSI in CI for compilers, dependencies, …

Image: freesvg.org

Testing EESSI: current status

- Initial test for compatibility layer
  - https://github.com/EESSI/compatibility-layer/blob/main/test/compat_layer.py
- Initial functional tests for software layer are work-in-progress
  - GROMACS: https://github.com/EESSI/software-layer/pull/156
  - TensorFlow/Horovod: https://github.com/EESSI/software-layer/pull/122

Image: freesvg.org

Testing EESSI: future outlook

- Auto-generate (part of) ReFrame's config file
  - E.g. though ReFrame's ability to autodetect processor information https://reframe-hpc.readthedocs.io/en/stable/configure.html#auto-detecting-processor-information
- Use the new 'features' support to specify required features, instead of our current logic in detecting e.g. presence of a GPU (https://github.com/reframe-hpc/reframe/pull/2479)
- Performance testing: open challenge...
  - How to determine expected performance for an application on a given system?
  - How do we implement "portable" performance tests?
- Should each software request *always* come with a test?
  - Create dashboard to show which software is tested?

Demo/hands-on: run GROMACS on CitC

- Login to CitC: `ssh <github_handle>@3.250.220.9`
- Start interactive job:

srun -N1 -n8 -C shape=c5a.16xlarge --time=1:0:0 --pty /bin/bash

```
# Install reframe, or use the reframe from EESSI
virtualenv reframe_venv
source reframe_venv/bin/activate
pip install reframe-hpc==3.12.0 –user

# Need to clone Reframe as well
# default test suite (hpctestlib) not part of standard install
git clone -b v3.12.0 https://github.com/reframe-hpc/reframe.git

# Clone the Gromacs test
git clone -b gromacs_cscs https://github.com/casparvl/software-layer.git
```

Demo/hands-on: run GROMACS on CitC

```
# Make sure the hpctestlib and eessi_utils are found
export PYTHONPATH=$PYTHONPATH:$PWD/reframe:$PWD/software-layer/tests/reframe

cd software-layer/tests/reframe

# Edit the config file, or copy an existing one for your system
cp config/settings_magic_castle.py config/settings.py
vi config/settings.py

    'systems': [
        {
            'name': 'citc',
             …
            'partitions': [
                {
                    'scheduler': 'squeue',
                    'access':  ['-C shape=c5a.16xlarge'],
                    'processor': {
                        'num_cpus': 64,
                    },
                    …
```

Demo/hands-on: run GROMACS on CitC

```
# List tests
reframe --config-file=config/settings.py --checkpath eessi-checks/applications/
--system=citc -l

# Limit with tags
reframe --config-file=config/settings.py --checkpath eessi-checks/applications/
--system=citc -l -t CI -t singlenode

# Run
reframe --config-file=config/settings.py --checkpath eessi-checks/applications/
--system=citc -t CI -t singlenode -r --performance-report
```

Demo/hands-on: how portable is the GROMACS test?

- Have access to your own system? Try along!
- Note: you don't need EESSI for this! But, your GROMACS modules need to be visible with 'module av' in the terminal where you run the reframe command

```
# Install reframe, or use the reframe from EESSI
virtualenv reframe_venv
source reframe_venv/bin/activate
pip install reframe-hpc==3.12.0

# Need to clone Reframe as well
# default test suite (hpctestlib) not part of standard install
# If using the reframe from EESSI, make sure to clone the same version (3.9.1)
git clone -b v3.12.0 https://github.com/reframe-hpc/reframe.git

# Clone the Gromacs test
git clone -b gromacs_cscs https://github.com/casparvl/software-layer.git
```

Demo/hands-on: how portable is the GROMACS test?

```
# Make sure the hpctestlib and eessi_utils are found
export PYTHONPATH=$PYTHONPATH:$PWD/reframe:$PWD/software-layer/tests/reframe

cd software-layer/tests/reframe

# Edit the config file, or copy an existing one for your system
cp config/settings_magic_castle.py config/settings.py
vi config/settings.py
```

Demo/hands-on: how portable is the GROMACS test?

…

```
'modules_system': 'lmod',
'hostnames': ['login', 'node'],
'partitions': [
    {
        'name': 'cpu',
        'scheduler': 'slurm',
        'launcher': 'mpirun',
        'access':  ['-p cpubase_bycore_b1 --exclusive --mem=94515M'],
        'environs': ['builtin'],
        'max_jobs': 4,
        'processor': {
            'num_cpus': 36,
        },
        'descr': 'normal CPU partition'
    },
```

…

Demo/hands-on: how portable is the GROMACS test?

EESSI

EUROPEAN ENVIRONMENT FOR SCIENTIFIC SOFTWARE INSTALLATIONS

```
# List tests
reframe --config-file=config/settings.py --checkpath eessi-checks/applications/ -l

# Limit with tags
reframe --config-file=config/settings.py --checkpath eessi-checks/applications/ -l -t CI -t
singlenode

# Run
reframe --config-file=config/settings.py --checkpath eessi-checks/applications/ -t CI -t
singlenode -r --performance-report
```

Inspecting gromacs_check.py

```
@rfm.simple_test
class GROMACS_EESSI(gromacs_check):

    scale = parameter([
        ('singlenode', 1),
        ('n_small', 2),
        ('n_medium', 8),
        ('n_large', 16)])
    module_info = parameter(find_modules('GROMACS', environ_mapping={r'.*': 'builtin'}))
```

Inherits from the hpctestlib ‘gromacs_check’, which defines all the test cases, sanity pattern, performance pattern, etc

Generate test at various scales

Run test once for every module that matches ‘GROMACS’

Inspecting gromacs_check.py

```
# Set correct tags for monitoring & CI
@run_after('init')
def set_test_purpose(self):
    # Run all tests from the testlib for monitoring
    self.tags.add('monitoring')
    # Select one test for CI
    if self.benchmark_info[0] == 'HECBioSim/hEGFRDimer':
        self.tags.add('CI')
```

Tag all test cases with the ‘monitoring’ tag

Only tag this particular test case with the ‘CI’ tag

Inspecting gromacs_check.py

```
# Skip testing GPU-based modules on CPU-based nodes
    @run_after('setup')
    def skip_gpu_test_on_cpu_nodes(self):
        hooks.skip_gpu_test_on_cpu_nodes(self)
```

Current logic in our eessi_utils hooks to skip tests on nodes that don't have GPUs (according to ReFrame config). Can probably be improved with https://github.com/reframe-hpc/reframe/pull/2479

```
    # Assign num_tasks, num_tasks_per_node and num_cpus_per_task automatically based on
current partition's num_cpus and gpus
    @run_after('setup')
    def set_num_tasks(self):
        hooks.auto_assign_num_tasks_MPI(test = self, num_nodes = self.num_nodes)
```

Hook that controls hybrid execution (number of processes vs threads). Currently: 1 process per GPU (for GPU test), or 1 per CPU core (CPU tests).

Summarizing GROMACS test

- ReFrame tags can be used to select what runs where (CI, monitoring, etc)
- Custom hooks provide capability of skipping tests on hardware where they don’t make sense (e.g. GPU test on CPU node). Can be partially replaced with native ReFrame ‘features’.
- Custom hooks can set some generic execution behaviour (e.g. 1 task per core, 1 task per socket + 1 thread per core, etc)
- Performance currently reported, but no reference specified (does not generalize).

EESSI
EUROPEAN ENVIRONMENT FOR SCIENTIFIC SOFTWARE INSTALLATIONS

EESSI

EUROPEAN ENVIRONMENT FOR SCIENTIFIC SOFTWARE INSTALLATIONS

Website: https://www.eessi-hpc.org

Join our mailing list & Slack channel

https://www.eessi-hpc.org/join

Documentation: https://eessi.github.io/docs

GitHub: https://github.com/eessi

Twitter: @eessi_hpc

Open monthly online meetings
(first Thursday, 2pm CET)