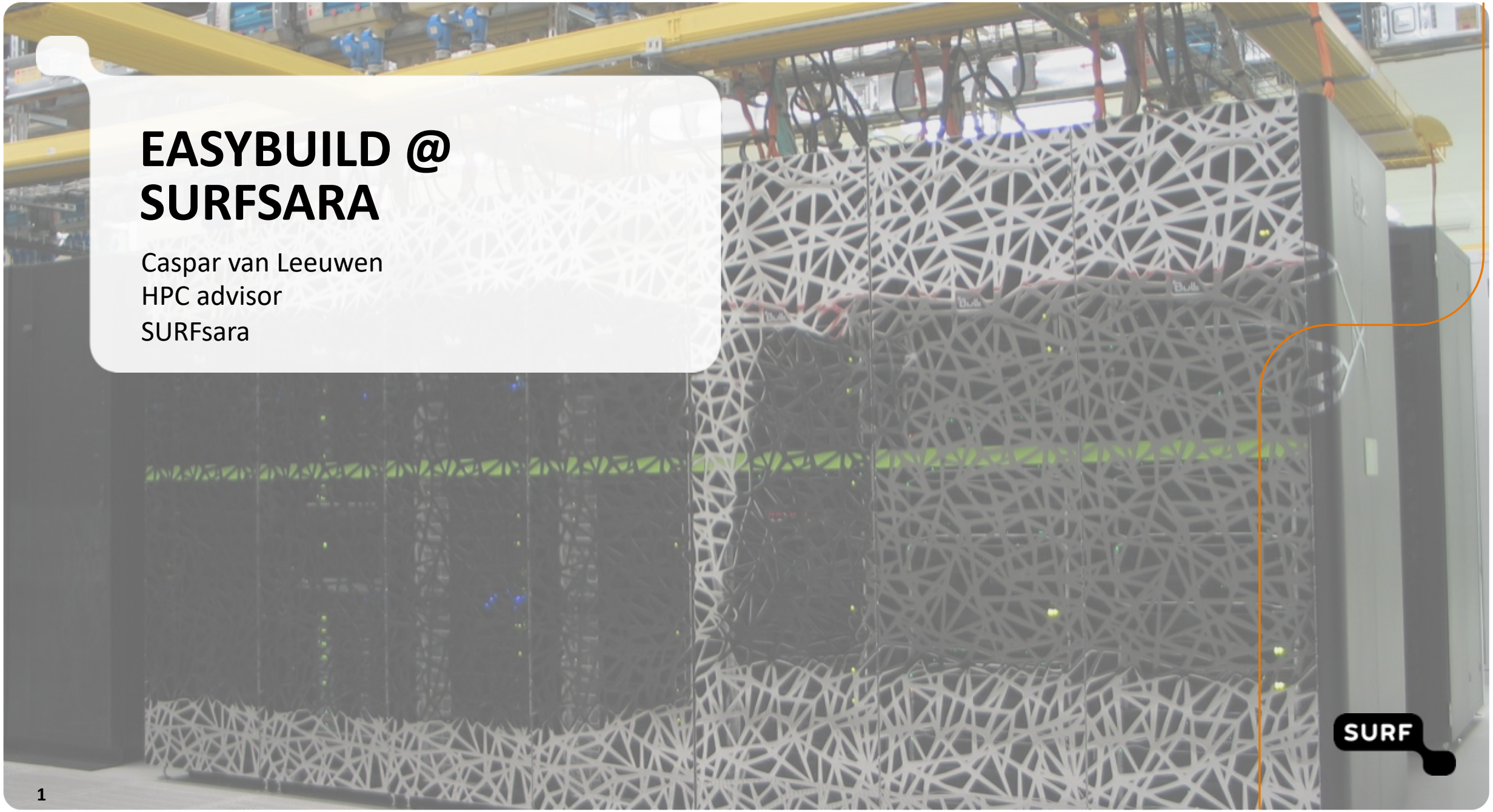


EASYBUILD @ SURFSARA

Caspar van Leeuwen
HPC advisor
SURFsara

SURF(sara)

Dutch national supercomputing center

- Supercomputing
- Clustercomputing
- Scientific visualization
- Data services
- High performance cloud

Overview

- History / motivation
- EasyBuild
- Jenkins
- ReFrame
- Xalt
- Remaining challenges

Swiss army knife for maintaining a software environment
Discovered @ CSCS site presentation, 3rd EasyBuild User meeting

A bit of history…

Pre-EasyBuild era @ SURFsara

- Hand-crafted bash installation scripts (*some* template)
- Hand-made modulefiles

A bit of history…

Issues

- Installation scripts: what is happening where (and why)? Where do I bump the version?
- Modulefiles for various versions of software X are all different…
- Different software environment @ cluster (Lisa) & supercomputer (Cartesius)
- “Incomplete” environment: software X only built with compiler A; software Y only with compiler B; only old installations of software Z available.
- Overall: poor reproducibility, lots of work to update, unclear which dependencies were used, etc

A bit of history…

Pre-ReFrame era @ SURFsara

- Repository with some bash scripts

Issues

- Test scripts are non-standard => difficult to read for others
- Modules loaded by test scripts need to be updated every time
- Results are not stored; performance was typically not tested (just functionality)

EasyBuild @ SURFsara - timeline

- November 2016: EasyBuild 3.0.0 released with RPATH support
- Mid 2017: first (publicly available) installations @ SURFsara through EasyBuild
- Start 2019: EasyBuild becomes the default installation method. Manual installation only by *rare* exception!

What EasyBuild brings us …

A clean, ‘complete’ environment

- We try to compile all ‘permutations’ of software versions, toolchains, suffixes etc.
- Two toolchains: issues with intel + RHEL 7.4[1] in 2017 motivated us to always have a fallback!

```
[casparl@int2 ~]$ module av GROMACS
--------- /sw/arch/.../modulefiles/bio ---------
GROMACS/2019.3-foss-2018b                  GROMACS/2019.3-intel-2018b
GROMACS/2019.3-foss-2018b-CUDA-10.0.130  GROMACS/2019.3-intel-2018b-CUDA-10.0.130
```

[1] https://software.intel.com/en-us/articles/inconsistent-program-behavior-on-red-hat-enterprise-linux-74-if-compiled-with-intel

What EasyBuild brings us …

An environment that is

```
[casparl@int2 ~]$ module av -l | grep -v '\-\-\-' | wc -l
582
```

… easy to create

- We installed close to 600 modules in 2019
- New colleague: “This is almost too easy. We shouldn’t tell anyone, or we’ll be out of a job!”

… easy to update

- Update policy: release complete new stack once per year
- Profit from EasyConfigs contributed by the community, add the rest ourselves

What EasyBuild brings us …

Professional ‘home’ installations

- User: ”I want X, but linked against Y”
  - Not common to all users
  - Prepare EasyConfig => local installation
- Wrapper *eblocalinstall:* EASYBUILD_INSTALLPATH = ~/.local/…
- Automatically generated modulefiles for local installs
- Users can easily share with direct colleagues

What EasyBuild brings us …

A community of fellow experts, with similar issues. Our community…

- … prefers source builds
- … performs optimized builds

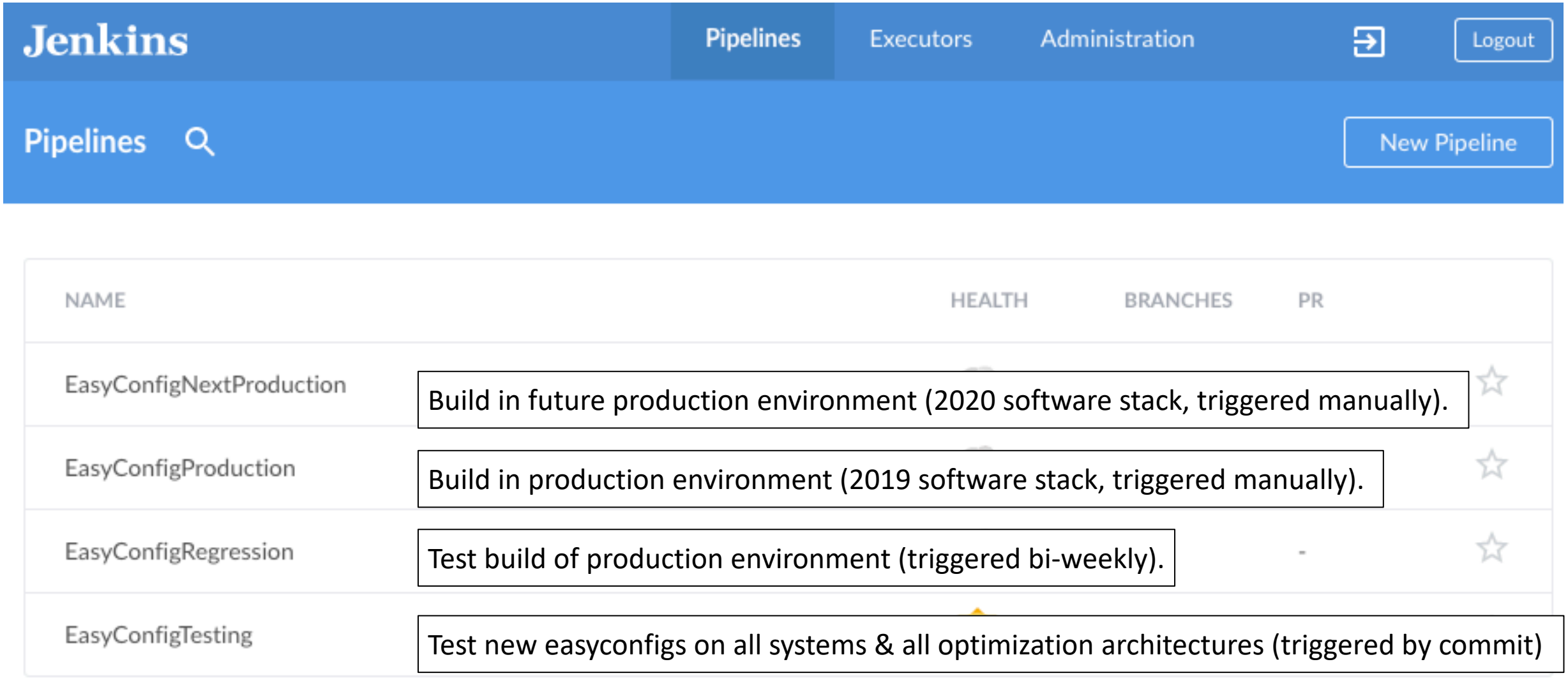
As a result, we run into issues that others never encountered!

- Search an installation error => regularly land on EB issue page (and solution!)

What is Jenkins?

Designed for continuous integration. Typical usage:

- Jenkins runs on a VM
- Connects to local or remote workers (e.g. multiple HPC systems)
- Jenkins *pipeline* file defines what should be tested and/or deployed (e.g. invoke *eb <easyconfig>)*
- “Normal CI”: Upon success, take some automatic decision (deploy in production, merge to master branch, etc) - we don’t do this part ☺


Jenkins
Pipelines
Executors
Administration
Logout
Pipelines
New Pipeline
NAME
HEALTH
BRANCHES
PR
EasyConfigNextProduction
Build in future production environment (2020 software stack, triggered manually).
EasyConfigProduction
Build in production environment (2019 software stack, triggered manually).
EasyConfigRegression
Test build of production environment (triggered bi-weekly).
EasyConfigTesting
Test new easyconfigs on all systems & all optimization architectures (triggered by commit)

EasyConfigTesting < 2293 >

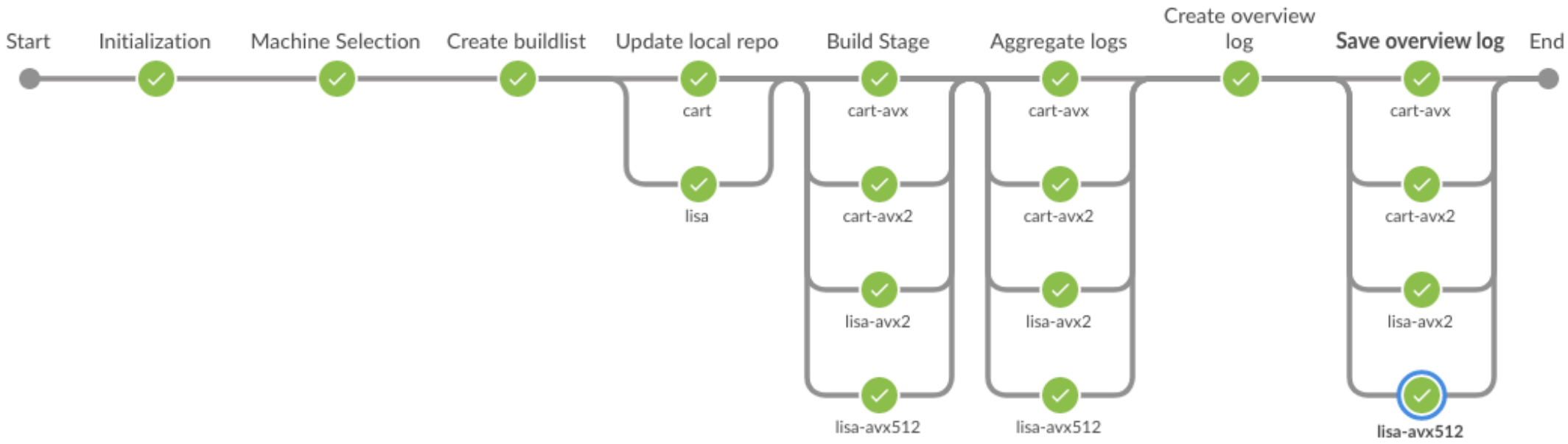
Pipeline | Changes | Tests | Artifacts | Logout

Branch: –
Commit: –

3m 33s
a day ago

Changes by Caspar van Leeuwen
Started by GitLab push by Caspar van Leeuwen

Description [cart,lisa] [NCCL-2.4.8-gcccuda-2019b.eb] [nextproduction] Test the testing pipeline...



Save overview log / lisa-avx512 - 20s

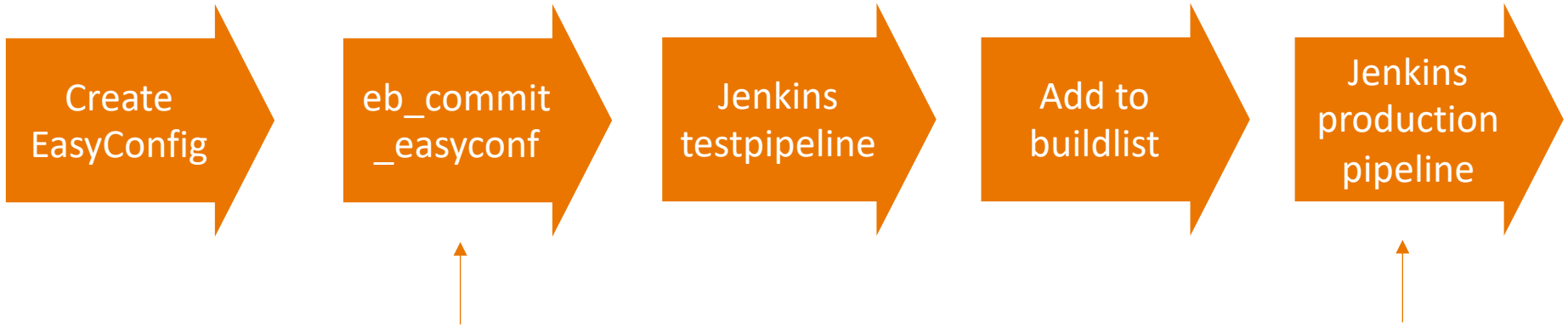
✓	> overview_logfile — Restore files previously stashed	<1s
✓	> srun ${jobarg_short} cp overview_summary.log ${JENKINS_LOGPATH_REAL}/overview_summary.log — Shell Script	20s

Aggregated log for builds on all systems + architectures:

Print Message <1s

EasyConfig	lisa-avx512	lisa-avx2	cart-avx2	cart-avx
GCCcore-7.3.0.eb	SUCCESS	SUCCESS	SUCCESS	SUCCESS
binutils-2.30-GCCcore-7.3.0.eb	SUCCESS	SUCCESS	SUCCESS	SUCCESS
GCC-7.3.0-2.30.eb	SUCCESS	SUCCESS	SUCCESS	SUCCESS
GCC-8.2.0-2.31.1.eb	SUCCESS	SUCCESS	SUCCESS	SUCCESS
OpenMPI-3.1.1-GCC-7.3.0-2.30-fixed-lisa.eb	SUCCESS	SUCCESS	SKIPPED	SKIPPED
OpenMPI-3.1.1-GCC-7.3.0-2.30.eb	ALREADY INSTALLED	ALREADY INSTALLED	SUCCESS	SUCCESS
gompi-2018b.eb	SUCCESS	SUCCESS	SUCCESS	SUCCESS
FFTW-3.3.8-gompi-2018b.eb	SUCCESS	SUCCESS	SUCCESS	SUCCESS
FFTW-3.3.8-gompi-2018b-avx.eb	SKIPPED	SKIPPED	SKIPPED	SKIPPED
ScaLAPACK-2.0.2-gompi-2018b-OpenBLAS-0.3.1.eb	SUCCESS	SUCCESS	SUCCESS	SUCCESS
foss-2018b.eb	SUCCESS	SUCCESS	SUCCESS	SUCCESS
icc-2018.3.222-GCC-7.3.0-2.30.eb	SUCCESS	SUCCESS	SUCCESS	SUCCESS
ifort-2018.3.222-GCC-7.3.0-2.30.eb	SUCCESS	SUCCESS	SUCCESS	SUCCESS
iccifort-2018.3.222-GCC-7.3.0-2.30.eb	SUCCESS	SUCCESS	SUCCESS	SUCCESS
icc-2019.1.144-GCC-8.2.0-2.31.1.eb	SUCCESS	SUCCESS	SUCCESS	SUCCESS
ifort-2019.1.144-GCC-8.2.0-2.31.1.eb	SUCCESS	SUCCESS	SUCCESS	SUCCESS
iccifort-2019.1.144-GCC-8.2.0-2.31.1.eb	SUCCESS	SUCCESS	SUCCESS	SUCCESS
impi-2018.3.222-iccifort-2018.3.222-GCC-7.3.0-2.30-tcp.eb	SUCCESS	SUCCESS	SKIPPED	SKIPPED
impi-2019.1.144-iccifort-2019.1.144-GCC-8.2.0-2.31.1-tcp.eb	SUCCESS	SUCCESS	SKIPPED	SKIPPED
impi-2018.3.222-iccifort-2018.3.222-GCC-7.3.0-2.30.eb	ALREADY INSTALLED	ALREADY INSTALLED	SUCCESS	SUCCESS
impi-2019.1.144-iccifort-2019.1.144-GCC-8.2.0-2.31.1.eb	ALREADY INSTALLED	ALREADY INSTALLED	SUCCESS	SUCCESS
iimpi-2018b.eb	SUCCESS	SUCCESS	SUCCESS	SUCCESS
iimpi-2019a.eb	SUCCESS	SUCCESS	SUCCESS	SUCCESS
imkl-2018.3.222-iimpi-2018b.eb	SUCCESS	SUCCESS	SUCCESS	SUCCESS
imkl-2019.1.144-iimpi-2019a.eb	SUCCESS	SUCCESS	SUCCESS	SUCCESS
intel-2018b.eb	SKIPPED	SKIPPED	SUCCESS	SUCCESS
intel-2019a.eb	SUCCESS	SUCCESS	SUCCESS	SUCCESS
FFTW-3.3.8-intel-2018b.eb	SUCCESS	SUCCESS	SUCCESS	SUCCESS
YAXT-0.6.0-intel-2018b.eb	SUCCESS	SUCCESS	SUCCESS	SUCCESS
YAXT-0.6.0-foss-2018b.eb	SUCCESS	SUCCESS	SUCCESS	SUCCESS
Python-2.7.15-foss-2018b.eb	ALREADY INSTALLED	ALREADY INSTALLED	SUCCESS	SUCCESS
Python-2.7.15-intel-2018b.eb	ALREADY INSTALLED	ALREADY INSTALLED	SUCCESS	SUCCESS
Python-2.7.15-GCCcore-7.3.0-bare.eb	ALREADY INSTALLED	ALREADY INSTALLED	SUCCESS	SUCCESS
Python-3.6.6-foss-2018b.eb	SUCCESS	SUCCESS	SUCCESS	SUCCESS
Python-3.6.6-intel-2018b.eb	SUCCESS	SUCCESS	SUCCESS	SUCCESS
Tkinter-2.7.15-foss-2018b-Python-2.7.15.eb	ALREADY INSTALLED	ALREADY INSTALLED	SUCCESS	SUCCESS
Tkinter-2.7.15-intel-2018b-Python-2.7.15.eb	ALREADY INSTALLED	ALREADY INSTALLED	SUCCESS	SUCCESS

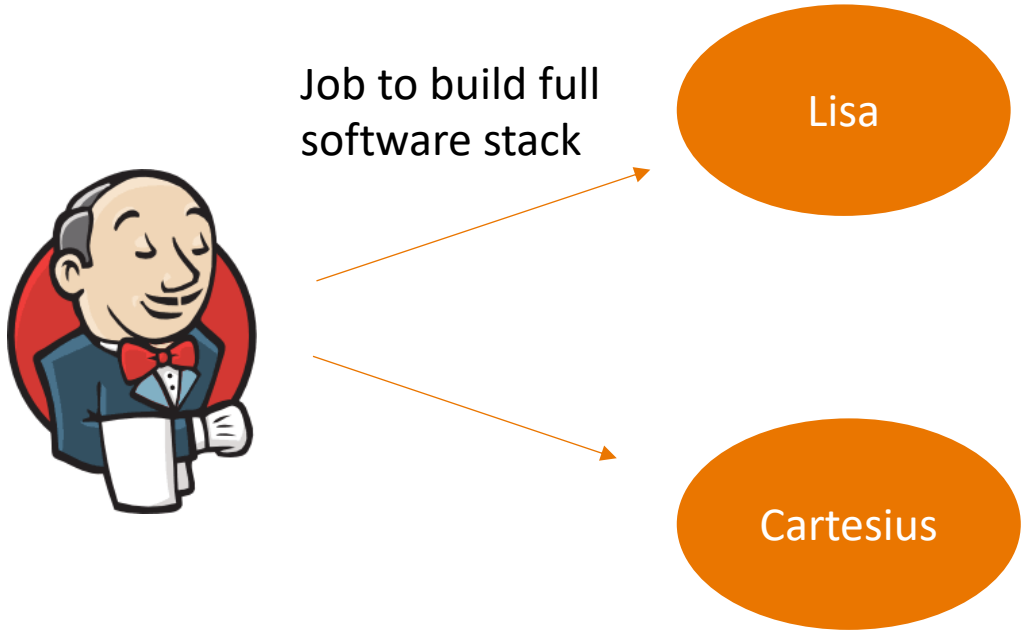
Current software installation workflow



What Jenkins brings us …

Identical software stacks on our cluster (Lisa) and supercomputer (Cartesius)

- Software availability should NOT be a factor in choosing the most suitable system for a user



What Jenkins brings us …

Regression pipeline provides (some) guarentee that we can reinstall software

- If a system change breaks the installation process of an EasyConfig, we notice within 2 weeks
- The 2-week window helps us nail down *which* change broke it.

What Jenkins brings us …

Architecture-specific optimization

Modulefiles only refer to the softlinks whenever they set a path.

Job to build full software stack

AVX node

Prefix = /sw/arch → Resolves to /hpc/arch/AVX

AVX2 node

Prefix = /sw/arch → Resolves to /hpc/arch/AVX2

WARNING: setup only works if software is present in both prefixes. Otherwise, a module may be loaded on one architecture, if the job lands on another, the installation is not available there

What Jenkins brings us …

Architecture-specific optimization that is transparent to the user (no more ‘gromacs-avx’ or ‘gromacs-avx2’ modules!)

AVX capable node =>

```
[casparl@tcn180 ~]$ module load GROMACS/2019.3-foss-2018b
[casparl@tcn180 ~]$ which gmx_mpi
/sw/arch/…/software/GROMACS/2019.3-foss-2018b/bin/gmx_mpi
[casparl@tcn180 ~]$ realpath $(which gmx_mpi)
/hpc/arch/AVX/…/software/GROMACS/2019.3-foss-2018b/bin/gmx_mpi
```

AVX2 capable node =>

```
[casparl@tcn900 ~]$ module load GROMACS/2019.3-foss-2018b
[casparl@tcn900 ~]$ which gmx_mpi
/sw/arch/…/software/GROMACS/2019.3-foss-2018b/bin/gmx_mpi
[casparl@tcn900 ~]$ realpath $(which gmx_mpi)
/hpc/arch/AVX2/…/software/GROMACS/2019.3-foss-2018b/bin/gmx_mpi
```

What EasyBuild + Jenkins brings us …

Better ability to deprecate installations! Policy:

- Install 1 complete stack per year
- Support for 2 years
- Deprecated in 3rd year
- Removed (hidden) in 4th year

https://userinfo.surfsara.nl/documentation/software-policy-lisacartesius

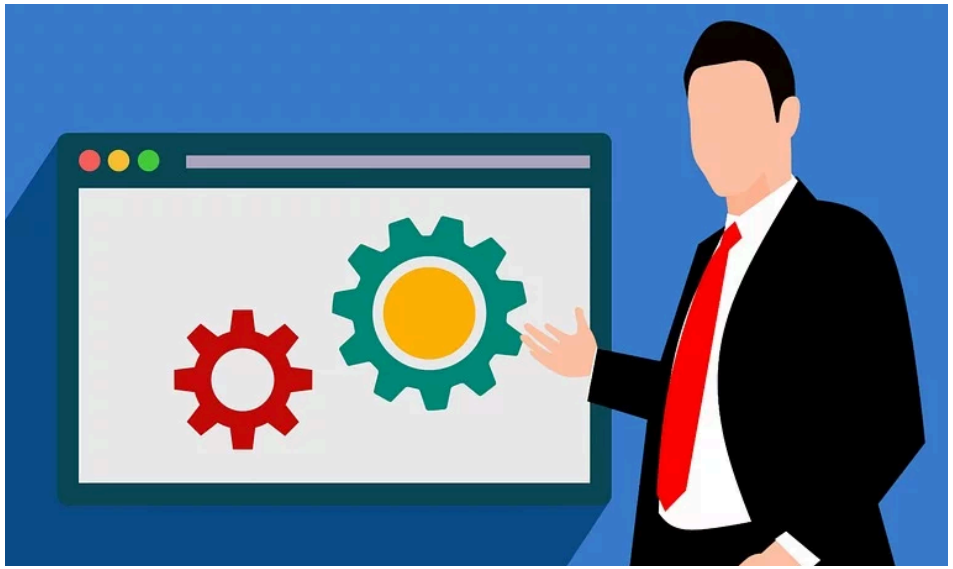
‘Meta-modules’

- Module load 2019 => modules installed in 2019 become available
- Confronts users with how old installation is that they are using!

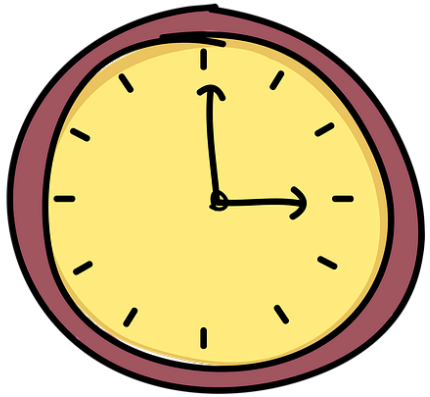
Why ReFrame

Software testing

- Standardized test scripts => easy to read
- ReFrame allows both performance and functionality tests



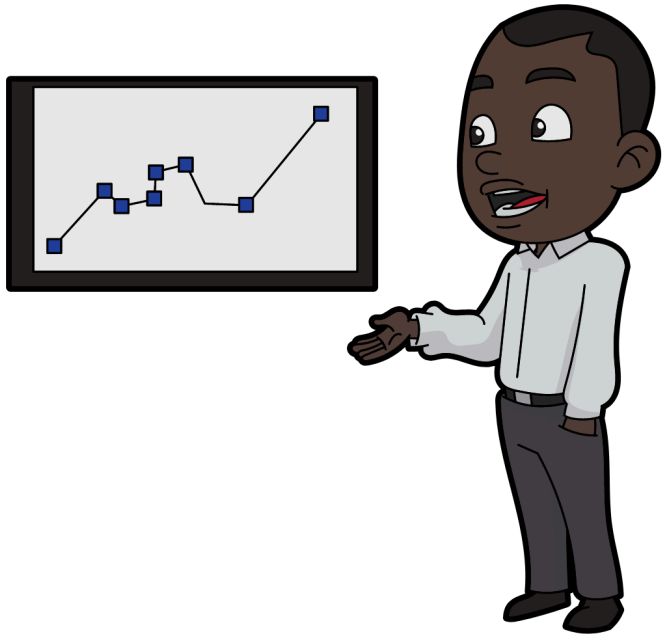
ReFrame @ SURFsara



- Developed *some* tests
- ReFrame can build & test, or test existing modules
- Reframe allows module mapping
  - GROMACS => GROMACS/2019.3-foss-2018b
  - New module enviroment? Update mapping, not tests
- Now defining a test suite, to run
  - Periodically
  - Before & after system upgrades (hardware, OS)

What ReFrame will bring us …

- More professional, consistent testing before/after system upgrades
- Insight in performance…
  - … between our systems
  - … between toolchains
  - … between old and new versions of software



Xalt

- Tool to track software usage
- Stores usage in an SQL database
- Technical implementation: done

Challenges

- Privacy issues: what do we need to put where? Anonimization/pseudononimize? Opt in/opt out? How long can we keep what?
- Documentation is a bit sparse on some topics (e.g. what additional info can I expect when I set '--with-trackGPU=yes')

What Xalt will bring us …

- Which modules to keep supporting (yearly update)
- Which software worth to spend time on (if installation fails, optimization, etc)
- Which proprietary software is (not) worth to spend money on
- Targetted communication to users (updates on certain modules; software alternatives; bugs found in modules/software)

Remaining challenges / discussion starters

- Do we need EasyBlock-specific documentation?
  - We have *eb –ae <easyblock>*, but we may need more. E.g. various EasyBlocks automatically add --with-X=$EBROOTX. That is *usually* what you want, completely invisible now (I now have to check PRs to figure out what happens & why).
  - Suggestion: give EasyBlocks a <docs> property that can be queried in a similar way to *eb –ae <easyblock>*

Remaining challenges / discussion starters

- Time from opening a PR until they are merged is sometimes very long.
  - Ok for 1 or 2 PRs, but it gets hard to keep track when jugling more. (what was I working on? Is it waiting for me, or reviewer?)
  - Note sure if there is a solution (channel in EB slack specifically for quick interaction with maintainer of the week?)

Remaining challenges / discussion starters

- Some EasyConfigs do a make check or similar, most don't
  - *Should* EasyBuild test installations? Or is it better to rely on e.g. ReFrame?
  - Is there an EasyBuild configuration to skip these for institutes that test installation functionality externally (e.g. ReFrame)? If not: should there be?

Remaining challenges / discussion starters

- When we write EasyConfigs for new toolchains, which versions should we use for dependencies?
  - PRs are checked that deps match previous versions… Can we use the same tool when developing?
  - Suggestion: can the core EasyBuild team nail down a version list in a way similar as defining the toolchains as soon as a toolchain is released?
  - Could integrate well with the eb ---tweak deps = <file_with_deplist> option proposed by Kenneth