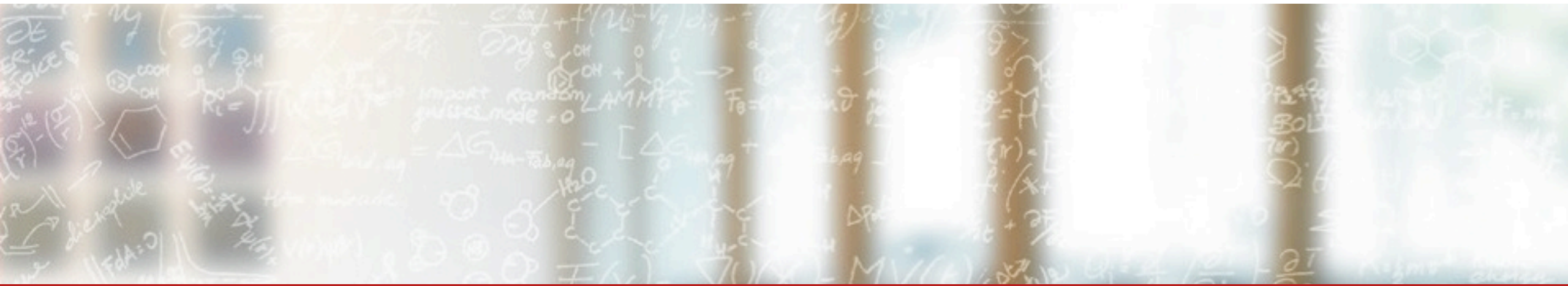




**CSCS**

Centro Svizzero di Calcolo Scientifico  
Swiss National Supercomputing Centre

**ETH** zürich



# EasyBuild @ CSCS: Current status and roadmap

EasyBuild Workshop

Guilherme Peretti-Pezzi, CSCS

September 8<sup>th</sup>, 2015

# Outline



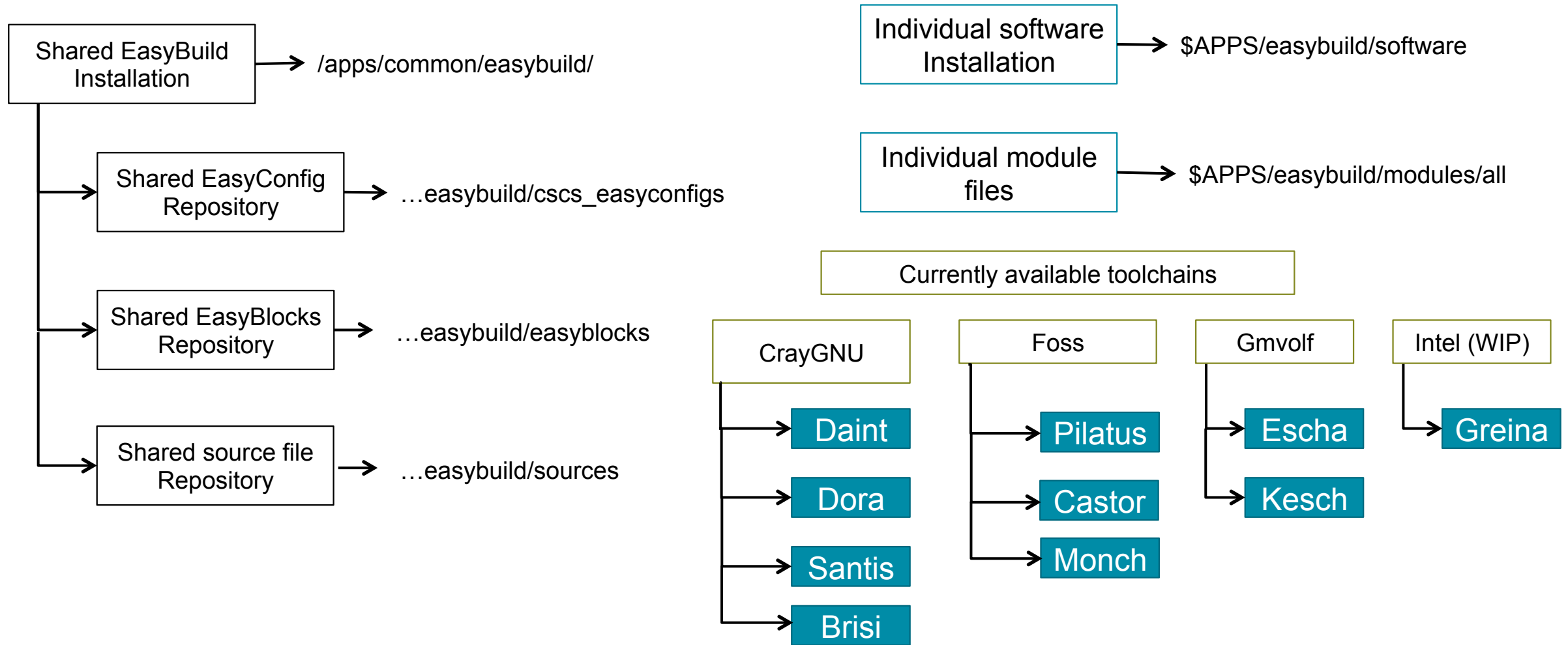
- EasyBuild setup @ CSCS
- Proposed workflow
- Jenkins integration
- Final thoughts & questions

# Some of the stock EasyBuild toolchains

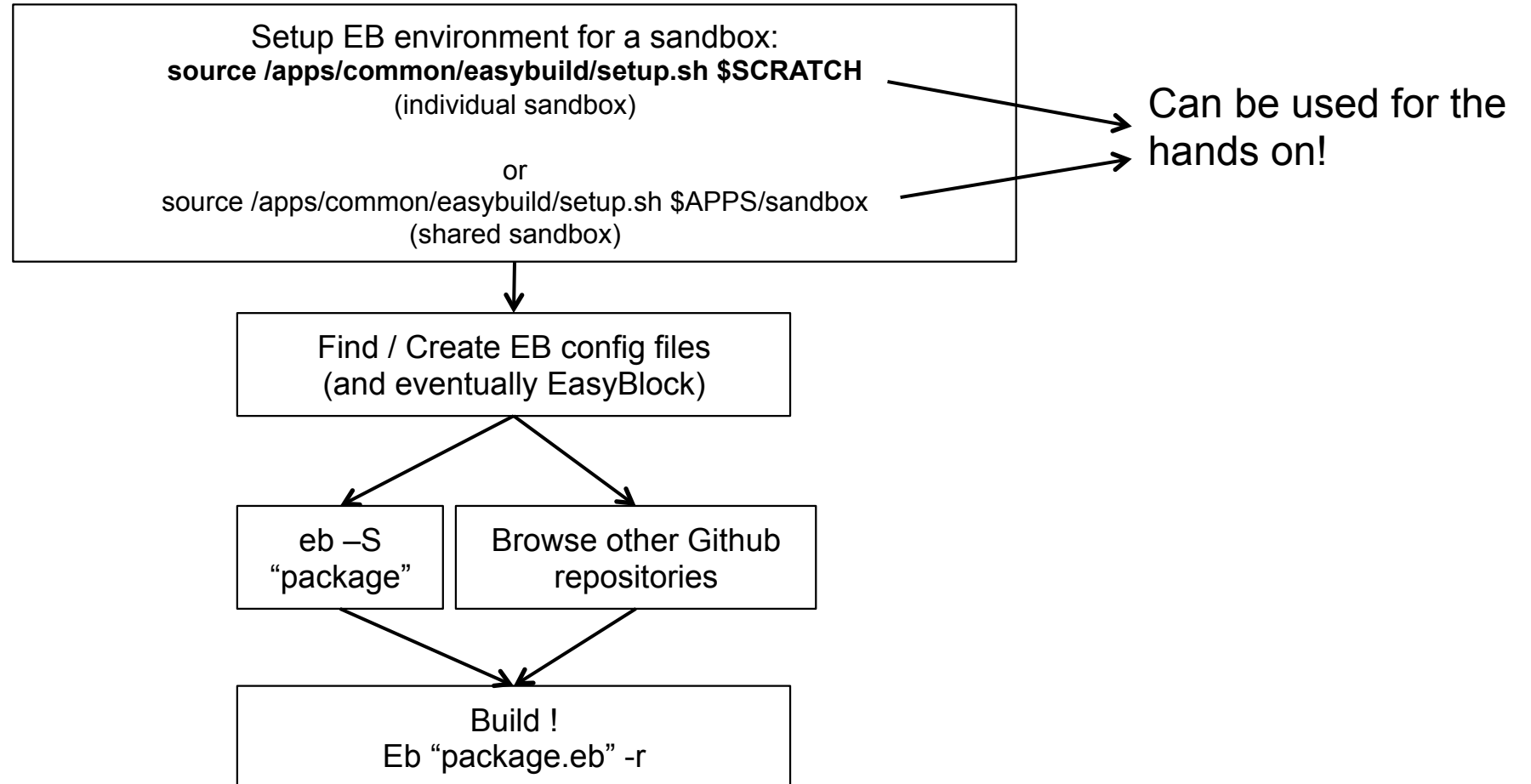
- ClangGCC: Clang, GCC
- CrayCCE: PrgEnv-cray, fftw
- **CrayGNU: PrgEnv-gnu, fftw**
- CrayIntel: PrgEnv-intel, fftw
- GCC: GCC
- cgmpich: Clang, GCC, MPICH
- cgmvpich2: Clang, GCC, MVAPICH2
- cgompi: Clang, GCC, OpenMPI
- **dummy: (system libs and compilers)**
- **foss: BLACS, FFTW, GCC, OpenBLAS, OpenMPI, ScaLAPACK**
- gcccuda: CUDA, GCC
- **gmvpich2: GCC, MVAPICH2**
- gmvolf: BLACS, FFTW, GCC, MVAPICH2, OpenBLAS, ScaLAPACK
- gompic: CUDA, GCC, OpenMPI
- gpsolf: BLACS, FFTW, GCC, OpenBLAS, ScaLAPACK, psmpl
- iccifort: icc, ifort
- ictce: icc, ifort, imkl, impi
- **intel: icc, ifort, imkl, impi**
- iomkl: OpenMPI, icc, ifort, imkl
- iqacml: ACML, BLACS, FFTW, QLogicMPI, ScaLAPACK, icc, ifort

Full list available with:  
`eb --list-toolchains`

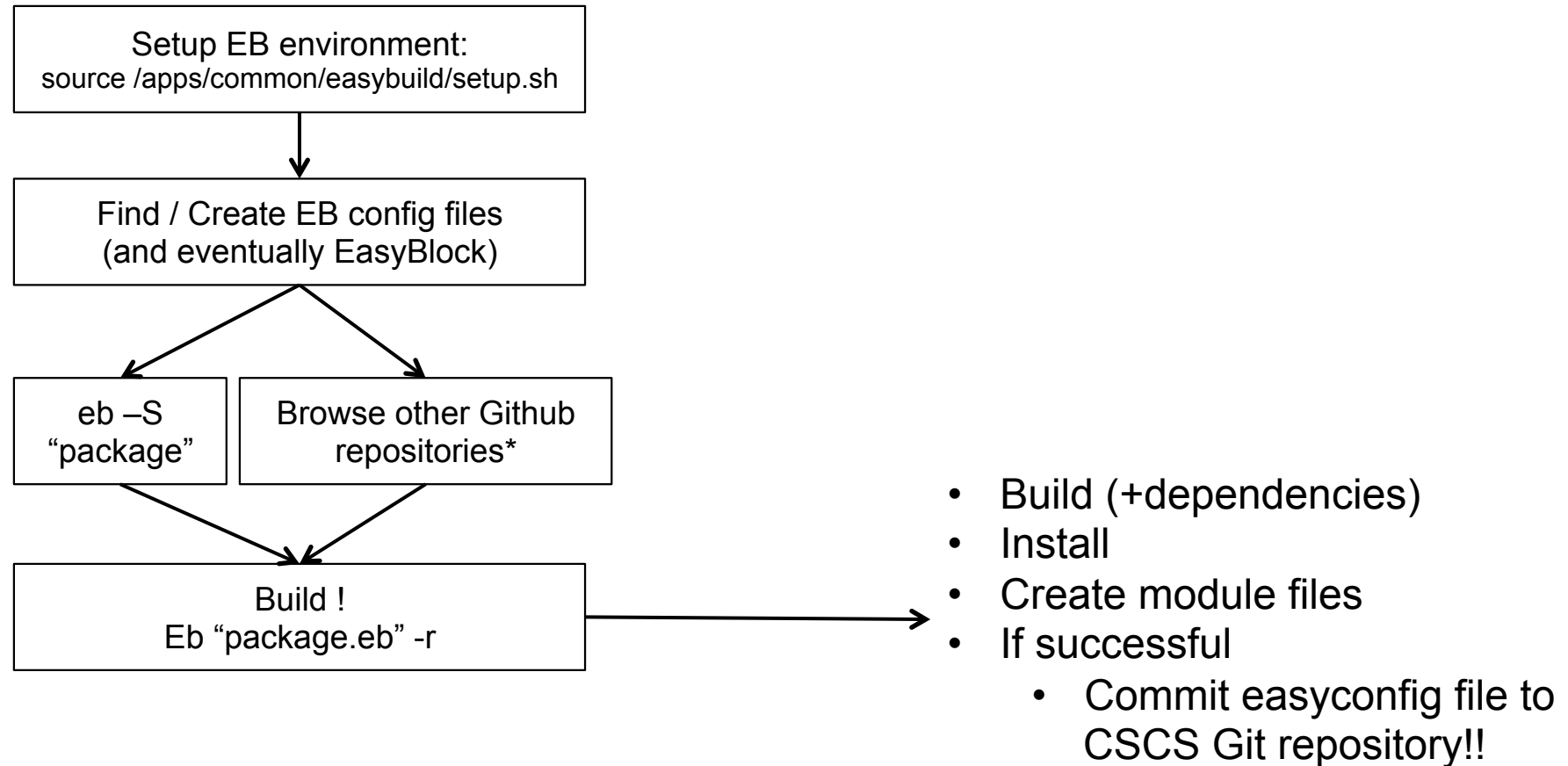
# EasyBuild setup @ CSCS



# Proposed EasyBuild workflow for development (usable by all CSCS)



# Proposed EasyBuild workflow for production builds (SCS):



\*Links on the last slide

# Python use case

- Supported modules for Python 2 and 3
  - Setuptools 17.1.1, Pip 7.0.3, Nose 1.3.7, Numpy 1.9.2, Scipy 0.15.1, mpi4py 1.3.1, Cython 0.22, Six 1.9.0, Virtualenv 13.0.3, pandas 0.16.2, h5py 2.5.0 (serial/parallel), Matplotlib 1.4.3, pyCuda 2015.1, netcdf4 1.1.8
- Example Easyconfig files (for Python 2.7.10 on Cray)
  - Python-2.7.10-CrayGNU-5.2.40.eb
  - matplotlib-1.4.3-CrayGNU-5.2.40-Python-2.7.10.eb
  - netcdf4-python-1.1.8-CrayGNU-5.2.40-Python-2.7.10.eb
  - h5py-2.5.0-CrayGNU-5.2.40-Python-2.7.10-parallel.eb
  - h5py-2.5.0-CrayGNU-5.2.40-Python-2.7.10-serial.eb
  - pycuda-2015.1-CrayGNU-5.2.40-Python-2.7.10.eb
- Easyblocks
  - h5py.py, netcdf\_python.py, pycuda.py

Now available on:

- Daint, Dora, Santis, Brisi (CrayGNU)
- Pilatus, Castor (foss)
- **Escha, Kesch (gmvolf) \*new\***

# MCH CS-Storm use case

- Autoconf/2.69
- Automake/1.15
- Autotools/20150215
- binutils/2.25
- Bison/3.0.3
- Boost/1.49.0
- bzip2/1.0.6
- CDO/1.6.9
- CMake/3.2.2
- Cube/4.3.2
- cURL/7.40.0
- ddt/5.0(default)
- Doxygen/1.8.9.1
- FFTW/3.3.4
- flex/2.5.39
- freetype/2.5.5
- GCC/4.8.2
- gettext/0.18.2
- GLib/2.34.3
- gmvapich2/2015a
- gmvolf/2015a
- GSL/1.16
- HDF/4.2.8
- HDF5/1.8.15
- JasPer/1.900.1
- Java/1.7.0\_80
- libffi/3.0.13
- libjpeg-turbo/1.4.0
- libpng/1.6.16
- libreadline/6.3
- libtool/2.4.6
- libxml2/2.9.1
- M4/1.4.17
- matplotlib/1.4.3
- MVAPICH2/2.0.1\_gnu48
- NASM/2.11.06
- NCO/4.5.1
- ncurses/5.9
- ncview/2.1.5
- netCDF/4.3.3.1
- netCDF-Fortran/4.4.2
- netcdf-python/1.1.8
- OPARI2/1.1.4
- OpenBLAS/0.2.13
- OTF2/1.5.1
- Python/2.7.10
- R/3.1.3
- Ruby/2.2.2
- ScaLAPACK/2.0.2
- Scalasca/2.2.2
- Score-P/1.4.2
- SQLite/3.8.8.1
- Szip/2.1
- Tcl/8.6.3
- UDUNITS/2.1.24
- zlib/1.2.8



# Jenkins

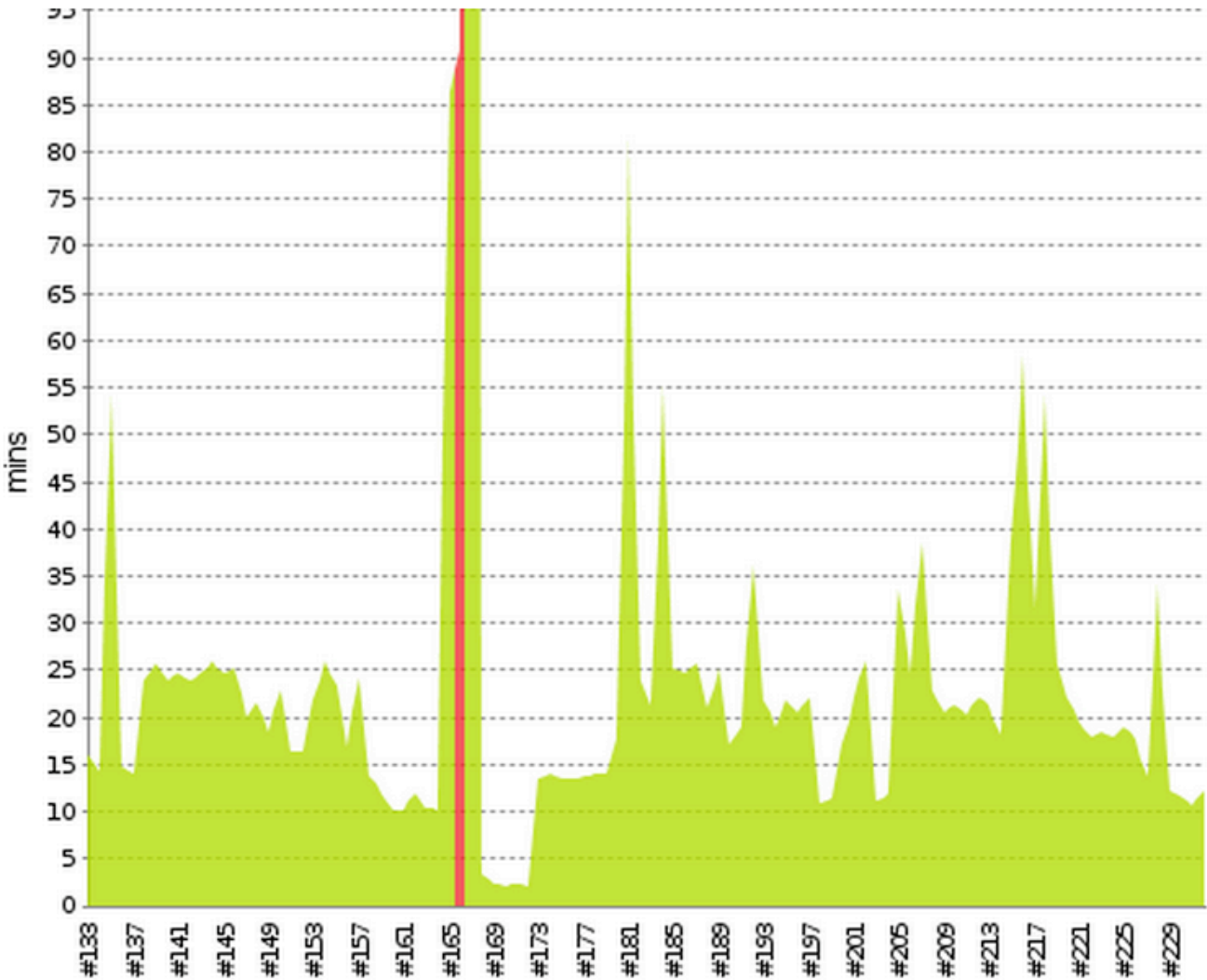
- Jenkins is a tool designed for continuous integration/validation
- But it is much more powerful than that
  - Thousands of plugins are available
  - Can be easily configured to run tasks by ssh anywhere
  - You get logs for all of your executions for free
  - Info for running / past jobs and logs are always accessible through the web interface
- Some usage examples:
  - Development/Integration:
    - Checkout svn/git repositories to automatically build on different platforms
  - Validation
    - Periodically run unit tests
  - Monitoring
    - Periodically run sanity and performance tests (**\*regression\***)
  - Run your favorite script or app
    - Use your creativity (example at CSCS: driving the acceptance of MCH machine)

# Jenkins example: Monitoring scratch performance for apps (netcdf5)



## Build Time Trend

By [lucamar™](#)




Build	↑	Duration	Slave
<a href="#">#2</a>		15 min	master
<a href="#">#3</a>		16 min	master
<a href="#">#4</a>		28 min	master
<a href="#">#5</a>		30 min	master
<a href="#">#6</a>		22 min	master
<a href="#">#7</a>		20 min	master
<a href="#">#8</a>		20 min	master
<a href="#">#9</a>		20 min	master
<a href="#">#10</a>		19 min	master
<a href="#">#11</a>		17 min	master
<a href="#">#12</a>		19 min	master
<a href="#">#13</a>		18 min	master
<a href="#">#14</a>		24 min	master
<a href="#">#15</a>		18 min	master
<a href="#">#16</a>		12 min	master
<a href="#">#17</a>		11 min	master
<a href="#">#18</a>		29 min	master
<a href="#">#19</a>		39 min	master
<a href="#">#20</a>		10 min	master

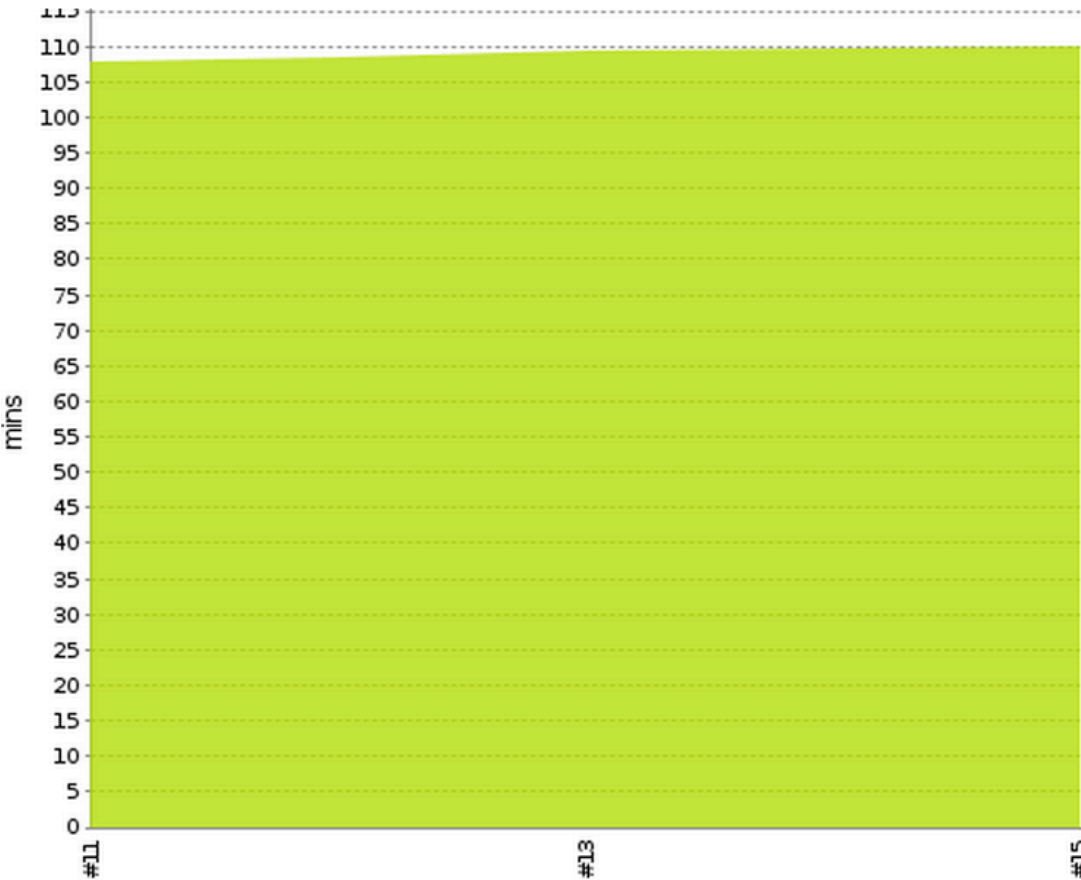


# Jenkins example: Rebuilding all software stack for Escha/Kesch

S	W	Name ↓	Last Success	Last Failure	Last Duration	
		<a href="#">RegressionEBKesch</a>	20 hr - <a href="#">#15</a>	N/A	1 hr 49 min	

## Build Time Trend

	Build ↑	Duration	Slave
	<a href="#">#11</a>	1 hr 47 min	master
	<a href="#">#13</a>	1 hr 49 min	master
	<a href="#">#15</a>	1 hr 49 min	master



# Jenkins + EB integration: example workflow for testing .eb files

- Testing new easyconfig files on all machines where the toolchain is available
- Workflow setup
  1. Create a folder accessible by jenscscs to store the .eb files
    - /path/to/eb-files/
  2. Create a jenkins project adding the target test systems
    - CrayGNU/5.2.40 = daint, dora, santis, brisi
    - Foss/2015a = castor, pilatus
  3. Add the following commands to the “Execute shell”
    - `source /apps/common/easybuild/setup.sh`
    - `find /path/to/eb-files/ -name '*CrayGNU-5.2.40*.eb' -exec eb {} "-r -f" \;`
      - (Foss/2015a: replace “\*CrayGNU-5.2.40\*” by “\*foss-2015a\*”
- Usage
  1. Copy .eb files to /path/to/eb-files/
  2. Go to Jenkins and click on “Build now”

# Jenkins: Example for testing .eb files

- /apps/common/tools/easybuild/jenkins/

- CrayGNU/5.2.40

- CDO-1.6.9-CrayGNU-5.2.40.eb
- NFFT-3.3.0-CrayGNU-5.2.40.eb



- Foss/2015a

- Ghostscript-9.10-foss-2015a.eb
- HDF5-1.8.15-foss-2015a.eb

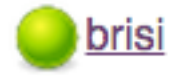


# Jenkins: Example for testing .eb files

- /apps/common/tools/easybuild/jenkins/

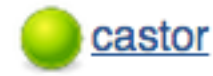
- CrayGNU/5.2.40

- CDO-1.6.9-CrayGNU-5.2.40.eb
- NFFT-3.3.0-CrayGNU-5.2.40.eb



- Foss/2015a

- Ghostscript-9.10-foss-2015a.eb
- HDF5-1.8.15-foss-2015a.eb



Red ball = tomato FAIL

# Final thoughts

- Current installation is ready for application level
  - Validation with Python use case (including modules)
    - Daint, Dora, Santis, Brisi, Pilatus, Castor **and Escha/Kesch (new)**
- Continuous validation techniques can be easily applied
  - Testing builds across all systems with Jenkins
  - Changes/errors on the PrgEnv can be detected early
- In order to get the most out of EasyBuild
  - We need to have consistent PrgEnv on most systems
    - OK on Cray systems
    - Not currently true on non-Cray
      - Achievable with EasyBuild

## Next steps (SCS)

- Try out EB for answering tickets requesting new software
  - Testing and feedback are very welcome
  - Can also be used to answer individual user requests
    - Builds that won't be officially supported
    - Such as the famous RT ticket “#19610: nano editor for dora”
- Agree on toolchains for non-Cray systems
  - Stock toolchain (foss + intel for example)
    - Default “foss” toolchain works just fine for Python use case
    - May be not optimal on all archs (for example concerning MPI)
  - Tailored toolchain using existing PrgEnv (supported by HPC Operations team)
    - This approach was used on the new Storm MCH (gmvolf)
      - System GCC+MPI were used
    - We might end up with a different toolchain on each system



# Links

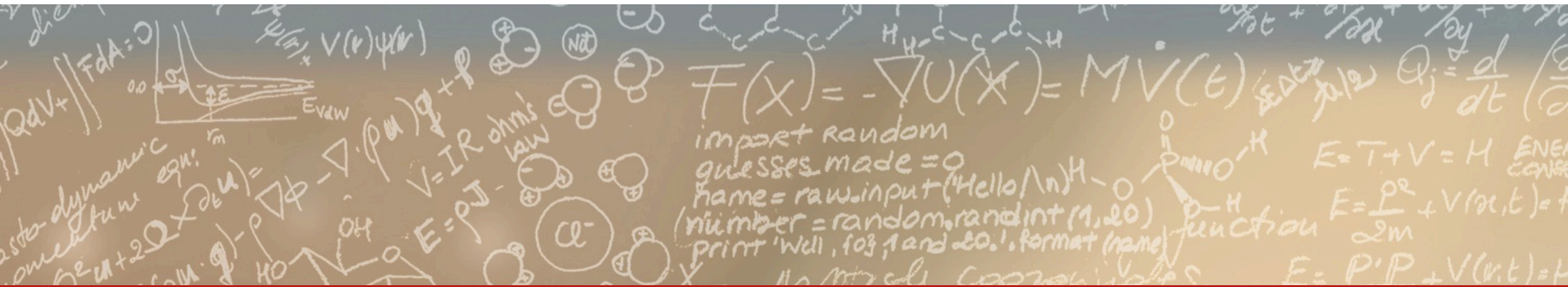
- Easybuild Documentation
  - GitHub
    - <https://github.com/hpcugent/easybuild>
  - Workflow example (WRF)
    - [http://easybuild.readthedocs.org/en/latest/Typical\\_workflow\\_example\\_with\\_WRF.html](http://easybuild.readthedocs.org/en/latest/Typical_workflow_example_with_WRF.html)
- CSCS Internal doc
  - <https://github.com/eth-cscs/tools/wiki/EasyBuild-at-CSCS>
- Additional easyconfig files repositories
  - Development EasyBuild branch
    - <https://github.com/hpcugent/easybuild-easyconfigs/tree/develop>
  - Successful production builds at CSCS
    - [https://github.com/eth-cscs/tools/tree/master/easybuild/ebfiles\\_repo](https://github.com/eth-cscs/tools/tree/master/easybuild/ebfiles_repo)



**CSCS**

Centro Svizzero di Calcolo Scientifico  
Swiss National Supercomputing Centre

**ETH** zürich



**Thank you for your attention.**