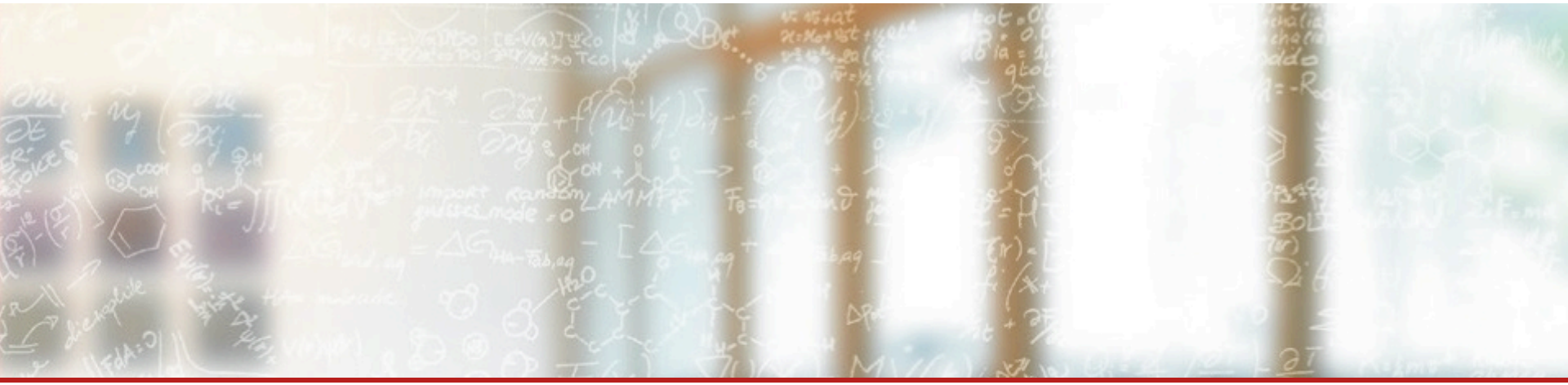
CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

ETH zürich

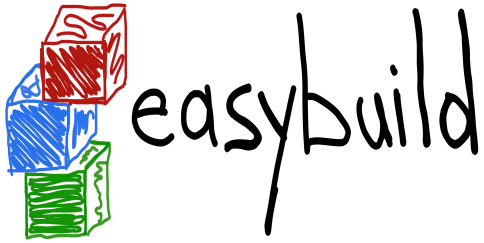


Reproducible HPC Software Installation on Cray Systems with EasyBuild

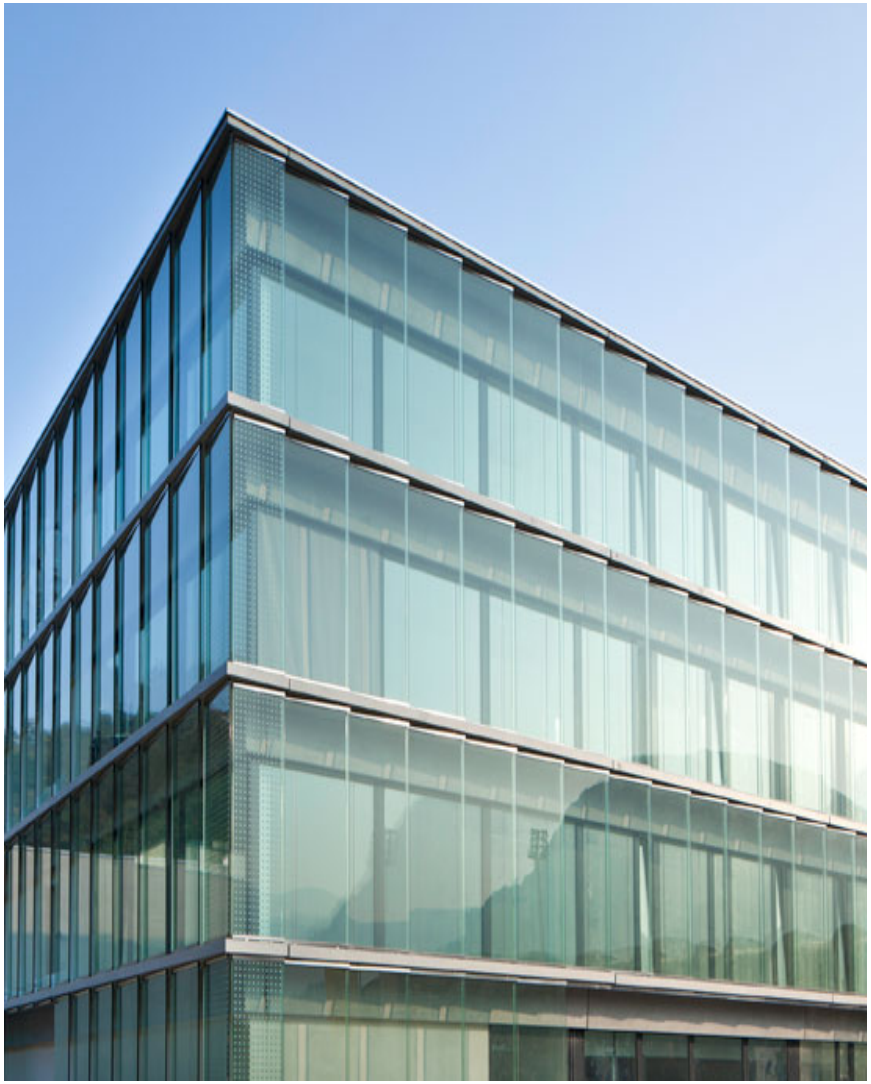
FOSDEM'17

Guilherme Peretti-Pezzi (Swiss National Supercomputing Centre - CSCS)
Petar Forai (Research Institute of Molecular Pathology - IMP)
Kenneth Hoste (Ghent University – UGent)

February 04, 2017



Outline



- **Background**
  - **Installing HPC software + EasyBuild**
- EB + Cray programming environment
  - External metadata / modules
- EB @ CSCS
  - Piz Kesch & Escha use case
    - Cray CS-Storm
  - Piz Daint use case
    - Cray XC
  - Github production repository and CI

The Ubiquitous Burden of Getting Scientific Software Installed

- One particularly time-consuming task for HPC support teams is installing software on (Cray) systems.
- Science teams demand rich and up to date software environment to be provided on the system(s).
- Not simple to do because
  - Packaged (distribution supplied) software is often out of date and a clear upgrade path is not provided by Cray during the system's life cycle.
  - Scientific software installation is non-trivial in itself.

Common issues with scientific software

Researchers focus on the *science* behind the software they implement, and care little about tools, build procedure, portability, …

Scientists are not software developers or sysadmins (nor should they be).

"*If we would know what we are doing, it wouldn't be called 'research'.*"

- This results in:
  - "incorrect" use of build tools
  - use of non-standard build tools (or broken ones)
  - incomplete build procedure, e.g., no configure or install step
  - interactive installation scripts
  - hardcoded parameters (compilers, libraries, paths, …)
  - poor/outdated/missing/incorrect documentation
  - dependency (version) hell

Example: WRF

Weather Research and Forecasting Model (http://www.wrf-model.org)

- Dozen dependencies: netCDF (C, Fortran), HDF5, tcsh, JasPer, . . .
- Interactive ‘configure’ script :(
- Resulting configure.wrf needs work:
  - fix hardcoded settings (compilers, libraries, . . .), tweaking of options
- Custom “compile” script (wraps around `make')
- Building in parallel is broken
- No actual installation step

Wouldn't it be nice to build & install WRF with a single command?

http://easybuild.readthedocs.org/en/latest/Typical_workflow_example_with_WRF.html

Houston, we have a problem

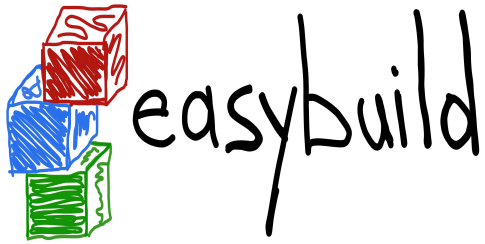
Installation of scientific software is a tremendous problem for HPC sites all around the world.

- huge burden on HPC user support teams.
- researchers lose lots of time (waiting).
- sites typically resort to in-house scripting (or worse).
- very little collaboration among HPC sites :(
- especially hard to reproduce builds at later point in time.

Also true on Cray systems, despite the extensive programming environment that Cray provides and similarity of software environment across installations.

EasyBuild: building software with ease

http://hpcugent.github.io/easybuild/

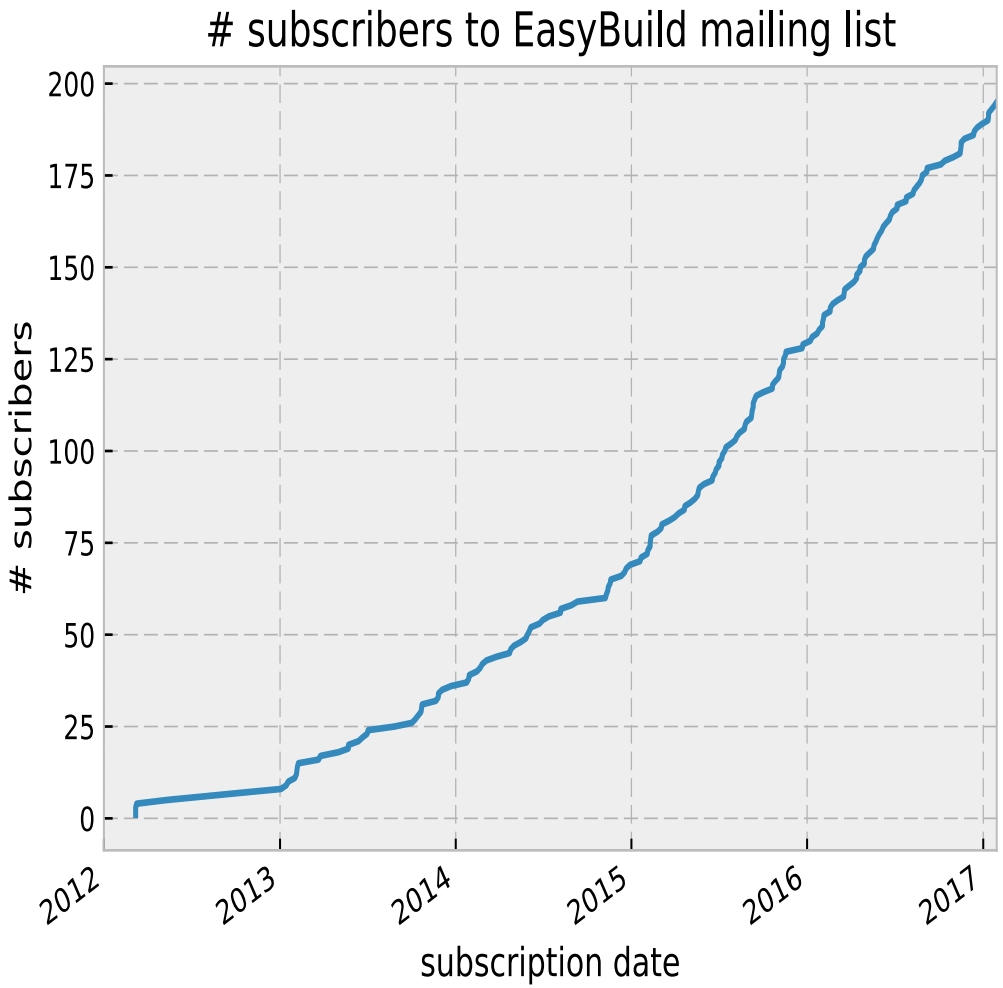


- framework for installing (scientific) software on HPC systems
- implemented as Python packages and modules
- started at University of Gent, Belgium, in 2009, open-source (GPLv2) since 2012
- now: thriving community; actively contributing, driving development
- new release every 6-8 weeks (latest: EasyBuild v3.1.0, Feb 2nd 2017)
- supports over 1.100 different software packages
- including CP2K, GAMESS-US, GROMACS, NAMD, NWChem, OpenFOAM, PETSc, QuantumESPRESSO, WRF, WPS, . . .
- well documented: http://easybuild.readthedocs.io

EasyBuild: feature highlights

- Fully **autonomously** building and installing (not only scientific) software
  - automatic dependency resolution.
  - full integration with Environment Modules (Tcl or Lua syntax)
- thorough **logging** of executed build/install procedure
- **archiving** of build specifications (‘easyconfig files’)
- highly **configurable**, via config files/environment/command line
- **dynamically extendable** with additional easyblocks, toolchains, etc.
- **reproducibility** of the installation as one of the major design goals
- **comprehensively tested**: lots of unit tests, regression testing, …
- actively developed, **collaboration** between various HPC sites (worldwide **community**)
- extensive **transparency** through verbose dry-runs and preview of all steps involved in the installation.

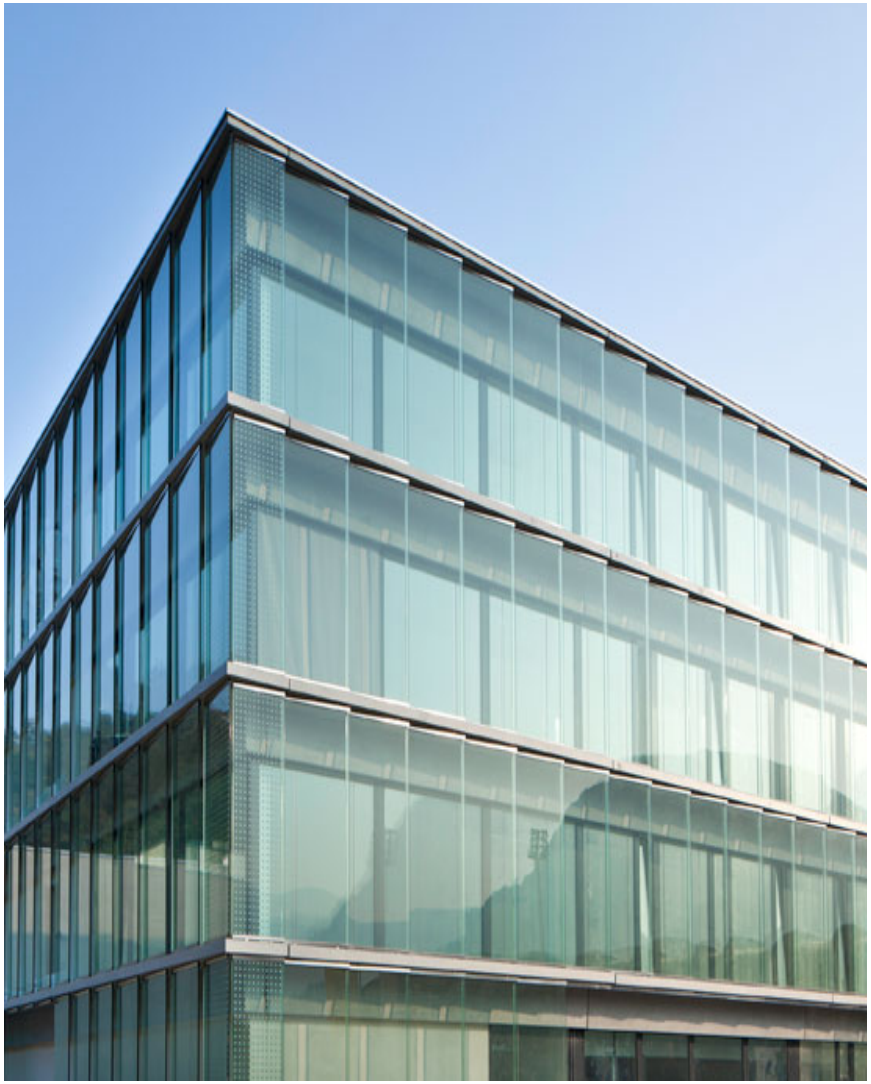
EasyBuild mailing list



EasyBuild terminology

- EasyBuild **Framework**
  - core of EasyBuild: Python modules & packages
  - provides supporting functionality for building and installing software
- **easyblock**
  - a Python module, 'plugin' for the EasyBuild framework
  - implements a (generic) software build/install procedure
- **easyconfig** file (*.eb)
  - build specification: software name/version, compiler toolchain, etc.
- compiler **toolchain**
  - compilers with accompanying libraries (MPI, BLAS/LAPACK, …)

Outline



- Background
  - Installing HPC software + EasyBuild
- **EB + Cray programming environment**
  - **External metadata / modules**
- EB @ CSCS
  - Piz Kesch & Escha use case
    - Cray CS-Storm
  - Piz Daint use case
    - Cray XC
  - Github production repository and CI

EasyBuild terminology & example

GMP-6.1.1-foss-2016.04.eb

Toolchain: foss/2016.04

foss-2016.04.eb

- GCC/5.3.0 (binutils/2.26)
- OpenMPI/1.10.2
- OpenBLAS/0.2.18
- FFTW/3.3.4
- ScaLAPACK/2.0.2 (LAPACK-3.6.0)

Easyblock: generic/ConfigureMake

ConfigureMake.py

```
def configure_step(self, cmd_prefix=''):

def build_step(self, verbose=False,
path=None):

def install_step(self):
```

$ eb GMP-6.1.1-foss-2016.04.eb -r

Software stack on a Cray XC

For example PE 2016.11 [1]

- Cray Compiling Environment - CCE 8.5.5
- Cray Message Passing Toolkit - MPT 7.5.0
- Perftools 6.4.3
- Cray Scientific and Math Libraries - CSML
  - LibSci 16.11.1
  - LibSci_ACC 16.11.1
  - PETSc 3.7.2.1
  - Trilinos 12.6.3.3
  - TPSL 16.07.1
  - FFTW 3.3.4.10

CrayGNU-2016.11.eb

PrgEnv-gnu
- gcc/5.3.0
- cray-mpich/7.5.0
- cray-libsci/16.11.1 (BLAS, LAPACK, ScaLAPACK, BLACS)

foss-2016.04.eb

- GCC/5.3.0 (binutils/2.26)
- OpenMPI/1.10.2
- OpenBLAS/0.2.18
- ScaLAPACK/2.0.2 (LAPACK-3.6.0)
- FFTW/3.3.4

- [1] http://docs.cray.com/books/S-9408-1611/

EasyBuild Enhancements for Cray Systems

- Support for external module files
- Definition of Cray-specific toolchains
- Custom easyblock for Cray toolchains
- Various smaller enhancements specific to the Cray environment

- **Thanks to Peter Forai & Kenneth Hoste**

Support for external module files

EasyBuild relies on the (environment) modules in a fundamental way as they contain information about the installed software they correspond to.

- EasyBuild can now leverage modules that were not generated by EasyBuild for example as part of the Cray PE.

- this includes support that was added to supply metadata for external modules, so that EasyBuild can be made aware of
  - the software name(s), version(s) and installation prefix

- since EasyBuild version 2.7.0, a file containing metadata for selected modules provided by the Cray PE is included as part of EasyBuild.

Custom easyblock for Cray toolchains

- This easyblock defines the **version pinned** components that make up the toolchain

- Easyblock implements logic to render the module files for the EasyBuild Cray toolchains
  - ensures that switching toolchain components works
  - avoids the need to run *module purge*

- New toolchain version combinations can then be placed in easyconfig file to ease creation of new toolchains.

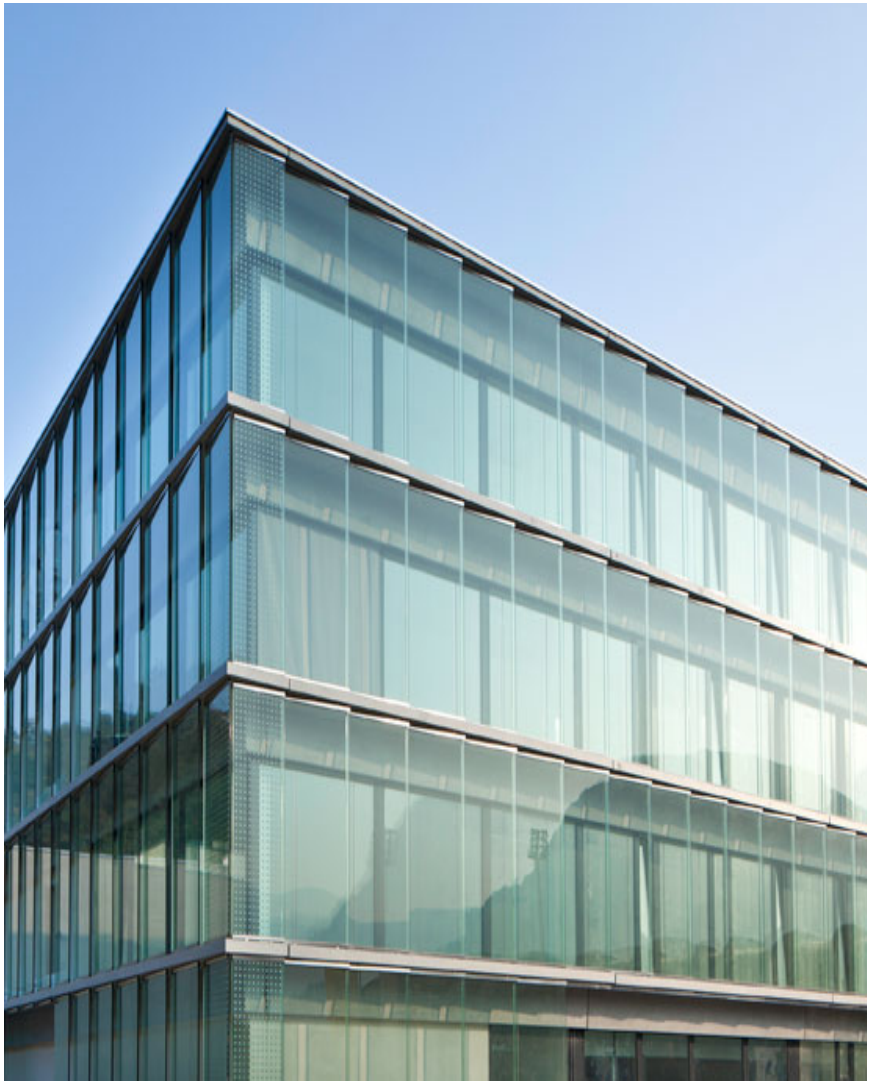
Definition of Cray-specific Toolchains

Cray-specific toolchains have been implemented for each PrgEnv module:

- CrayCCE for PrgEnv-cray
- CrayGNU for PrgEnv-gnu
- CrayIntel for PrgEnv-intel
- CrayPGI for PrgEnv-pgi

- Compiler component of the toolchains EasyBuild leverages the compiler wrappers provided by the Cray PE and EasyBuild exposes
  - $CC, $CXX, $CFLAGS, $CXXFLAGS, $F77

Outline



- Background
  - Installing HPC software + EasyBuild
- EB + Cray programming environment
  - External metadata / modules
- **EB @ CSCS**
  - **Piz Kesch & Escha use case**
    - **Cray CS-Storm**
  - **Piz Daint use case**
    - **Cray XC**
  - **Github production repository and CI**

Piz Kesch & Escha use case (MeteoSwiss / Cray CS-Storm)

"Kesch" and "Es-cha" consist of identical systems (production and failover), each comprising:

Cray CS-Storm: 12 nodes

- 2 x Intel Haswell E5-2690v3 2.6 GHz 12-core CPUs per node
  - total of 24 E5-2690v3 processors
- 256 GB 2133 MHz DDR4 memory per node
  - total of 3 TB
- 8 NVIDIA® Tesla® K80 GPU devices per node
  - total of 192 GPUs

MeteoSwiss, the Swiss national weather forecasting service, hosts their dedicated production systems at Cray CS-Storm at CSCS, Lugano.

Piz Kesch & Escha use case (MeteoSwiss / Cray CS-Storm)

- Cray PE is partially supported on the CS-Storm series:
  - PrgEnv-cray is available but not GNU or Intel
- System provided GCC-based compiler stack was unable to assemble optimized (AVX2) instructions for system's Intel Haswell processor.
  - GNU binutils was too old
- While waiting for a definitive fix from Cray…
  - The whole software stack required by MeteoSwiss was deployed using EasyBuild
    - Using a standard open source toolchain (gmvolf)
- EasyBuild software stack now in production since September/2015

Piz Daint

Model	Cray XC50/XC40
XC50 Compute Nodes	Intel® Xeon® E5-2690 v3 (Haswell) @ 2.60GHz (12 cores, 64GB RAM) and NVIDIA® Tesla® P100 16GB
XC40 Compute Nodes	Intel® Xeon® E5-2695 v4 (Broadwell) @ 2.10GHz (18 cores, 64/128 GB RAM)
Login Nodes	Intel® Xeon® CPU E5-2650 v3 @ 2.30GHz (10 cores, 256 GB RAM)
Interconnect Configuration	Aries routing and communications ASIC, and Dragonfly network topology
Scratch capacity	6.2 PB (Luster / Sonexion 3000)

Piz Daint

- #8 Top 500
  - #1 in Europe
  - 9,779.0 PFLOPS
- #2 Green 500
  - 7453.5 MFLOPS/W

Available software on Cray using EasyBuild

- Stock EasyBuild repository
  - Python, including
    - accelerated numpy + scipy, h5py, …
  - WRF
  - CP2K[*]
  - GROMACS[*]
  - Boost
  - GSL

- CSCS Production Github repository
  - Amber[*]
  - CDO
  - CPMD
  - LAMMPS[*]
  - NCL
  - NCO
  - ParaView[*]
  - Octave
  - QuantumESPRESSO[*]
  - R
  - Scalasca
  - ScoreP
  - TensorFlow[*]
  - VASP[*]
  - Visit
  - VMD[*]
  - VTK[*]

[*] = GPU-enabled recipe available

Github, EasyBuild & Continuous Integration

- Github repository hosting all recipes in production on Piz Daint
  - https://github.com/eth-cscs/production
  - Recipes go through a reviewing process (standard PR procedure)

- Automatic checking of build recipes on Piz Daint (by Jenkins)
  - GitHub Pull Request Builder Plugin for Jenkins
    - https://github.com/janinko/ghprb
  - Less error-prone & improved reproducibility
    - Robot ensures that recipes work without any extra tweaking
      - Such as exports and custom .bashrc files

- Autonomous deployment of software/modules on production (Jenkins)
  - List of easyconfig files is automatically deployed by Jenkins
  - For example, list of GPU-enabled software stack for the P100 partition
    - https://github.com/eth-cscs/production/blob/master/jenkins-builds/6.0.UP02-2016.11-gpu

Final comments: EasyBuild & Cray

- Proprietary and FOSS can co-exist

- Best of two worlds
  - Integrates new applications with optimized proprietary stack
  - Support is assured by Cray and also by the enthusiasts of the EB community

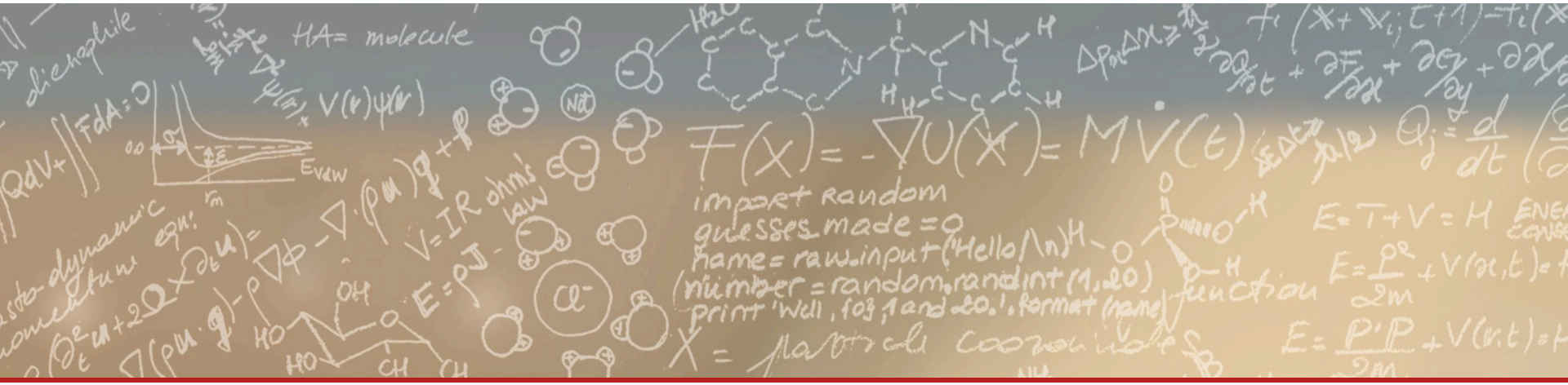
- Minimizes risks of vendor lock-in
  - EasyBuild provides alternatives in case of issues with software provided by Cray

Do you want to know more?

- Paper on the Cray User Group 2016
  - **Making Scientific Software Installation Reproducible On Cray Systems Using EasyBuild**
  - https://cug.org/proceedings/cug2016_proceedings/includes/files/pap145.pdf
- CSCS Website: http://www.cscs.ch
- CSCS Production Repository: https://github.com/eth-cscs/production

- EasyBuild website: http://hpcugent.github.io/easybuild
- EasyBuild documentation: http://easybuild.readthedocs.org
- Stable EasyBuild releases: http://pypi.python.org/pypi/easybuild

- EasyBuild mailing list: easybuild@lists.ugent.be - https://lists.ugent.be/wws/subscribe/easybuild
- Twitter: http://twitter.com/easy_build
- IRC: #easybuild on chat.freenode.net



Thank you for your attention.