Online introductory tutorial

Tue June 23rd 2020

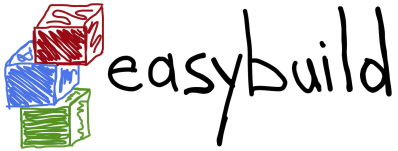
https://github.com/easybuilders/easybuild/wiki/EasyBuild-tutorial

Acknowledgements

Tutorial organisers:

- Maxime Boissonneault (Compute Canada)
- Markus Geimer (Jülich Supercomputing Centre, Germany)
- Kenneth Hoste (HPC-UGent, Belgium)
- Christian Kniep (AWS)
- Alan O'Cais (Jülich Supercomputing Centre, Germany)
- Åke Sandgren (Umeå University, Sweden)

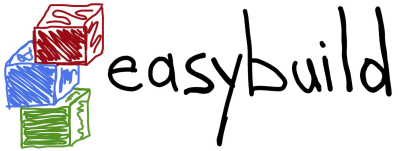
Acknowledgements



Reviewers & helping hands:

- Michael Kelsey (Texas A&M University, US)
- Terje Kvernes (University of Oslo, Norway)
- Miguel Dias Costa (National University of Singapore)

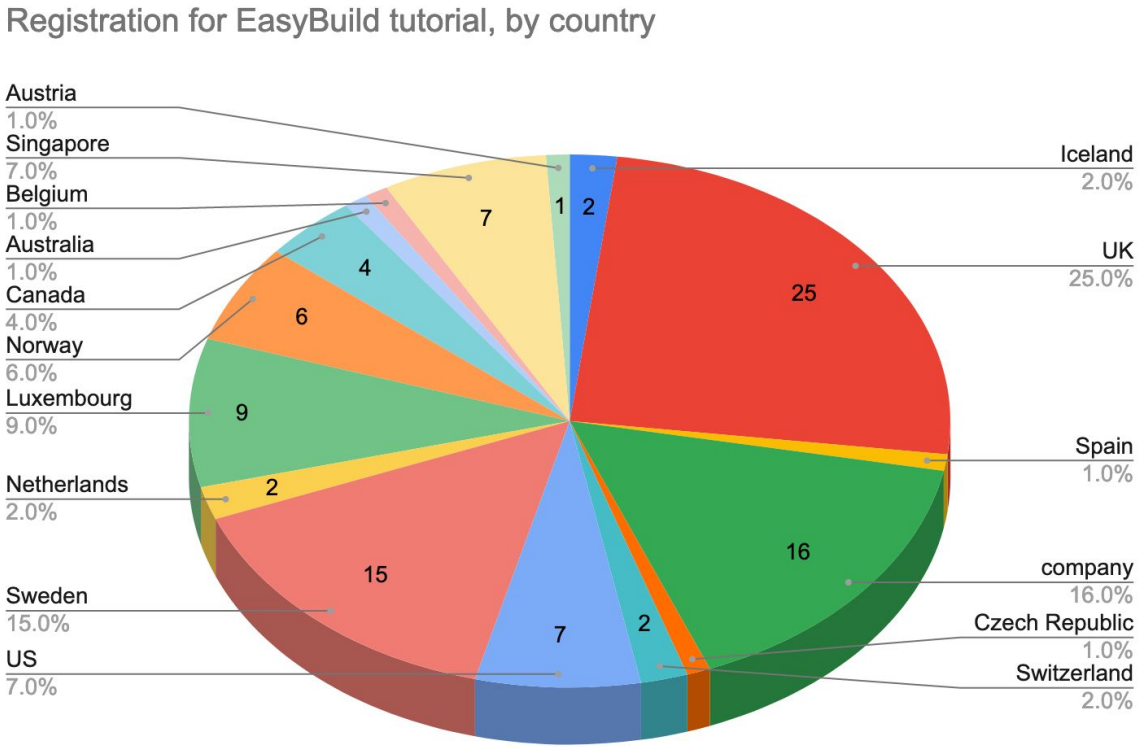
Background



- EasyBuild tutorial proposal accepted for ISC'20
- Accepted ISC'20 tutorials have been postponed to ISC'21
- We figured to seize the opportunity and host it online in 2020 as well...

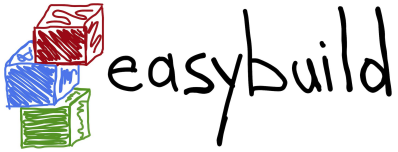
Attendance

- 101 registrations
- 7 companies
- 15 countries



Agenda (1/3)

(times are in UTC)



[11:00 - 11:10] Welcome & practical info

[11:10 - 11:25] General introduction to EasyBuild

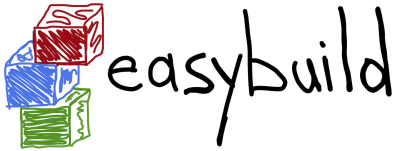
[11:25 - 11:50] Installation and configuration of EasyBuild (hands-on)

[11:50 - 12:30] Basic usage of EasyBuild + installing software (hands-on)

[12:30 - 12:40] *(short break)*

Agenda (2/3)

(times are in UTC)



[12:40 - 13:00] Troubleshooting (hands-on)

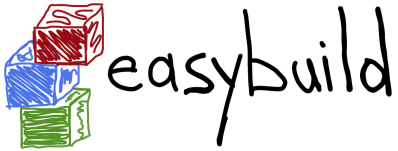
[13:00 - 13:20] Hierarchical module naming schemes

[13:20 - 14:00] Adding support for additional software (hands-on)

[14:00 - 14:10] *(short break)*

Agenda (3/3)

(times are in UTC)



[14:10 - 15:25] EasyBuild at the Jülich Supercomputing Centre

[14:25 - 15:40] EasyBuild at Compute Canada

[14:40 - 15:55] Contributing back to EasyBuild

[14:55 - 16:10] Comparison with other tools

[15:10 - 16:15] Getting help

[15:15 - 16:00] Q&A

Practical information

- Event page: https://github.com/easybuilders/easybuild/wiki/EasyBuild-Tutorial
- These slides: https://github.com/easybuilders/easybuild-tutorial/raw/master/docs/files/easybuild_tutorial_slides_isc20.pdf
- Tutorial site: https://easybuilders.github.io/easybuild-tutorial
- Streaming via YouTube: https://www.youtube.com/c/easybuilders
- Recordings will be available shortly after the live tutorial

Practical information: Slack

- Questions or problems?

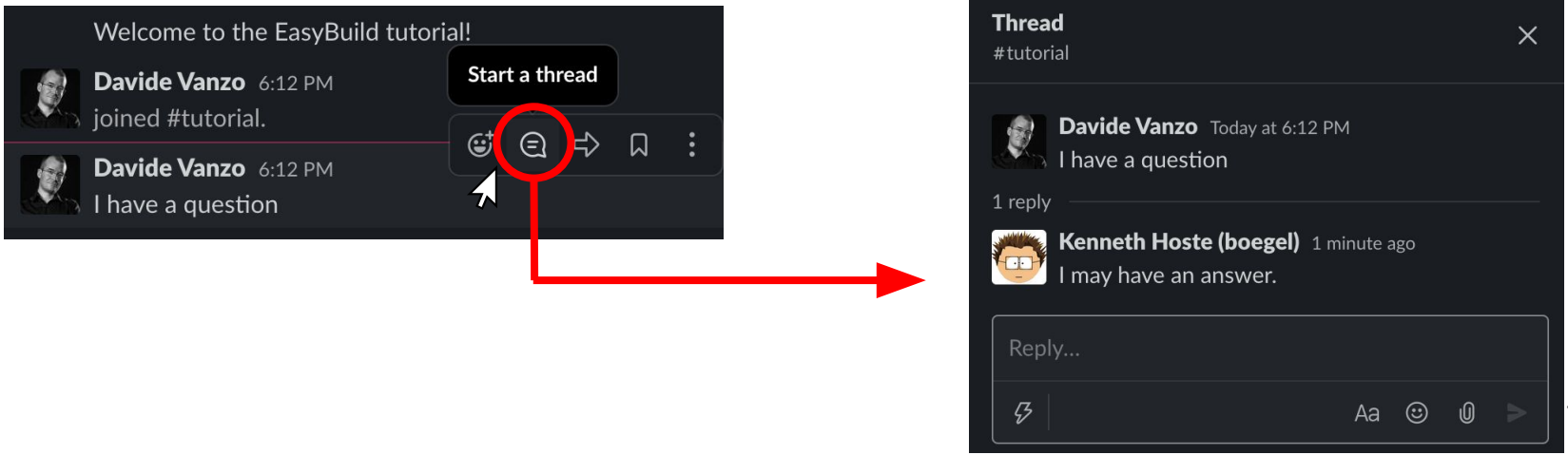
  Speak up in `#tutorial` on EasyBuild Slack!

Practical information: Slack

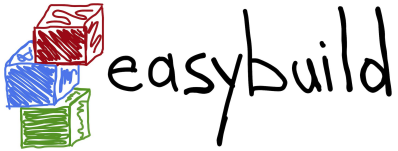
- Questions or problems?

  Speak up in `#tutorial` on EasyBuild Slack!

- **Use threads to avoid overflowing the channel!**



Prepared environment



- Docker container image (also usable via Singularity)
  - also usable via Singularity *(with some limitations)*
  - CentOS 7.8 + Lmod 8
  - Pre-installed software stack in `/easybuild`
  - Use `python3` to run EasyBuild

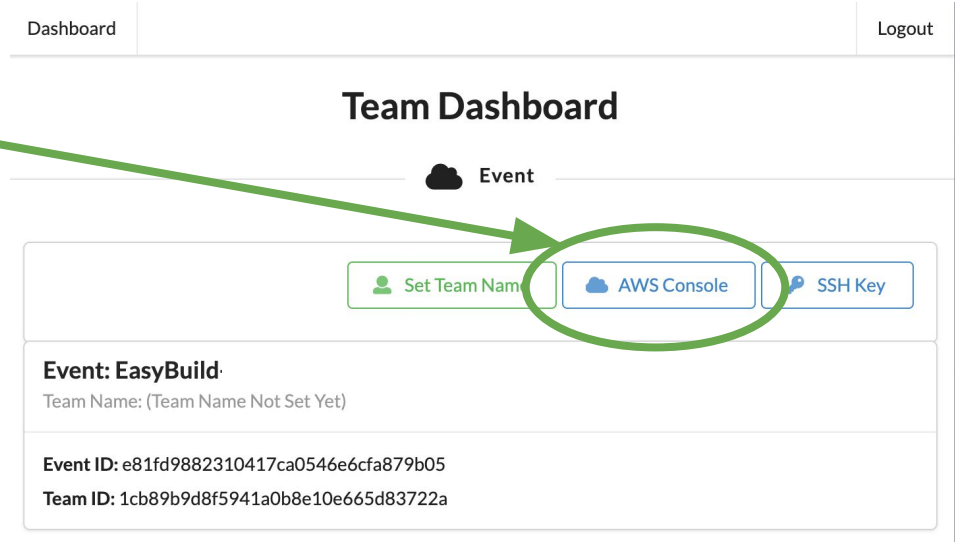
- Resources available in AWS Cloud9

Cloud9 login procedure

- **Use the login URL you received via email**
- Open AWS console
- Open IDE
- Search for Cloud 9
- Start a Terminal session

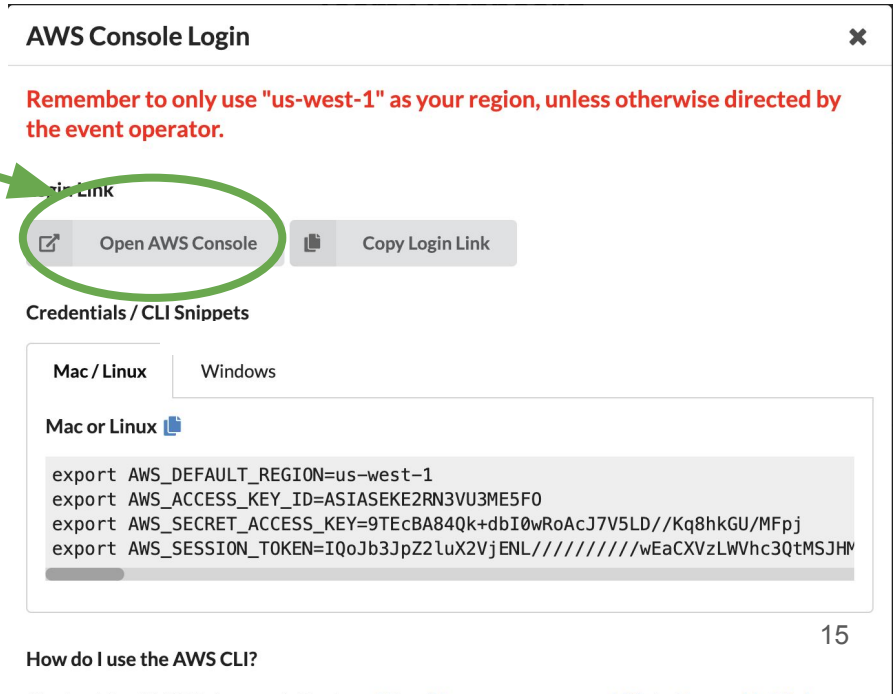
Cloud9 login procedure

- **Use the login URL you received via email**
- **Click “AWS console”**
- Open IDE
- Search for Cloud 9
- Start a Terminal session



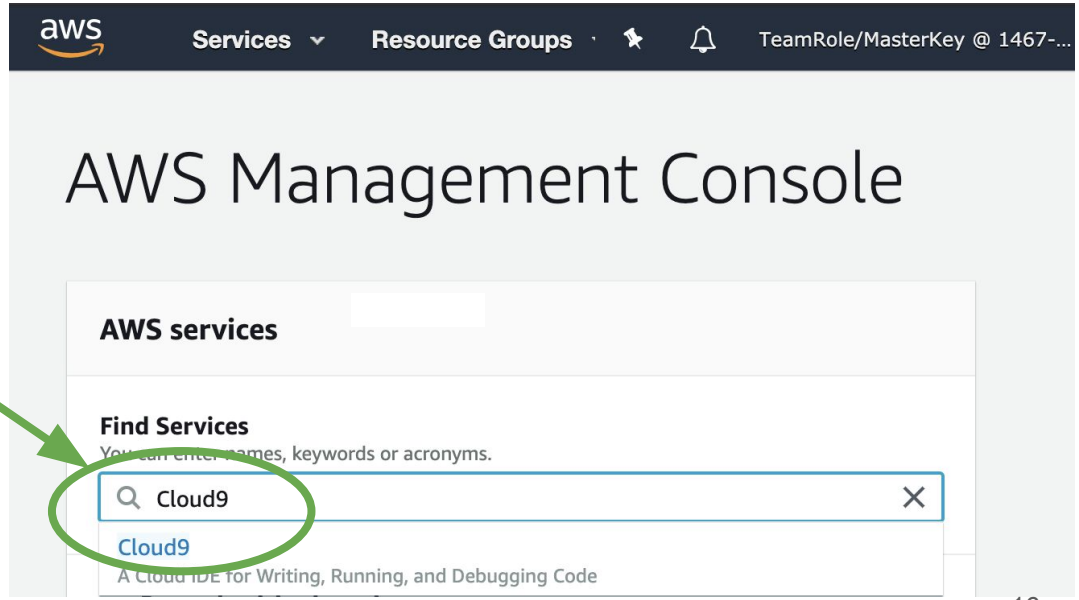
Cloud9 login procedure

- **Use the login URL you received via email**
- **Click “Open AWS console”**
- Open IDE
- Search for Cloud 9
- Start a Terminal session



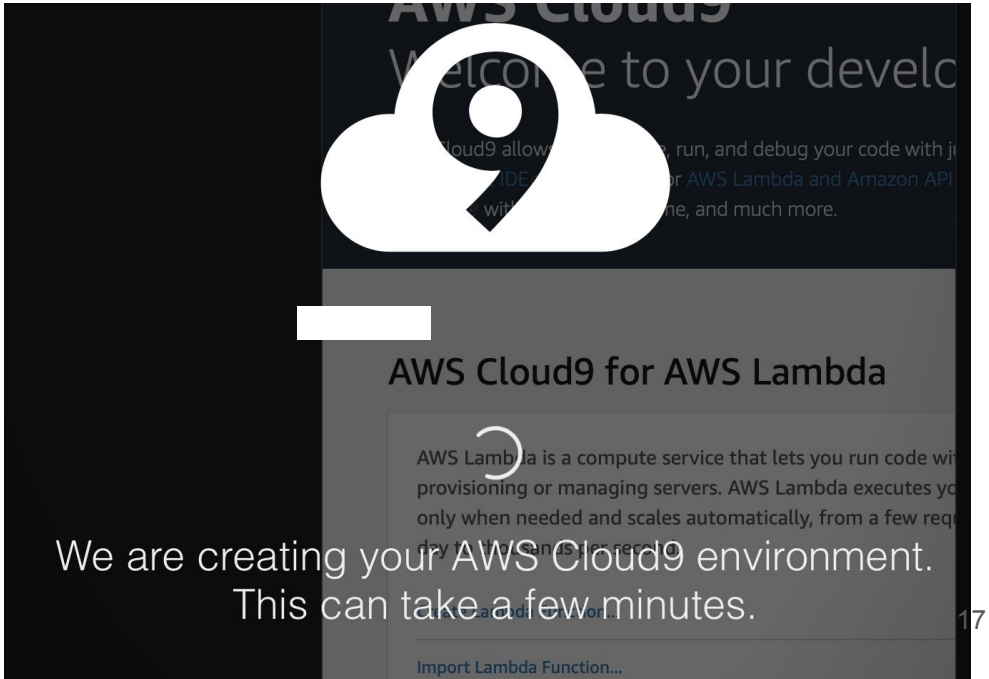
Cloud9 login procedure

- **Use the login URL you received via email**
- Open AWS console
- Open IDE
- **Search for Cloud 9**
- Start a Terminal session



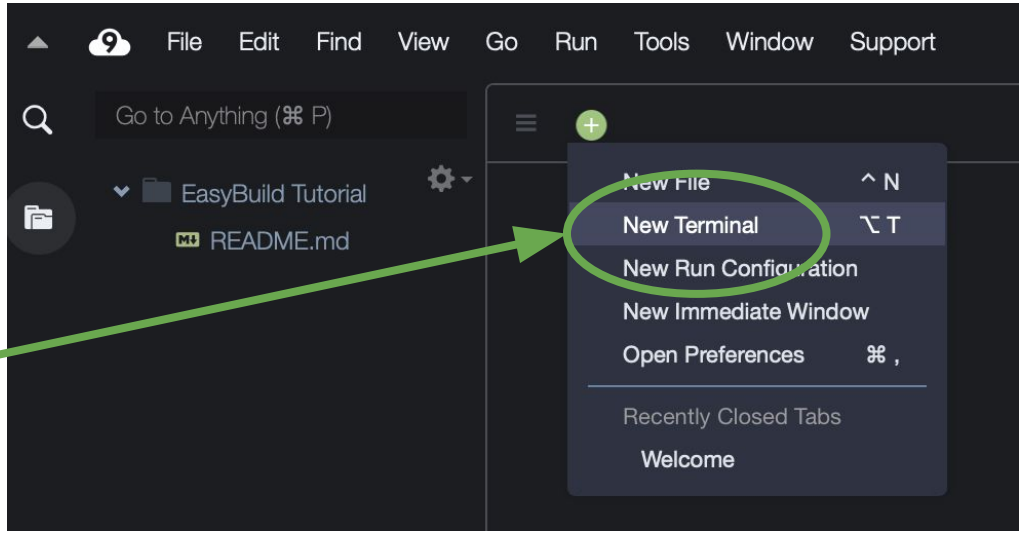
Cloud9 login procedure

- **Use the login URL you received via email**
- Open AWS console
- Open IDE
- **Search for Cloud 9**
- Start a Terminal session

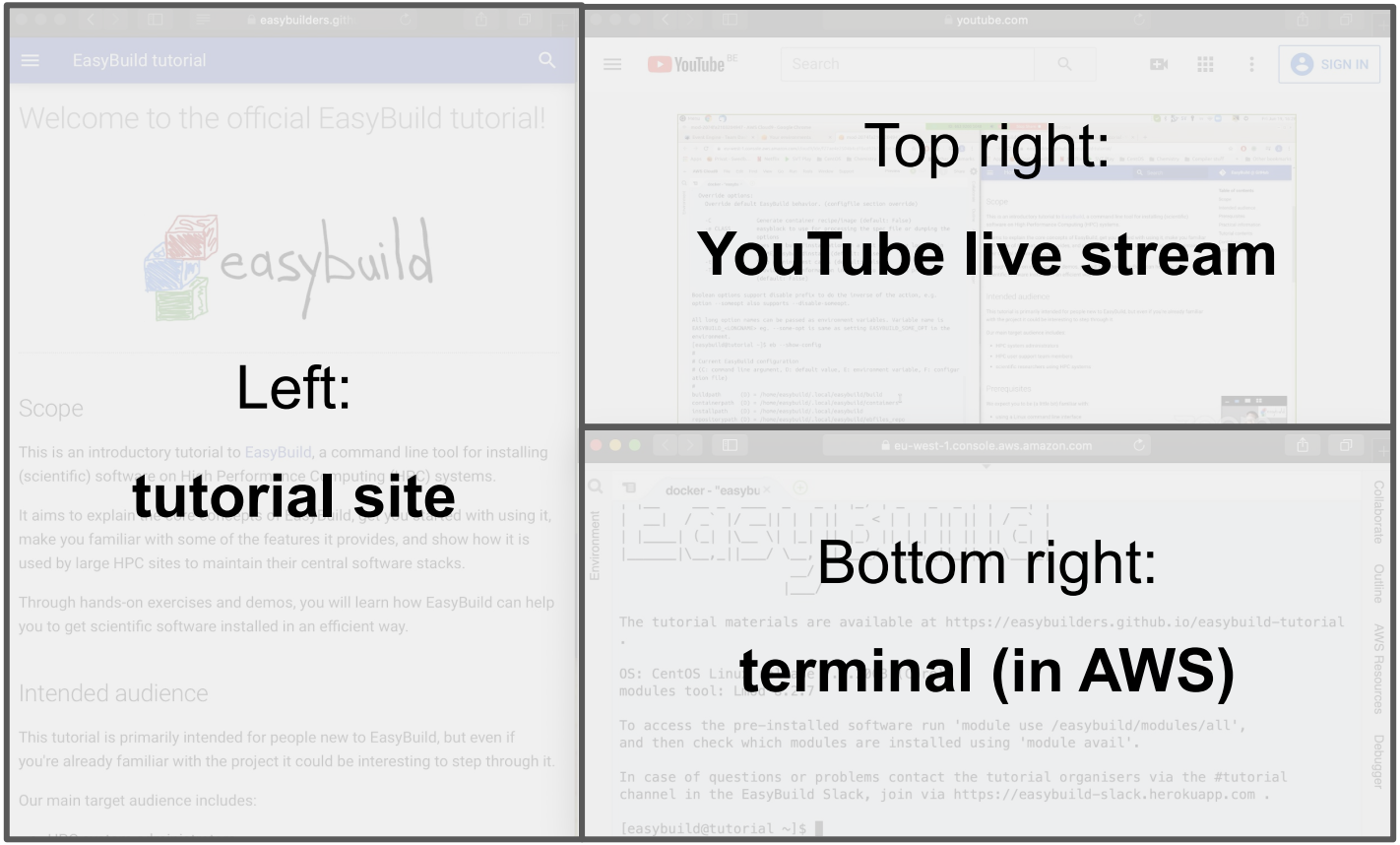


Cloud9 login procedure

- **Use the login URL you received via email**
- Open AWS console
- Open IDE
- Search for Cloud 9
- **Start a Terminal session**



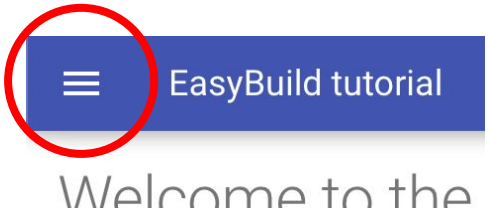
Recommended screen setup



Tutorial site

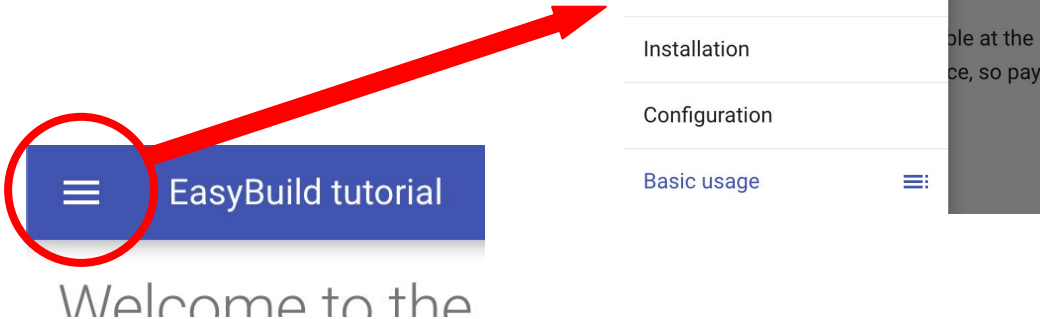
Hamburger button to

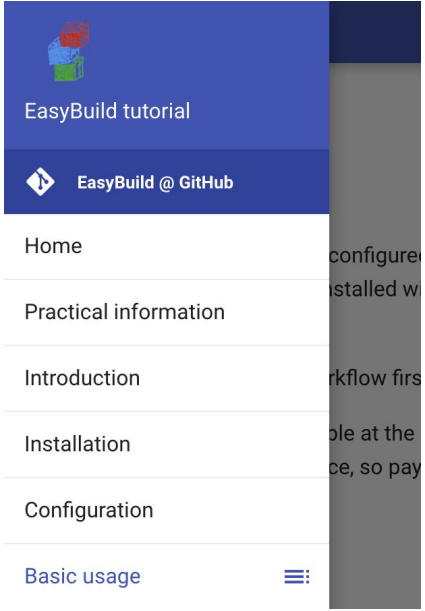
access tutorial overview



Tutorial site

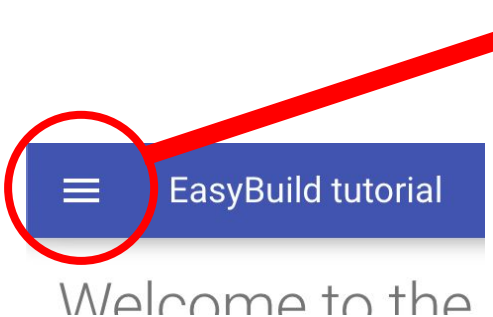
Hamburger button to access tutorial overview

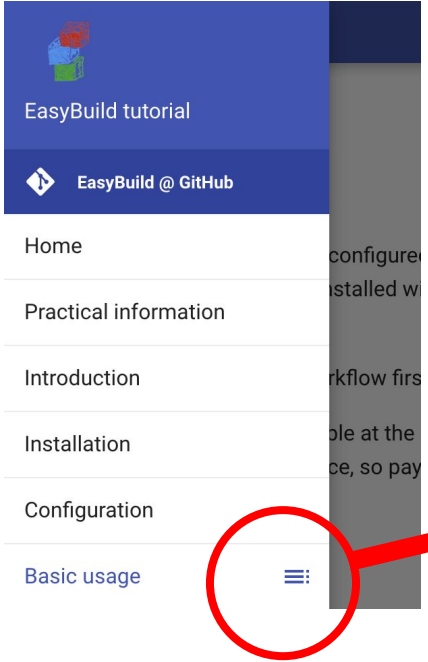


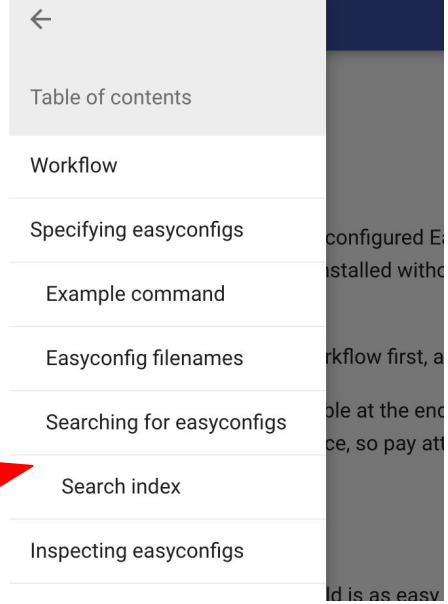


Tutorial site

Hamburger button to access tutorial overview



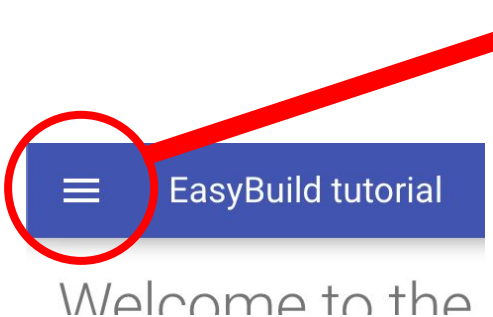


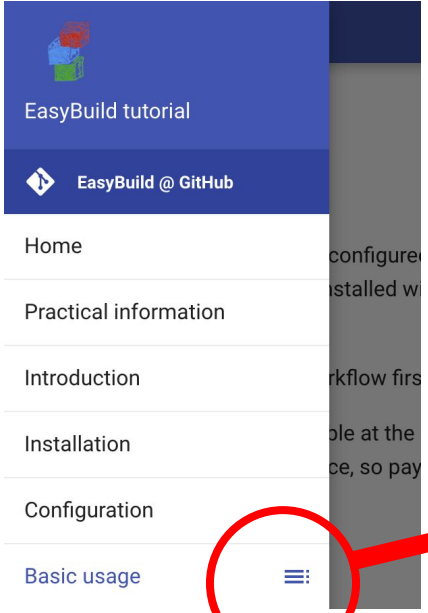


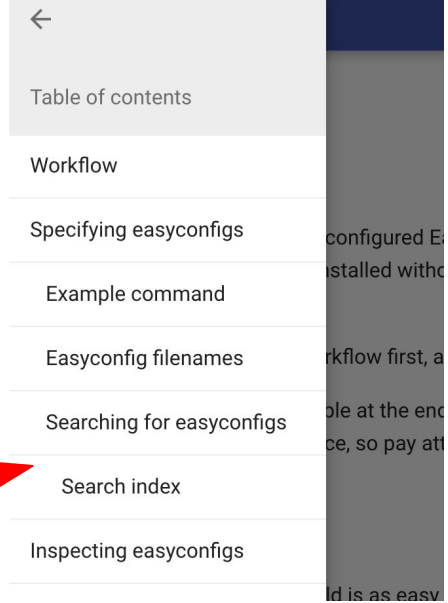
Mini hamburger button to access tutorial section contents

Tutorial site

Hamburger button to access tutorial overview

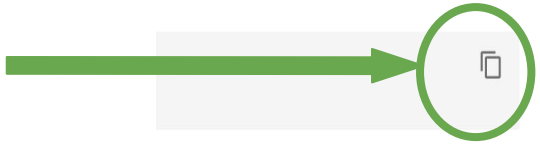






Mini hamburger button to access tutorial section contents

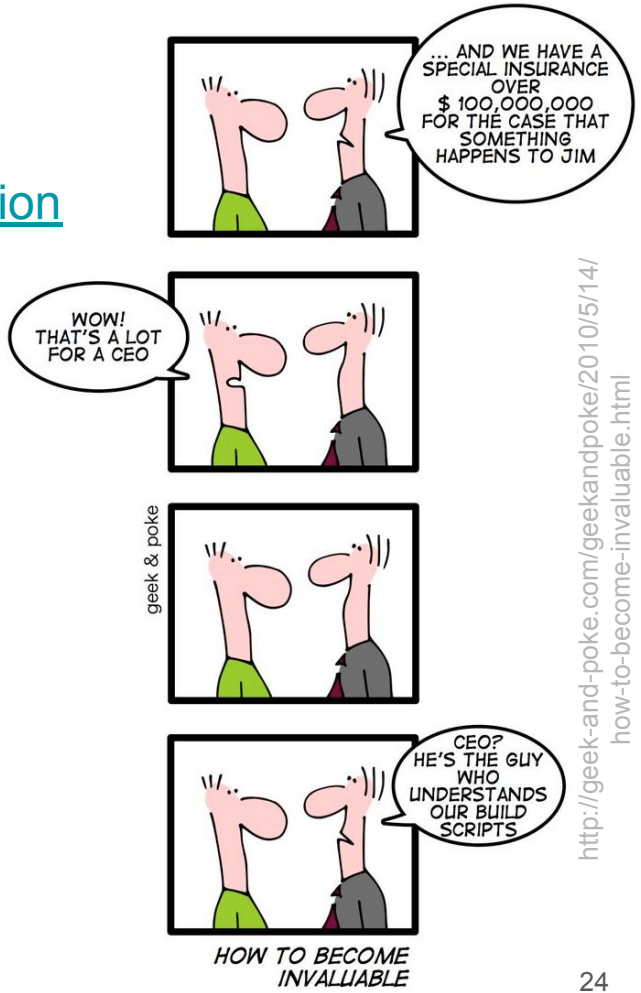
Use copy button In code snippets!



General introduction to EasyBuild

https://easybuilders.github.io/easybuild-tutorial/introduction

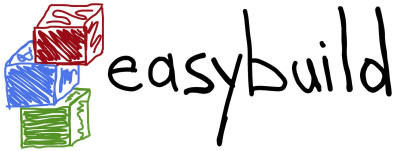
- What is EasyBuild?
- Terminology
- Focus points



Installing EasyBuild

https://easybuilders.github.io/easybuild-tutorial/installation

Installing EasyBuild



https://easybuilders.github.io/easybuild-tutorial/installation

Recommended when using prepared container image:

```
export PATH=$HOME/.local/bin:$PATH
export EB_PYTHON=python3
pip3 install --user easybuild
```

Configuring EasyBuild

https://easybuilders.github.io/easybuild-tutorial/configuration

Configuring EasyBuild

https://easybuilders.github.io/easybuild-tutorial/configuration

Recommended when using prepared container image:

```
export EASYBUILD_PREFIX=$HOME/easybuild
export EASYBUILD_BUILDPATH=/tmp/$USER
```

Using pre-installed software stack

In prepared container image:

```
module use /easybuild/modules/all
```

Basic usage

https://easybuilders.github.io/easybuild-tutorial/basic_usage

- Workflow
- Usage of **`eb`** command
- Installing software
- Exercises

```
$ eb SAMtools-1.10-GCC-9.3.0.eb
== temporary log file in case of crash /tmp/eb-zh7_fyre/easybu
== found valid index for /home/example/.local/easybuild/easyco
== processing EasyBuild easyconfig /home/example/.local/easybu
== building and installing SAMtools/1.10-GCC-9.3.0...
== fetching files...
== creating build dir, resetting environment...
== unpacking...
== patching...
== preparing...
== configuring...
== building...
== testing...
== installing...
== taking care of extensions...
== restore after iterating...
== postprocessing...
== sanity checking...
== cleaning up...
== creating module...
== permissions...
== packaging...
== COMPLETED: Installation ended successfully (took 11 sec)
```

Short break

Next up:

[12:40 - 13:00] Troubleshooting (hands-on)

[13:00 - 13:20] Hierarchical module naming schemes

[13:20 - 14:00] Adding support for additional software (hands-on)

[14:00 - 14:10] *(short break)*

(times are in UTC)

Troubleshooting

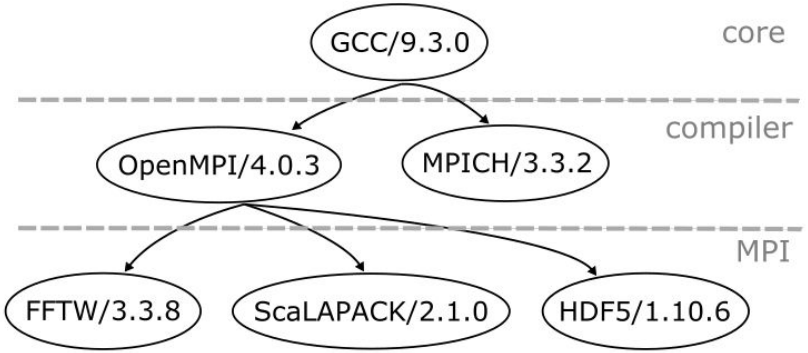
https://easybuilders.github.io/easybuild-tutorial/troubleshooting

- EasyBuild error messages
- EasyBuild log files
- Build directory
- Exercises

Hierarchical module naming schemes

https://easybuilders.github.io/easybuild-tutorial/hmns

- Flat vs hierarchical module naming schemes
- Pros & cons
- Example (demo)
- Exercise



Note: you will run into problems if you are using Singularity because `/easybuild` is read-only!

Adding support for additional software

https://easybuilders.github.io/easybuild-tutorial/adding_support_software

- Easyconfigs vs easyblocks
- Writing easyconfigs
- Generating & copying easyconfigs
- Example (demo)
- Exercise

```
easyblock = 'CMakeMake'

name = 'eb-tutorial'
version = '1.0.0'

homepage = 'https://easybuilders.github.io/easybuild-t
description = "EasyBuild tutorial example"

source_urls = ['https://github.com/easybuilders/easybu
sources = [SOURCE_TAR_GZ]
checksums = ['87643c9a950d02471fc283b31e8a088da7d5d49b

toolchain = {'name': 'GCC', 'version': '9.3.0'}

builddependencies = [('CMake', '3.16.4')]

configopts = "-DEBTUTORIAL_MSG='Hello from the EasyBui

sanity_check_paths = {
    'files': ['bin/eb-tutorial'],
    'dirs': [],
}

sanity_check_commands = ['eb-tutorial']
```

Short break

Next up:

[14:10 - 15:25] EasyBuild at the Jülich Supercomputing Centre

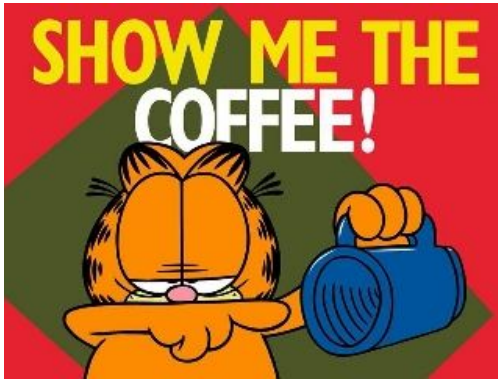
[14:25 - 15:40] EasyBuild at Compute Canada

[14:40 - 15:55] Contributing back to EasyBuild

[14:55 - 16:10] Comparison with other tools

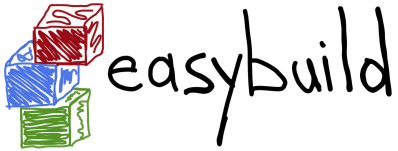
[15:10 - 16:15] Getting help

[15:15 - 16:00] Q&A



(times are in UTC)

EasyBuild at JSC



https://easybuilders.github.io/easybuild-tutorial/jsc

- by Alan O’Cais

Jülich Supercomputing Centre

- JSC is a German supercomputing centre since 1987
  - About 200 experts for all aspects of supercomputing and simulation sciences

JÜLICH Forschungszentrum

Jülich Supercomputing Centre

- JSC is a German supercomputing centre since 1987
  - About 200 experts for all aspects of supercomputing and simulation sciences
- We have 3 primary systems at the moment
  - JUWELS - modular supercomputing, 70 petaflops in 2020
  - JURECA - CPU, GPU and KNL. To be replaced by in 2020
  - JUSUF - AMD, V100 GPU. Interactive workflows and community services

JÜLICH Forschungszentrum

EasyBuild at JSC

- Geared towards *average* user experience
  - Hide lots of indirect software
  - Lots of toolchains => Module hierarchy
  - Renaming some modules, Lmod tweaks

JÜLICH Forschungszentrum

EasyBuild at JSC

- Geared toward *average* user experience
  - Hide lots of indirect software
  - Lots of toolchains => Module hierarchy
  - Renaming some modules, Lmod tweaks
- Custom MNS, toolchains, easyconfigs, easyblocks
  - Maintenance and contribution issue
  - Working hard to remove this where possible

JÜLICH Forschungszentrum

Upgrading and retiring software

- Provide latest software to new projects by default
  - ***Stages*** concept
  - Updates twice per year
  - Encourages users to adopt latest software & dependencies (performance, bug fixes,...)

Upgrading and retiring software

- Provide latest software to new projects by default
  - ***Stages*** concept
  - Updates twice per year
  - Encourages users to adopt latest software & dependencies (performance, bug fixes,...)
- Give indirect access to "retired" software

JÜLICH Forschungszentrum

Leveraging hooks for users & maintainers

- Very powerful alternative to customisations
  - Much more automated and flexible
  - Easier to maintain (particularly for easyconfigs)

Leveraging hooks for users & maintainers

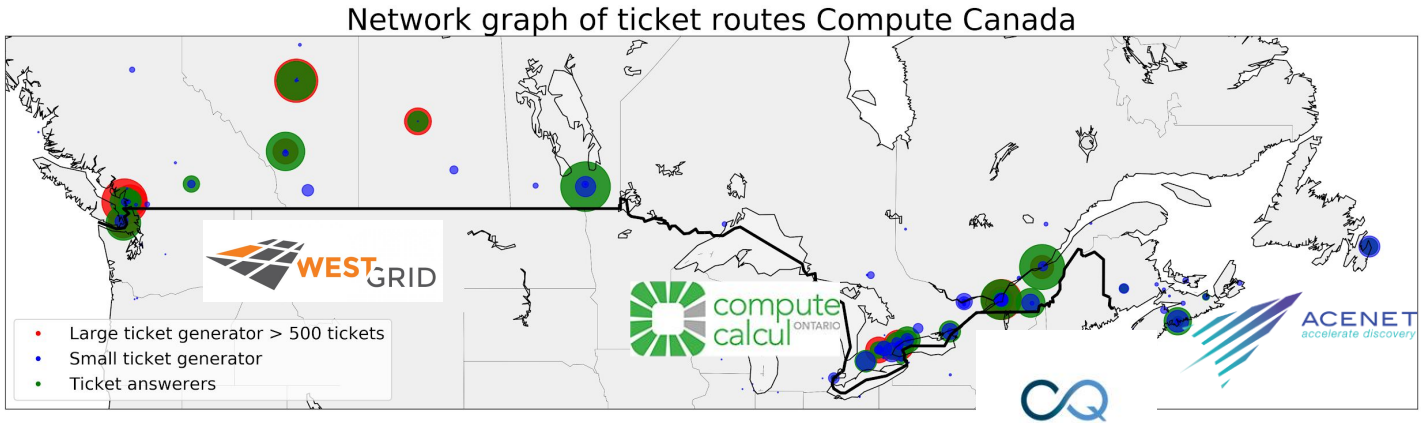
- Very powerful alternative to customisations
  - Much more automated and flexible
  - Easier to maintain (particularly for easyconfigs)
- Enable user space installations
  - Can be leveraged to guide people on how to do this “properly”

EasyBuild at Compute Canada

https://easybuilders.github.io/easybuild-tutorial/computecanada

- by Maxime Boissonneault

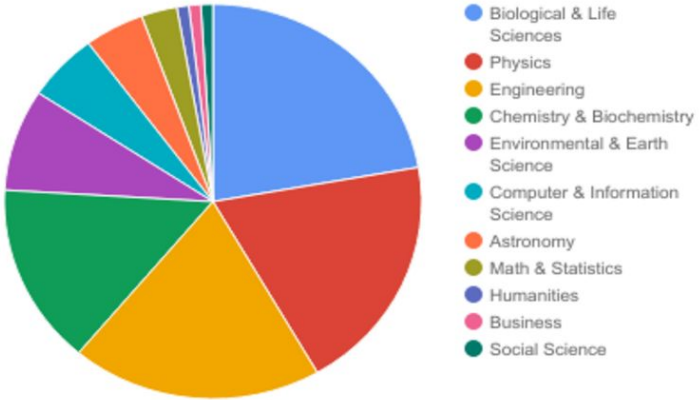
Compute Canada : the people



All research disciplines supported

Free access for any researcher at a Canadian institution

- 4 regional consortia
- 35 member institutions
- ~200 technical staff
- ~15,000 user accounts
  - 20% growth per year



Before 2015

- Around 30 Compute Canada sites hosting hardware
- Over 50 clusters or other hardware resources
- All configured differently

Compute Canada : the hardware

5 major national systems
~15 legacy systems
270K cores, 2500 GPUs,
70 PB disk, 180 PB tape

System	Type	Network	Production
Arbutus	Cloud	10 GbE	2016 H2
Cedar	General	OPA	2017 H1
Graham	General	EDR IB	2017 H1
Niagara	Large MPI	EDR IB	2018 H1
Béluga	General	EDR IB	2019 H1

Goal

Users should be presented with an interface that is as **consistent** and **easy to use** as possible across **all sites**. It should also offer **optimal performance**.

1. All software should be accessible on every site, reliably and performantly.
2. Software should be independent from the underlying OS stack.
3. Software installation should be tracked and reproducible via automation.
4. The user interface should make it easy to use a large and evolving software stack.

What this means

All new Compute Canada sites

1. Need a distribution mechanism
   a. CVMFS : CERN Virtual Machine File System

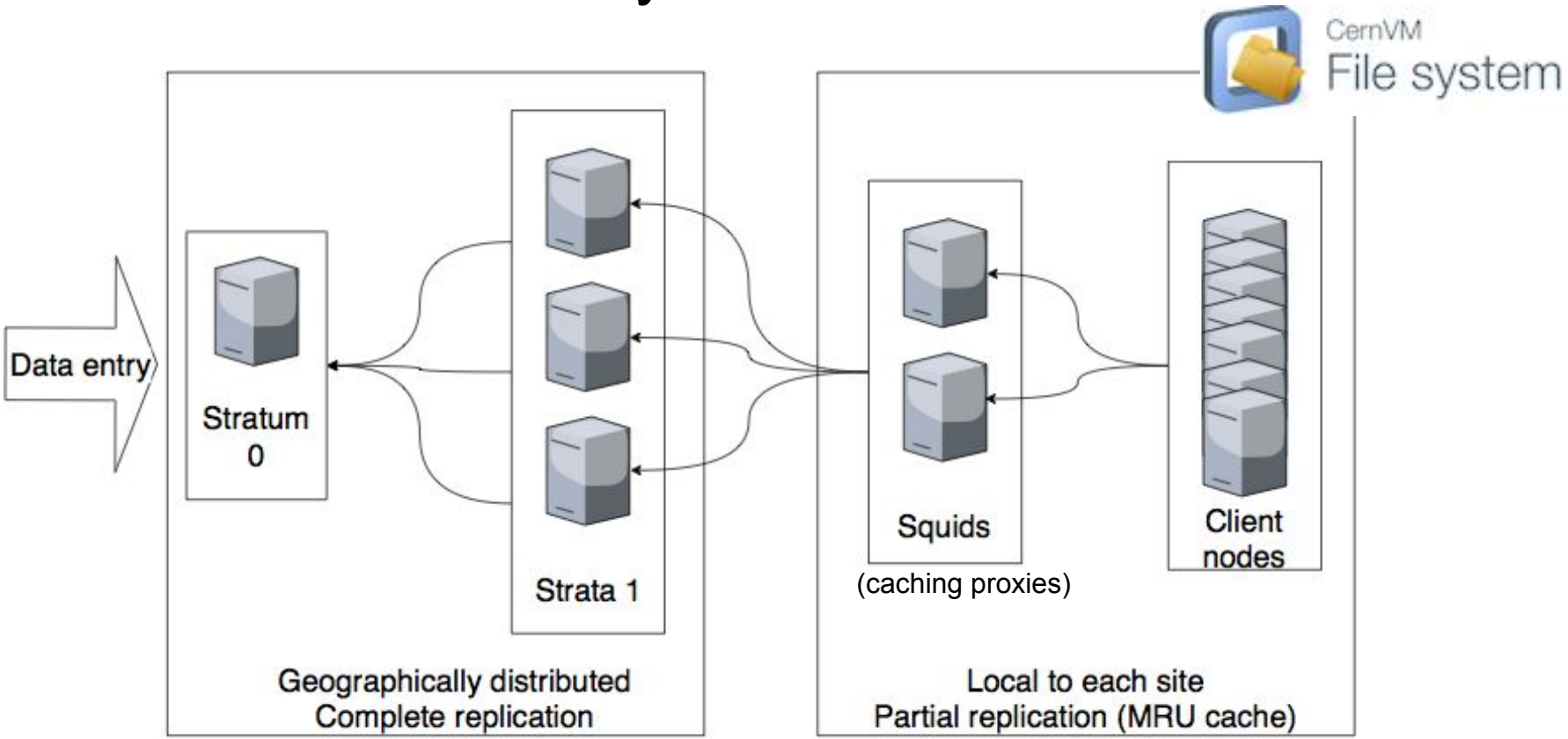
Consistency

2. Independent of the OS (Ubuntu, CentOS, Fedora, etc.)
   a. Nix ===> Gentoo Prefix
3. Automated installation (humans are not so consistent)
   a. EasyBuild

Easy to use

4. Needs a module interface that scale well
   a. Lmod with a hierarchical structure

CVMFS content delivery



Software: design overview

Easybuild layer: modules for Intel, PGI, OpenMPI, CUDA, MKL, high-level applications. Multiple architectures (sse3, avx, avx2, avx512)
`/cvmfs/soft.computecanada.ca/easybuild/{modules,software}/2017`

Easybuild-generated modules around Nix profiles (mostly deprecated): GCC, Eclipse, Qt+Perl+Python no longer
`/cvmfs/soft.computecanada.ca/nix/var/nix/profiles/[a-z]*`

Nix/Gentoo layer: GNU libc, autotools, make, bash, cat, ls, awk, grep, etc.
`module nixpkgs/16.09 => $EBROOTNIXPKGS=`
`/cvmfs/soft.computecanada.ca/nix/var/nix/profiles/16.09`

Gray area: Slurm, Lustre client libraries, IB/OmniPath/InfiniPath client libraries (all dependencies of OpenMPI). In Nix layer, but can be overridden using PATH & LD_LIBRARY_PATH.

OS kernel, daemons, drivers, libcuda, anything privileged (e.g. the sudo command): always local. Some legally restricted software too (VASP)

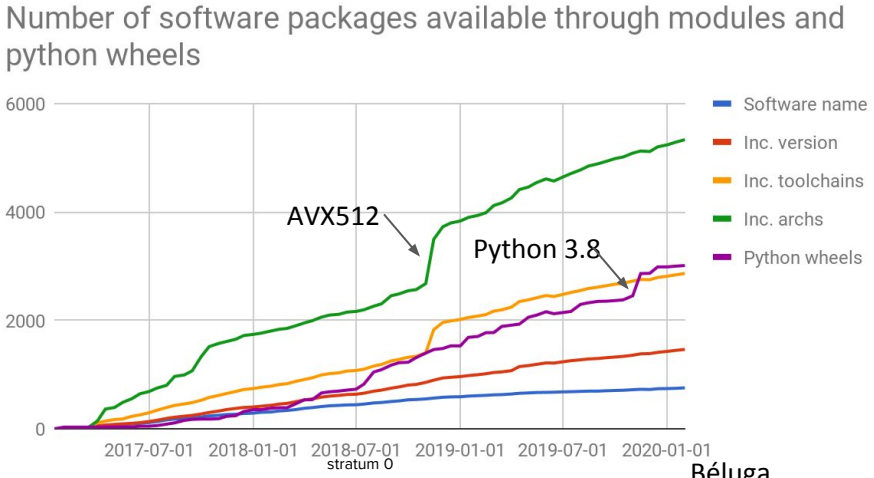
Compute Canada Software Stack

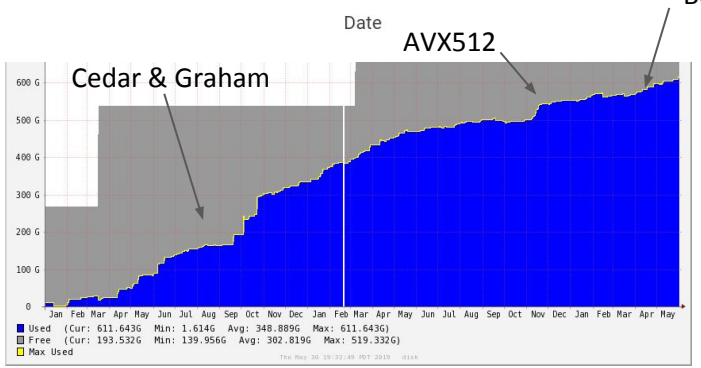
Available software

800+ scientific applications

6,000+ permutations of version/arch/toolchain

Type	Modules
AI	5
Bioinformatics	239
Chemistry	63
Data	19
Geo/Earth	23
Mathematics	82
MPI libraries	7
Physics	48
Various tools	176
Visualisation	28
Misc	38





53

- Two major new clusters with Skylake CPUs
- Built new modules with AVX512 for most packages
- High deduplication
- Further details

Design choices / EB features

- Compatibility layer => filtering of a lot of dependencies (M4, cmake, etc.)
- Toolchains based combinations of
  - Intel/GCC, OpenMPI, MKL, Cuda
  - Not “foss” nor “intel”
  - => We are (ab)using the `--try-toolchain`, `--try-software-version`, `--try-update-deps` options
- Custom MNS :
  - Hierarchical, lower case
  - No version suffix at all
  - Toolchains are hidden
- No LD_LIBRARY_PATH

Hooks

- Injecting custom configuration options for OpenMPI
- Injecting footer code in compiler and MPI modules to support installation in user's home directories
- Splitting the installation of Intel into redistributable and non-redistributable parts
- Stripping down Python modules (dropping extensions)

Handling Python

- Installing Python wrappers and side packages (PyQt5 with Qt5, OpenCV-python with OpenCV, etc.) whenever possible
- Using `multi_deps` so that modules are compatible with all versions of python
- Not installing most python packages as modules (see next slide)
- Not supporting Anaconda

Python wheels

What are wheels?

Wheels are the new standard of Python distribution and are intended to replace eggs. Support is offered in pip >= 1.4 and setuptools >= 0.8.

Advantages of wheels

1. Faster installation for pure Python and native C extension packages.
2. Avoids arbitrary code execution for installation. (Avoids setup.py)
3. Installation of a C extension does not require a compiler on Linux, Windows or macOS.
4. Allows better caching for testing and continuous integration.
5. Creates .pyc files as part of installation to ensure they match the Python interpreter used.
6. More consistent installs across platforms and machines.
7. **You can compile your own wheels, linking against your compiled libraries**

You can use this too!

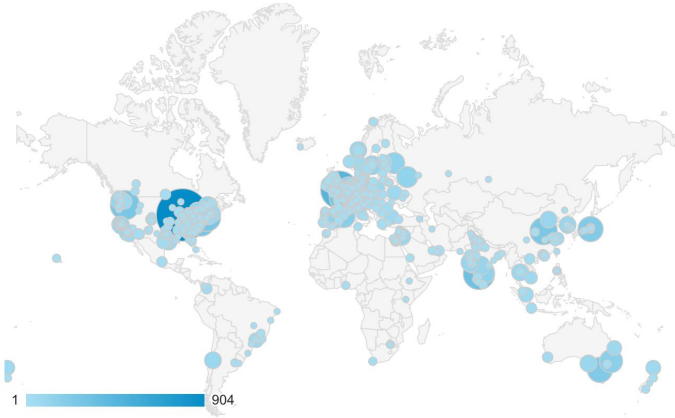
Mounting our software stack:

https://docs.computecanada.ca/wiki/Accessing_CVMFS

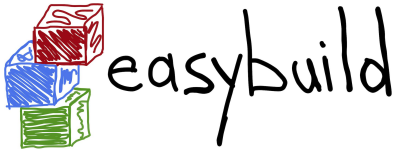
The EasyBuild community

https://easybuilders.github.io/easybuild-tutorial/community

- Brief history of the project
- Who is using EasyBuild?
- Who is maintaining EasyBuild?
- EasyBuild events



Contributing to EasyBuild



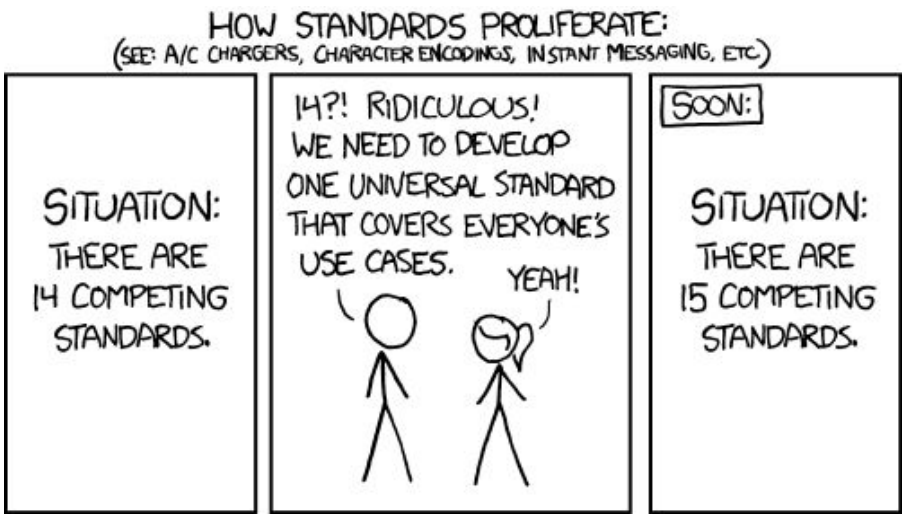
https://easybuilders.github.io/easybuild-tutorial/contributing

- Contribution workflow
- GitHub integration features in EasyBuild
- Contribution stats

Comparison with other tools

https://easybuilders.github.io/easybuild-tutorial/comparison_other_tools

- Spack
- Nix
- Guix
- CONDA

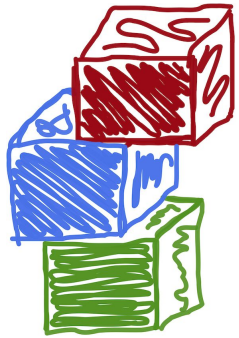


https://xkcd.com/927

Getting help

https://easybuilders.github.io/easybuild-tutorial/getting_help

- Documentation
- GitHub
- Mailing list
- Slack
- Bi-weekly conf call



easybuild

Any more questions?

Please take a minute to fill out the evaluation survey !

https://nl.surveymonkey.com/r/eb-tutorial