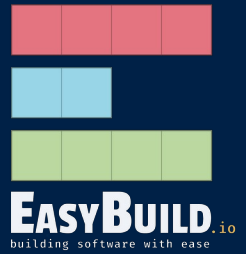
Imperial College London



The Challenges of Installing Software on HPC Systems

Santiago Lacalle Puig

slacalle@imperial.ac.uk
Github: @slacalle

EUM23 - 8th EasyBuild User Meeting

Imperial College
London

Overview

- Challenges of installing scientific software.
- Compiling software.
- Other package manager: conda
- System package managers (yum, apt).
- Containers: Singularity.
- Challenges not related to software itself.
- High performance software.
- EasyBuild.

Imperial College London

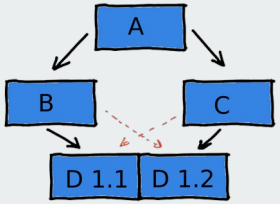
Challenges of installing scientific software

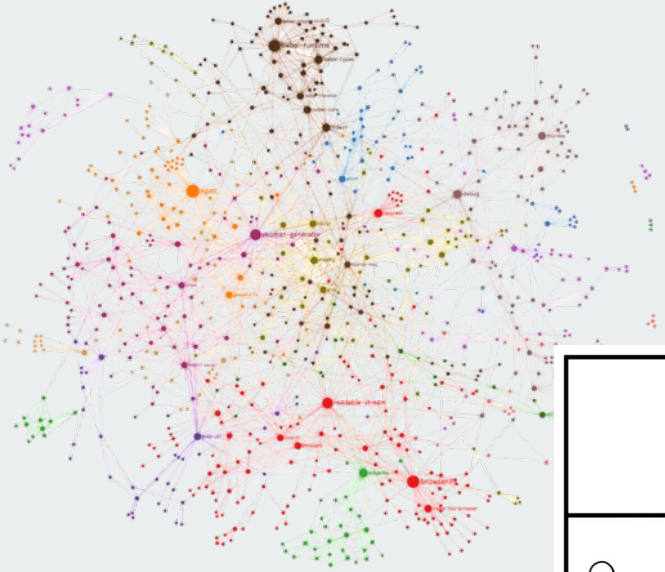
- Non-standard installations. Very time consuming.
- Support Software that is no longer actively developed.
- Lack of documentation and poor software engineering practices from software developers. e.g.
  - Enhancing code readability
  - Keeping code efficient
  - Version control
  - Being descriptive
  - Applying KISS - Keep it Simple, Stupid.
- Dependencies

Imperial College London

Dependencies Hell...

Long chains of dependencies
- Conflicting dependencies
- Circular dependencies
- Package manager dependencies
- Diamond dependency

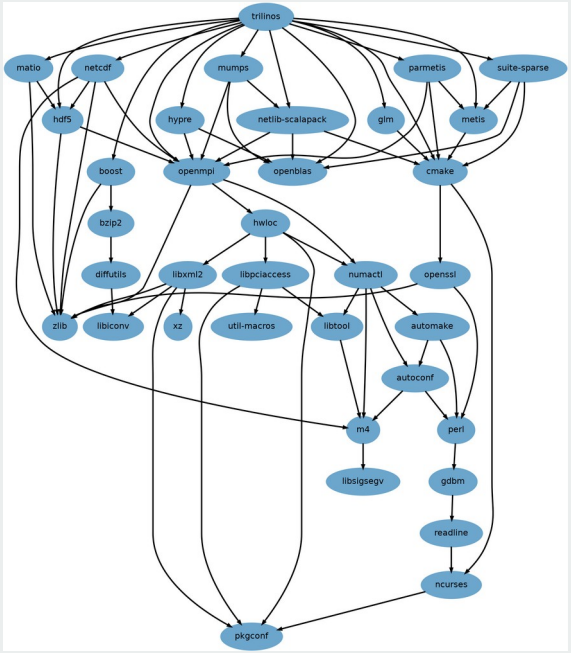


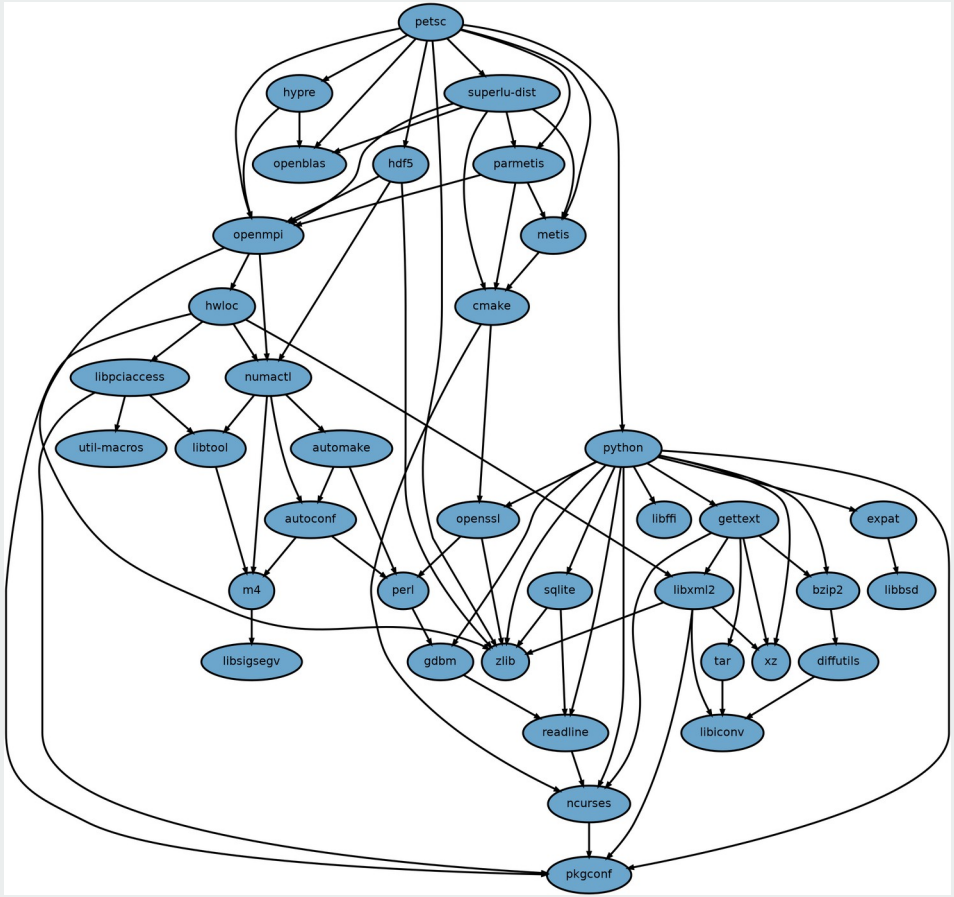


EasyBuild

Imperial College London

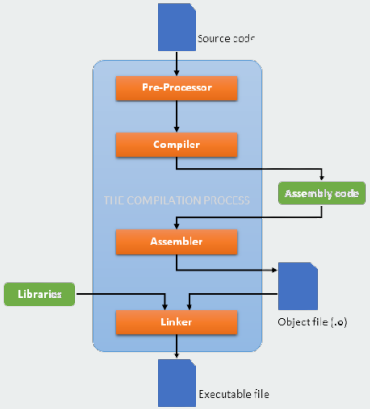
Dependency Hell





Imperial College London

Compiling Software



- Environment modules.
- Environment variables: PATH, LDFLAGS, CPATH, CPPFLAGS, LD_LIBRARY_PATH, RPATH.
- MPI, MKL, BLAS, LAPACK
- Autotools, Make, Cmake

Example: quantum espresso

```
$ ./install/configure –prefix=/apps/espresso/6.3-new BLAS_LIBS=-L/apps/intel/2017.6/compilers_and_libraries_2017.6.256/linux/mkl/lib/intel64/ \
-lmkl_intel_lp64 -lmkl_sequential -lmkl_core LAPACK_LIBS=-L/apps/intel/2017.6/compilers_and_libraries_2017.6.256/linux/mkl/lib/intel64/ \
-lmkl_intel_lp64 -lmkl_sequential -lmkl_core CPPFLAGS=-I/apps/mpi/intel/2018.1.163/include \
-I/apps/intel/2017.6/compilers_and_libraries_2017.6.256/linux/mkl/include/fftw \
-I/apps/hdf5/1.8.15/parallel/include LDFLAGS=-L/apps/mpi/intel/2018.1.163/lib \
-L/apps/hdf5/1.8.15/parallel/lib -L/apps/intel/2017.6/compilers_and_libraries_2017.6.256/linux/mkl/lib/intel64 \
-L/apps/intel/2017.6/clck/2017.2.019/lib/intel64 --with-hdf5=/apps/hdf5/1.8.15/parallel
```

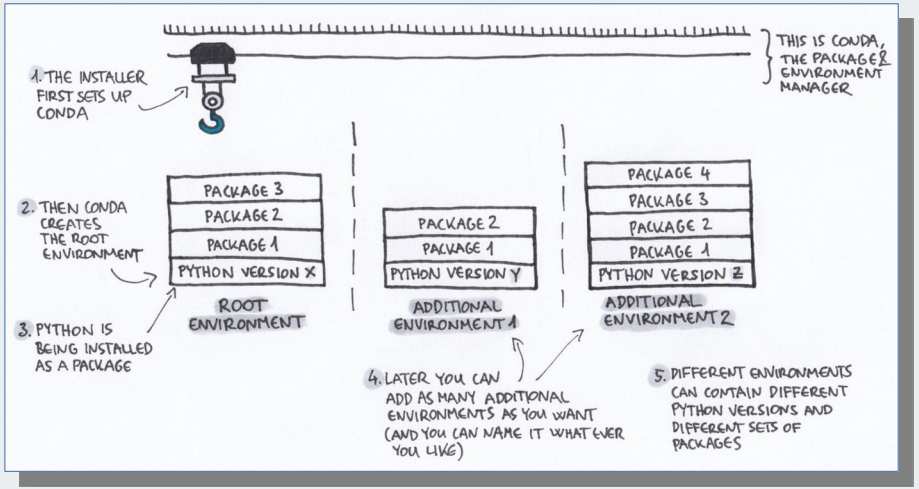
Imperial College London

CONDA®

Other package managers - Conda

Conda

- Over 7500 packages.
- Aimed at end users/researchers.
- Empower users!
- Reduces support requests.
- Pre-built binaries.
- Relatively quick.
- Plays well with other services (e.g. Jupyter)
- Wide adoption.



EasyBuild

Imperial College London

Caveats and considerations of using Conda

- Cannot be used for everything. Mainly python and R.
- Environments can become delicate.
- Increased user support requests to fix environments.
- Requires some playing around to learn a number of quirks.
- Installations for non-conda packages can become tedious and more complicated than using alternatives.
- Doesn't play nice at times with pip or install.packages()

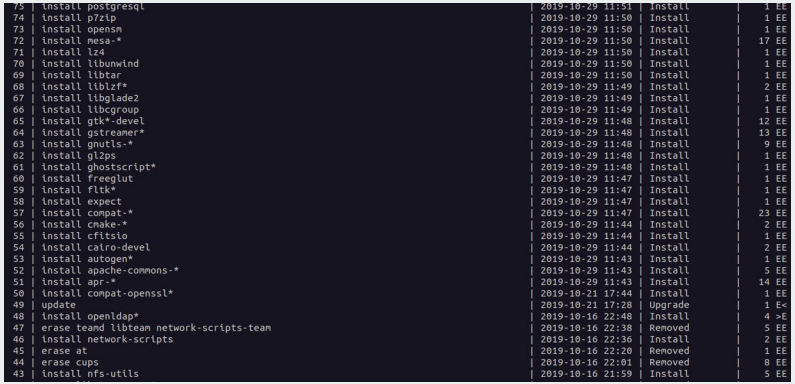
```
$ conda create -n test_renv r-base \
r-data.table r-plyr r-ggplot2 r-seurat r-biocmanager \
r-doparallel -c conda-forge -c bioconda
```

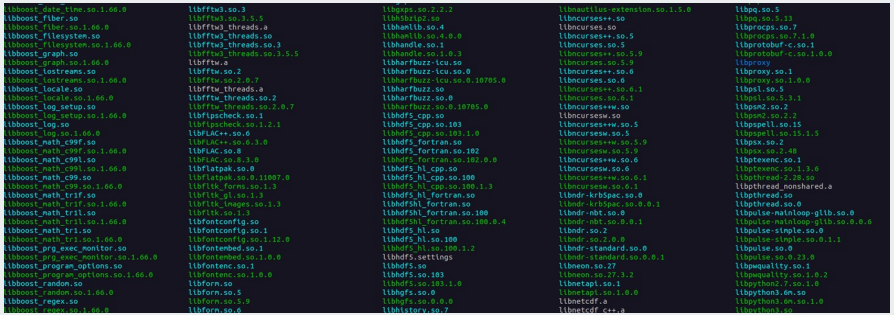
r-base + 6 packages > 247 Packages!

Imperial College
London

Other Package managers – yum, apt

- Quick solution
- Can cause problems in the future
- Grows the system image
- And most importantly…. Not optimized!





Imperial College London

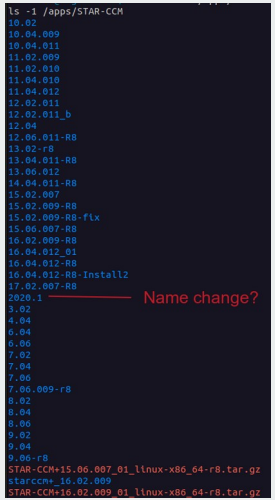
Containers - Singularity

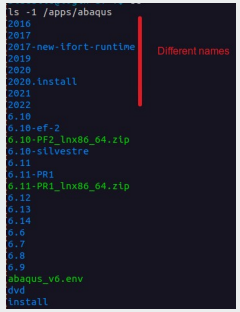
- Containers provide a practical solution for replicating and circulating a pre-existing software stack.
- Some containers are not as easy to create. They can have have numerous dependencies.
- Leveraging the os package manager within a container is not enough.
  - Generic binaries not tailored to specific architectures. Not optimized!
- Containers may require ongoing support or rebuilding to account for updates/changes or accommodate multiple architectures.

Imperial College London

Challenges not related to software itself:

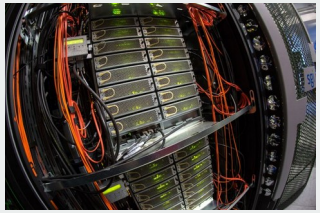
- Manage incoming requests and deliver in a timely manner.
- Provide a good level of documentation for service users.
- Efficiently maintain software stacks.
- Ensure organization and standardization across the stack (directory naming, modules).
- Ensure software is optimized for each architecture.





Imperial College London

HPC



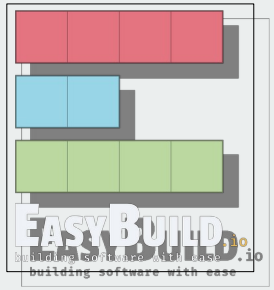
Are we delivering High Performance Software?

- Are we delivering the most efficient software?
- Who should care for optimized software?
  Package managers/sys admins or end users?

- Considerable performance gains in certain cases.
- Running more efficient code can mean:
  - Saving users time.
  - Reducing compute time.
  - Decreasing carbon footprint.
  - Money saved.

Imperial College London

EasyBuild



Misconceptions:

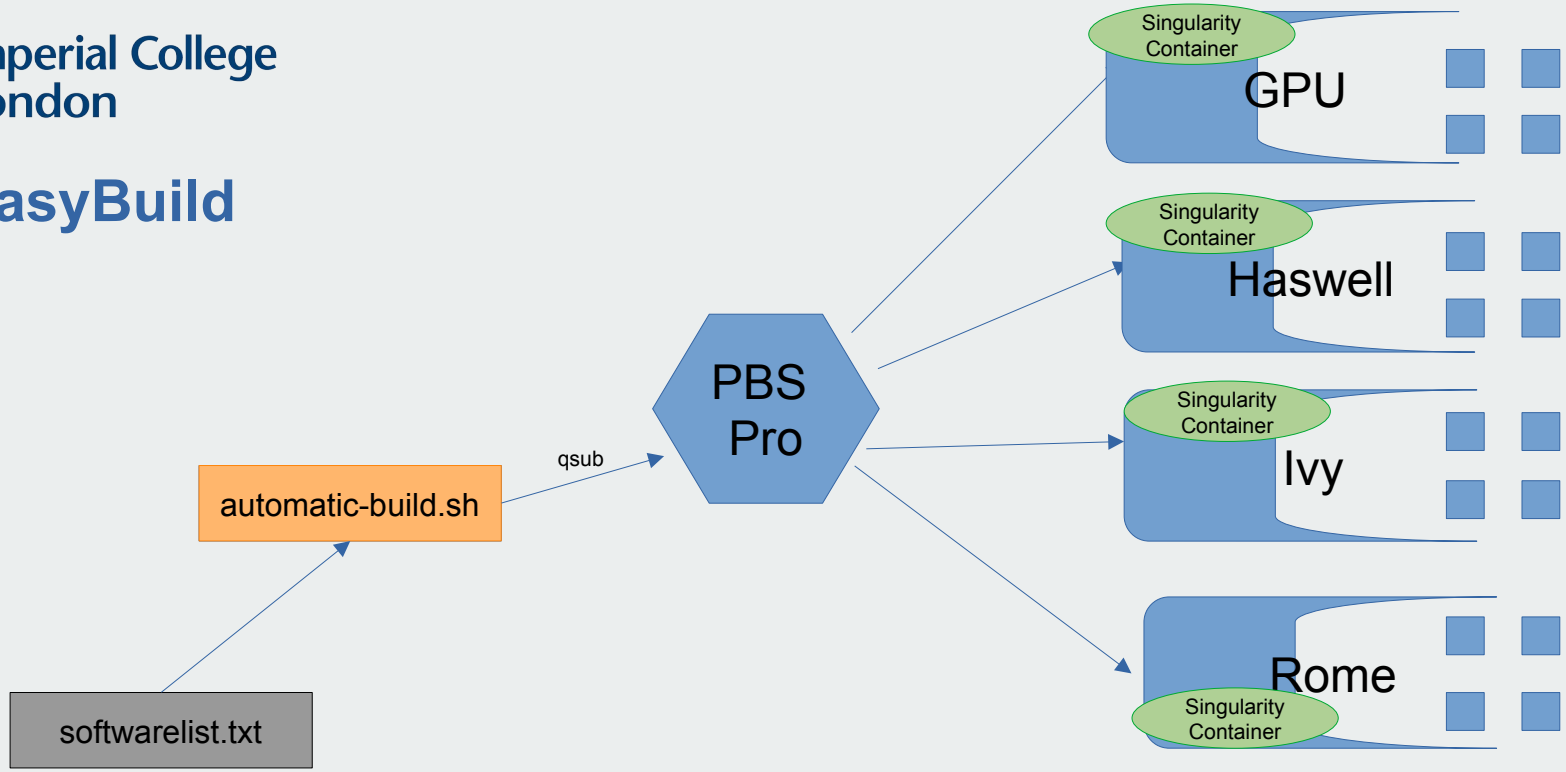
- EasyBuild will make it harder to install software.
  - Adds another layer of complexity on top of the software.
  - Will slow down complex installs.

EasyBuild solves:

- Used to install multiple versions and manage dependencies
- Automatically installs (less pain for package managers)
- Reduces overall install time
- Ensures standardization in naming and organization in modulefiles
- Optimized software for our hardware. PERFORMANCE!

Imperial College London

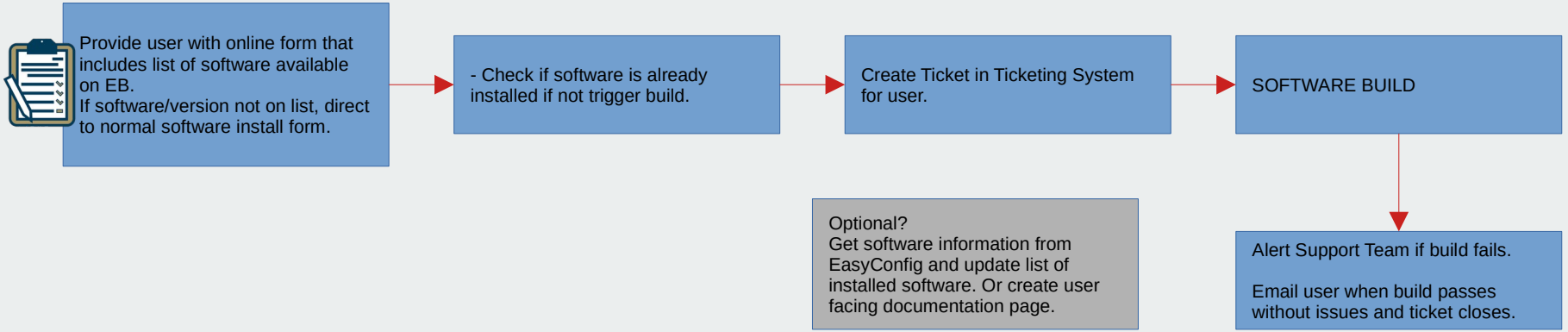
EasyBuild



Imperial College London

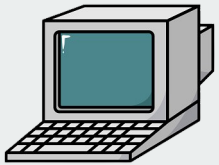
Future plans:

- Automate user software installation requests to trigger EasyBuild



Imperial College London

EasyBuild



Old software stack (manual installs)

Pre 2013

Total modules: 2776

New Software stack (Easybuild)

Since March 2022

1585 modules in development stack
1152 modules in production stack
Total modules: 2737

Imperial College London

Thank you!