

High Performance Containers (HPC)

with



easybuild or

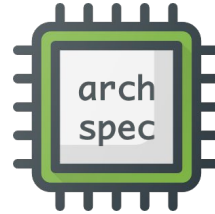


Spack

Kenneth Hoste (HPC-UGent)

kenneth.hoste@ugent.be

and



Todd Gamblin (LLNL)

tgamblin@llnl.gov

6th HPC Container Workshop (ISC'20)

<https://hpcw.github.io>

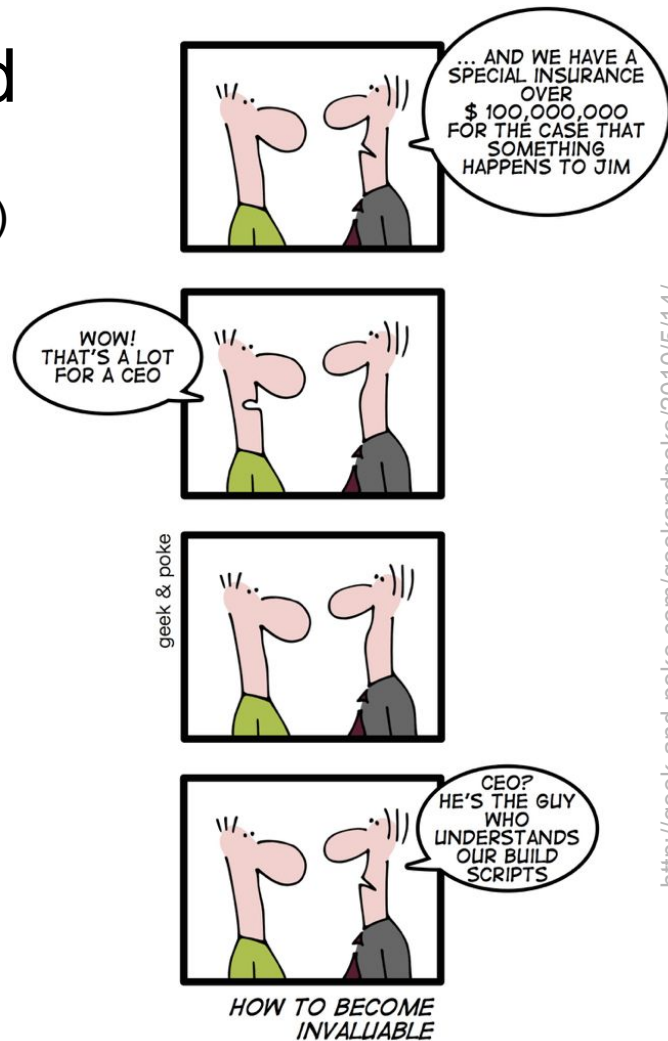
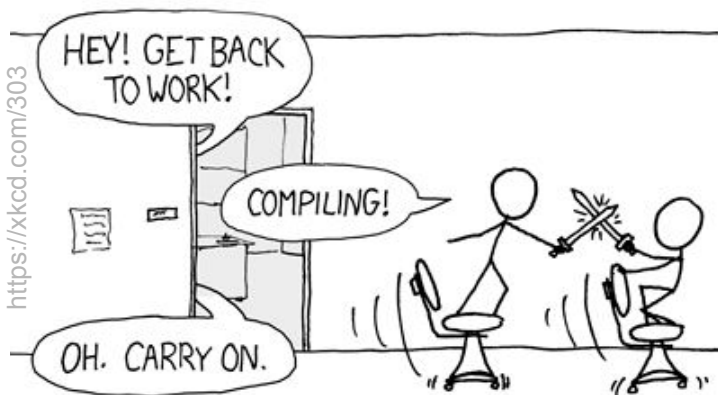


Getting scientific software installed

- compilation from source (because of the 'P' in HPC)
- things get messy fast, and it's getting worse...

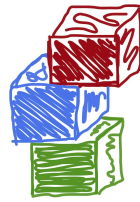
THE #1 PROGRAMMER EXCUSE
FOR LEGITIMATELY SLACKING OFF:

"MY CODE'S COMPILING."



github.com/easybuilders

easybuild.readthedocs.io



easybuild

easybuild.slack.com

(join via easybuild-slack.herokuapp.com)

youtube.com/c/easybuilders

- framework for **installing scientific software on HPC systems**
- originally created by HPC team at Ghent University (Belgium) in 2009
- **stable since 2012**, frequent releases (every 6-8 weeks), extensively tested
- installations are optimized for processor architecture of host by default
- supports **almost 2000 different software packages** (excl. extensions), including NWChem, OpenFOAM, PETSc, PyTorch, R, TensorFlow, Trilinos, ... (see https://easybuild.rtd.io/en/latest/version-specific/Supported_software.html)
- frequent hackathons & meetups, incl. 5 **EasyBuild User Meetings** (2016-2020)

Stacking software installations is trivial

- Availability of modules is sufficient to leverage existing installations (for example in Cray PE).
- Centrally installed software can be used as dependencies by end users who prefer maintaining their own software stack.
- Custom installations can shadow existing ones.



GitHub integration

- create PRs from EasyBuild CLI
`eb --new-pr example.eb`
- use easyconfigs from PR, upload test report
`eb --from-pr 12345 --upload-test-report`
- PR preview: compare with existing easyconfigs
`eb --preview-pr example.eb`



Easy customization through hooks & plugins

- Support for additional software & toolchains can be provided via `--include-*` options.
- Hooks are provided to customize the software installation procedures where required, without having to maintain patches.

Submitting software installations as jobs

```
$ eb example.eb -r --job
```

```
...
```

```
3 jobs submitted.
```

```
$ squeue
```

123	dep-one-1.0	RUNNING	node001
124	dep-two-2.0	RUNNING	node002
125	example-1.2.3	PENDING	(Dependency)





Spack

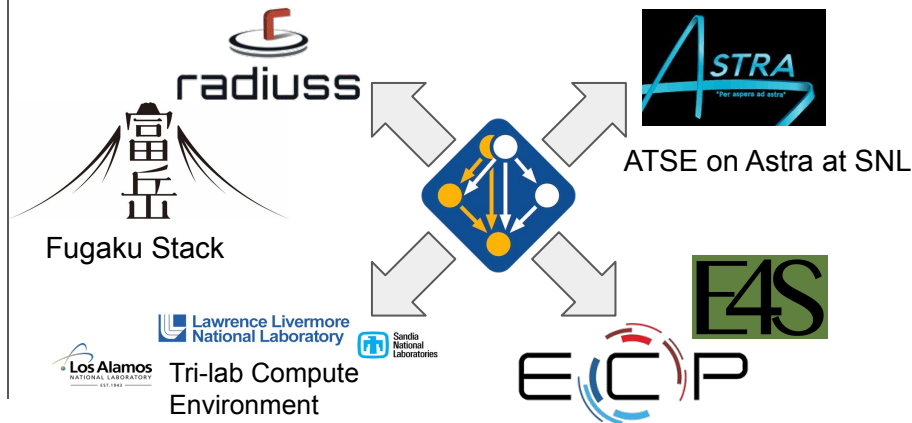
- Single slide intro goes here

TODD

Flexible command line interface

```
$ spack install mpileaks
$ spack install mpileaks@3.3
$ spack install mpileaks@3.3 %gcc@4.7.3
$ spack install mpileaks@3.3 %gcc@4.7.3 +threads
$ spack install mpileaks@3.3 cppflags="-O3 -g3"
$ spack install mpileaks@3.3 target=skylake
$ spack install mpileaks@3.3 ^mpich@3.2 %gcc@4.9.3
```

Many offshoot distributions



Spack

Packages are templates

TODD

Binary packaging

TODD



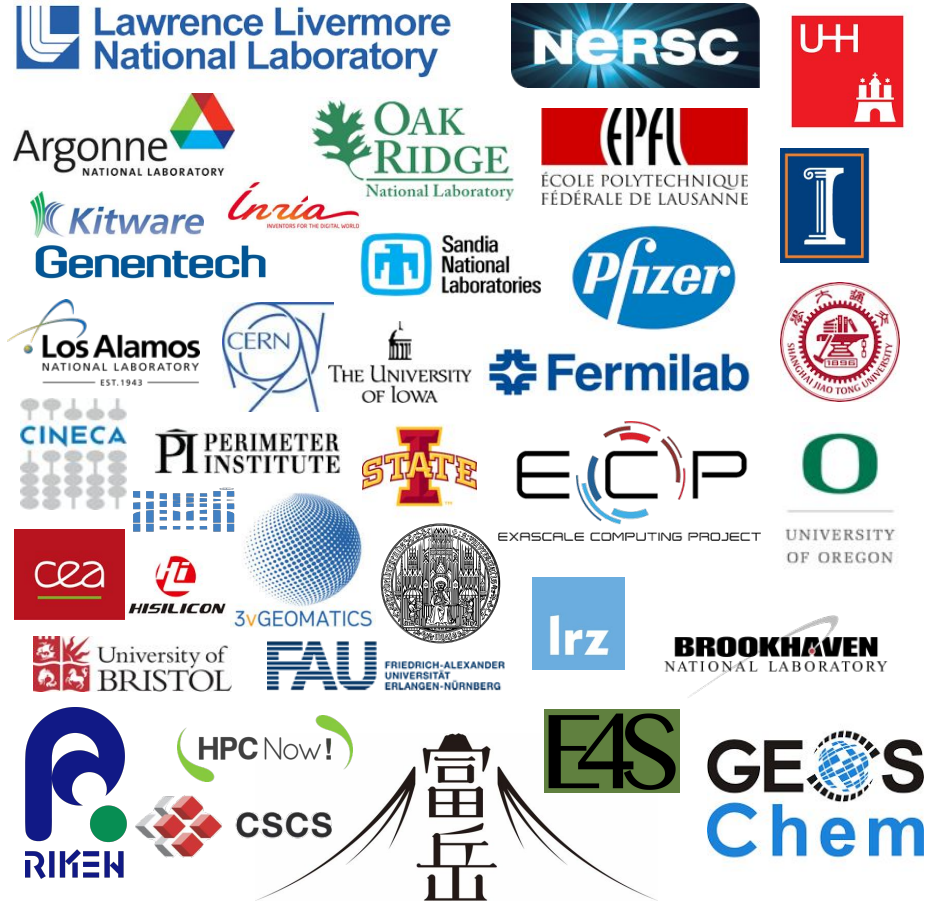
easybuild vs Spack : similarities

- implemented in Python (compatible with both Python 2 & 3)
- open source, development in  **GitHub**
- installing (scientific) software without admin privileges
- focus on HPC: compiling software from source, **performance** is key
- similar high-level structure (framework + easyblocks, core + packages)
- highly configurable
- well documented (easybuild.rtdf.io, spack.rtdf.io)
- **worldwide communities**
- broad spectrum of supported software (1000s)
- **support for building containers!**





Worldwide communities



easybuild vs Spack : key differences

license	GPLv2	MIT or Apache 2.0 (dual license)
OS support	Linux, Cray PE	Linux, macOS, Cray PE
command line interface	eb example-1.2.3.eb -r	spack install example@1.2.3
packages	easyblocks + easyconfigs	templated packages
configuration (TODD: OK?)	INI cfg files, env. vars, CLI opt.	YAML files
main focus	HPC support teams	HPC support teams, software developers, and end users
dependency management	fixed versions	flexibility + lock files
access to installed software	environment modules	modules, environments, spack load
unique aspects & features	distributed installation, GitHub integration, hooks & plugins, central easyconfigs repository, ...	concretization of abstract spec, binary packages, environment, CI integration (GitLab), uninstall

Building containers with Spack

1. Make an environment

```
spack:
  specs:
  - gromacs+mpi
  - mpich
```

2. Run a command

spack containerize

3. Build generated container recipe

- Can generate a container image from any Spack environment
- Generated builds are multi-stage
 - Final image is stripped
 - Contains only what's needed to run
 - Compilers, build deps, etc. are removed
- Optionally customize:
 - base image
 - Labels
 - format (singularity/Dockerfile)
 - extra commands to run as part of the build

```
container:
  # Select the format of the recipe e.g. docker,
  # singularity or anything else that is currently supported
  format: docker

  # Select from a valid list of images
  base:
    image: "centos:7"
    spack: develop
```

```
# Build stage with Spack pre-installed and ready to be used
FROM spack/centos7:latest as builder

# What we want to install and how we want to install it
# Is specified in a manifest file (spack.yaml)
RUN mkdir /opt/spack-environment \
&& (echo "spack:" \
&& echo "  specs:" \
&& echo "    - gromacs+mpi" \
&& echo "    - mpich" \
&& echo "  concretization: together" \
&& echo "  config:" \
&& echo "    install_tree: /opt/software" \
&& echo "    view: /opt/view") > /opt/spack-environment/spack.yaml

# Install the software, remove unnecessary deps
RUN cd /opt/spack-environment && spack install && spack gc -y

# Strip all the binaries
RUN find -L /opt/view/* -type f -exec readlink -f '{}' \; | \
xargs file -l | \
grep 'charset=binary' | \
grep 'x-executable|x-archive|x-sharedlib' | \
awk -F: '{print $1}' | xargs strip -s

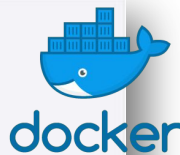
# Modifications to the environment that are necessary to run
RUN cd /opt/spack-environment && \
spack env activate --sh -d . >> /etc/profile.d/z10_spack_environment.sh

# Bare OS image to run the installed executables
FROM centos:7

COPY --from=builder /opt/spack-environment /opt/spack-environment
COPY --from=builder /opt/software /opt/software
COPY --from=builder /opt/view /opt/view
COPY --from=builder /etc/profile.d/z10_spack_environment.sh /etc/profile.d/z10_spack_environment.sh

RUN yum update -y && yum install -y epel-release && yum update -y
&& yum install -y libgomp \
&& rm -rf /var/cache/yum && yum clean all

RUN echo 'export PS1="\[\$(tput bold)\]\[\$(tput setaf 1)\][gromacs]\[\$(tput setaf 2)\]\u\[\$(tput
```



Building containers with easybuild



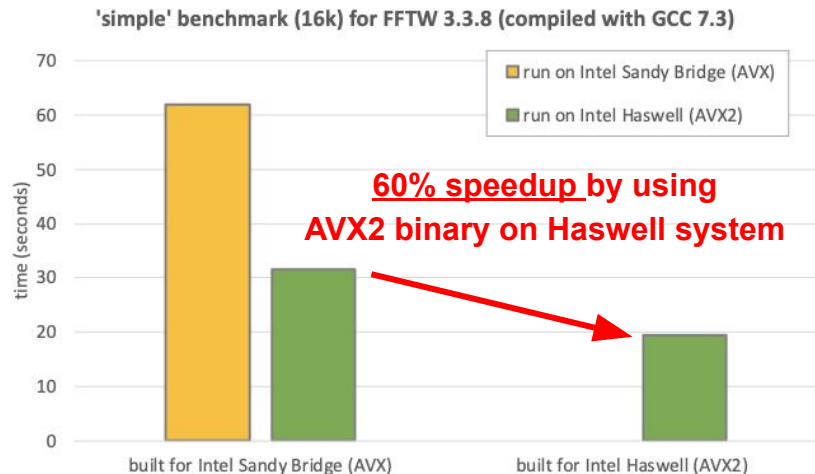
easybuild.readthedocs.io/en/latest/Containers.html

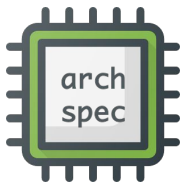
youtube.com/watch?v=FPqRDWbtv1M

- **experimental** support for both Docker & Singularity
- generate container recipe using a single command: `eb --containerize`
- recipe will use EasyBuild to install specified software (and all missing dependencies)
- use existing container image as a base, or start from scratch (bare OS)
- currently focused on using RHEL-based OS in container
- ... or you can just use EasyBuild as an easy way to install software in a container


Containers vs HPC

- container images are typically built with *generically optimized* binaries
 - to ensure they can always be run, regardless of processor architecture of the host
 - similar to packages installed by traditional package managers (yum/dnf, apt, ...)
 - also applies to conda (to some extent)
- result: capabilities of the host processor may largely go unused
- impact on performance can be dramatic
- basically, **performance is sacrificed** to provide “mobility of compute”
- this performance aspect is largely ignored in the container community...





A library to reason about system architecture aspects

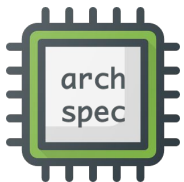
- originally part of Spack, now a standalone library
- currently focused on CPU microarchitectures, intention is to extend the concept to network, GPUs, ...
- JSON file with “database” of CPU microarchitectures (Intel, AMD, ARM, POWER)
- **fine-grained human-readable labels**, a.k.a. codenames (haswell, skylake, ...)
-  Python library available to leverage this data: `pip install archspec`
- **query instruction set support, compatibility between microarchitectures, ...**
- cross-project collaboration between Kenneth Hoste (EasyBuild) and Todd Gamblin + Massimiliano Culpo (Spack)



github.com/archspec

archspec.readthedocs.io

license: Apache 2.0 or MIT



What can archspec be used for?

- collect host info (via `/proc/cpuinfo` on Linux, `sysctl` command on macOS)
- determine codename of host CPU via archspec CLI →
- check **compatibility** of host with given microarchitecture:

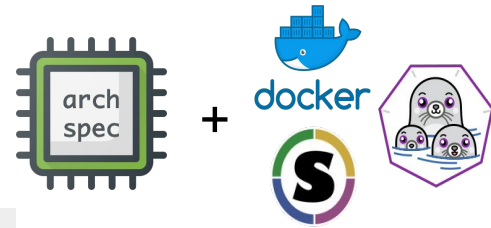
```
$ archspec cpu  
skylake
```

“Can the current host run binaries that were compiled for Intel Haswell?”

- query **capabilities** of microarchitectures: “Does Intel Haswell support AVX-512?”
- query **compiler flags**: *“How do I compile for Intel Broadwell with icc?”*
- **partial ordering** of microarchitectures, picking best match from list of options:

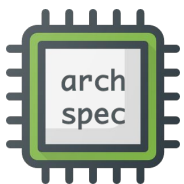
“Given AVX, AVX2 and AVX-512 binaries, what’s the best match on a Broadwell system?”

Enabling HPC (High Performance Containers)



Using `archspec` we can avoid having to choose between performance and mobility of compute!

- build **optimized** container images for a specific CPU microarchitecture
- **tag** container images with the CPU microarchitecture they were built for
- **check compatibility** with host from container and act on it
(informative error message, warn about potential performance loss)
- picking **most suitable container image** for host w.r.t. CPU microarchitecture
- avoid need for using fat binaries (increases container size, requires rebuilds)



We need your help!

- We would like to see the HPC community to **standardize** the labels/codenames
- CPU vendors could contribute to the `archs-spec-json` repo
 - add feature info for new microarchitectures
 - review current data & submit corrections (if needed)
- Companies (Cray, NVIDIA, Red Hat, ...) could adopt archspec & contribute to it
- Additional language bindings can be implemented (Go is already WIP)
- We're open to ideas & contributions to extend the concept to GPUs, network, ...
- **Reach out to us!**

Kenneth: kenneth.hoste@ugent.be



@boegel



@kehoste

Todd: tgamblin@llnl.gov



@tgamblin