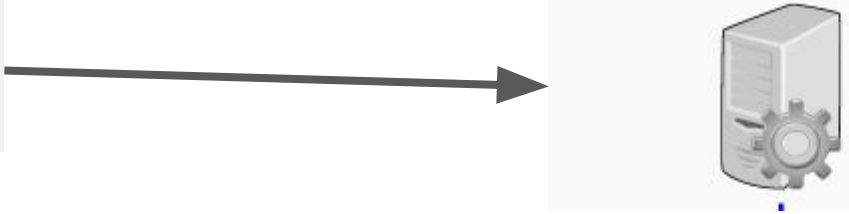
Building HEP Software with Spack: Experiences from Pilot Builds for Key4hep and Outlook for LCG Releases

vCHEP 2021

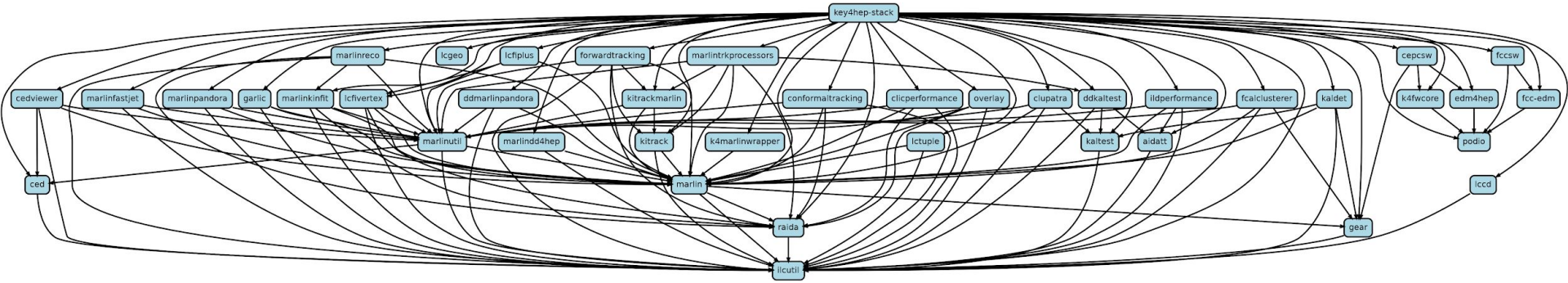
Valentin Volkl (CERN), Thomas Madlener, Tao Lin, Joseph Wang, Dmitri Konstantinov, Ivan Razumov, Andre Sailer, and Gerardo Ganis

Intro: Goals for a HEP build tool / package manager

```
for (auto p:
particles) {...
```



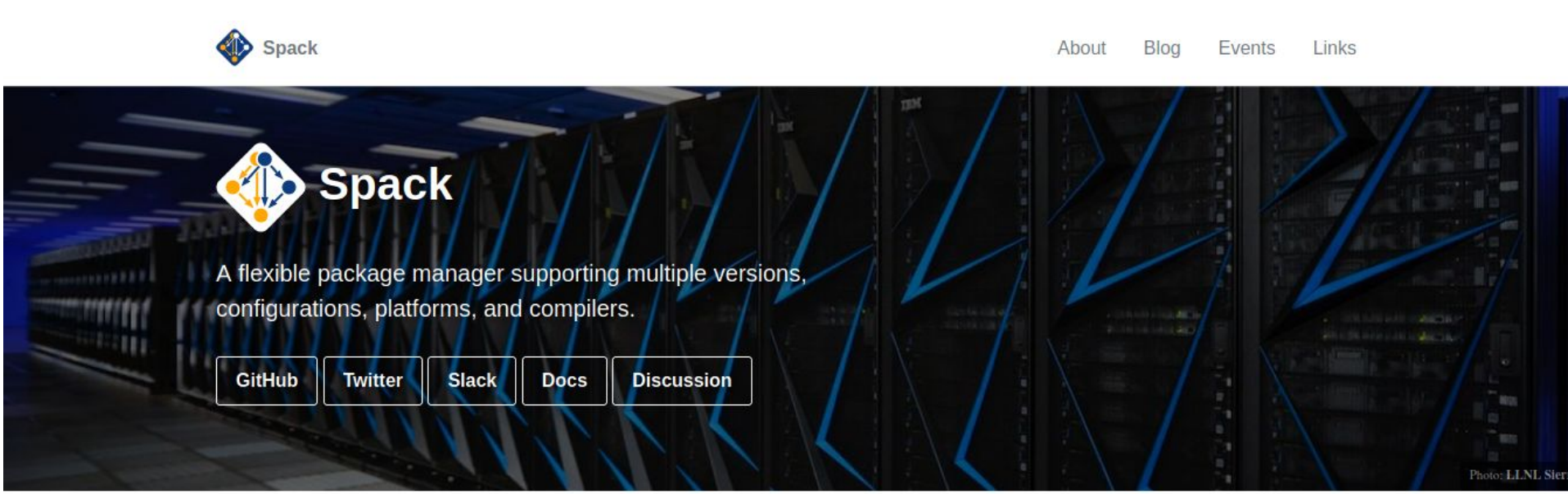
- Goal: efficiently expoit computing resources
  - Make the most out of what is available: take advantage modern architectures if possible
- Goal: good tooling for **Users**, **Developers** and **Librarians**
- Goal: Deal with complex software stacks
  - Variety of build systems, patches, platforms ...



Spack for HEP

- Originally written for/by HPC community
  - Emphasis on dealing with **multiple configurations** of the same packages
    - Different versions, compilers, external library versions …
    - … may coexist on the same system
- Not the only solution in this problem space:
  - EasyBuild
  - Nix/Guix
  - Conda
  - Gentoo Prefix

See https://spack.io/files/spack-rd100-2019-final.pdf for a nice, if possibly biased, comparison



Welcome to Spack!

Spack is a package manager for supercomputers, Linux, and macOS. It makes installing scientific software easy. Spack isn't tied to a particular language; you can build a software stack in Python or R, link to libraries written in C, C++, or Fortran, and easily swap compilers or target specific microarchitectures. Learn more here.

spack.io

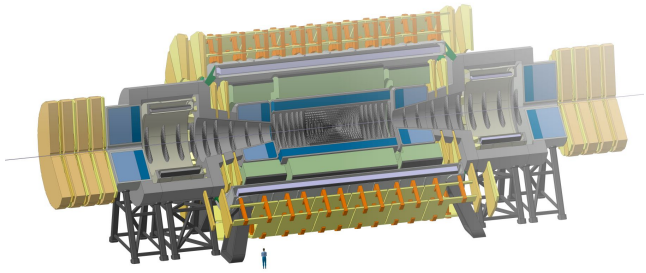
- Many more topics than could be covered here
  - CDash, containers, gitlab ci integration, environments, …
- Refer to the documentation or the recent CPPCon lightning talk
  - Also, presentation by Ben Morgan at pre-GDB software deployment meeting
- Slack channel and mailing list for informal discussions
- Github repository with a lot of activity

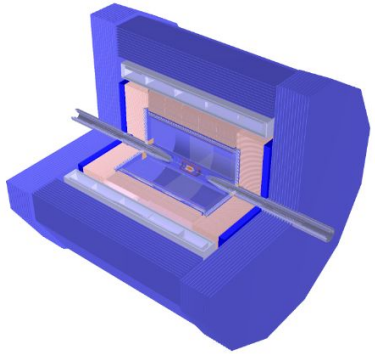
Previously...

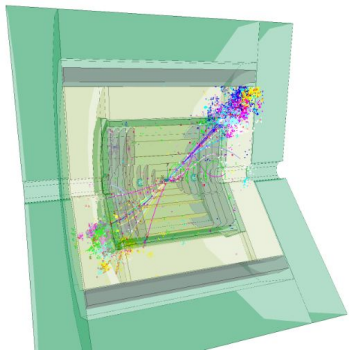
CHEP 2019:

- SpackDev: Multi-Package Development with Spack
  - … now part of upstream spack!
- Modern Software Stack Building for HEP
  - … still mostly built against externals, first steps towards a full spack build

Key4HEP



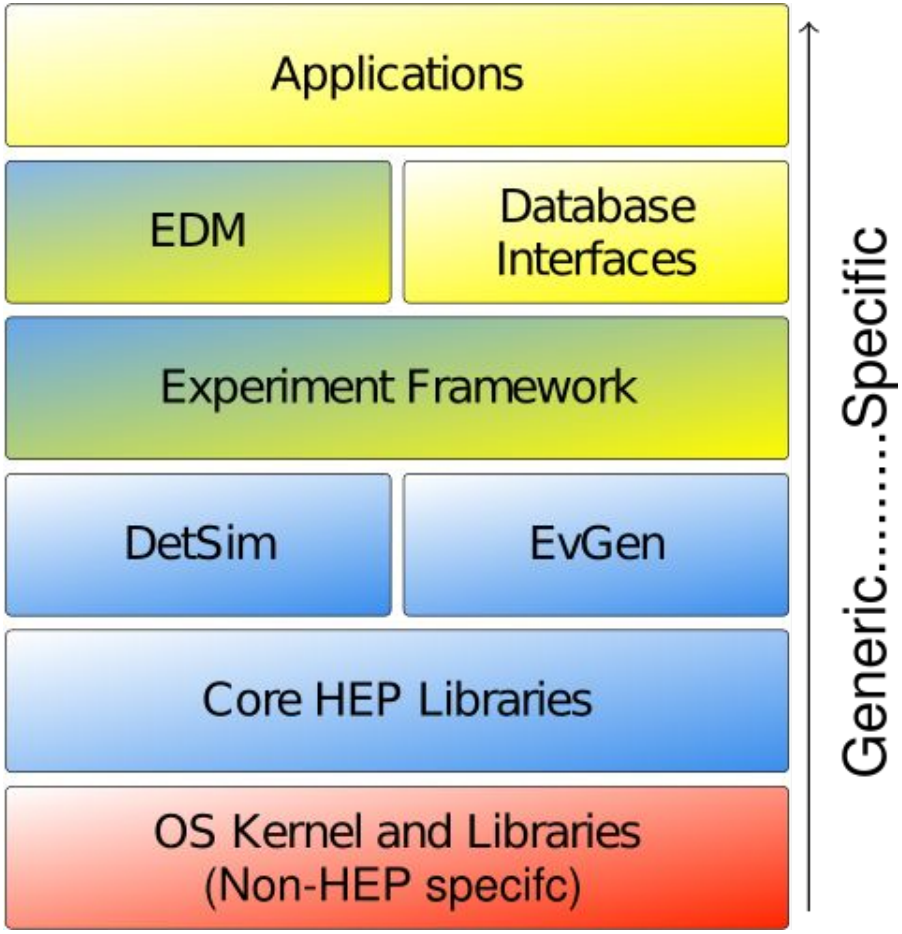




- Common Software for ***STC/SCT, FCC, ILC/CLIC, CEPC***
  - built with spack

Stack with focus on detector simulations:

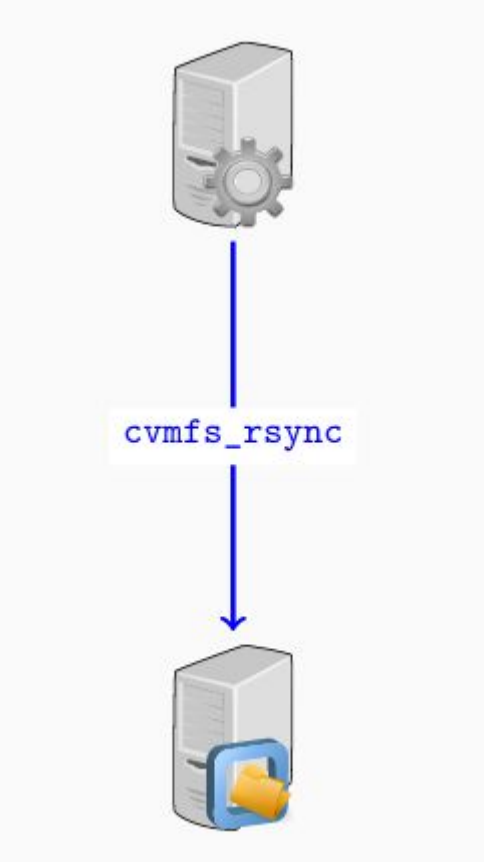
  - 60 experiment packages
  - 50 HEP libraries
  - ~14 GB install size
  - ~6h to build (4 cores)



CVMFS Deployment to /cvmfs/sw.hsf.org

Several possible workflows (see presentation by Jakob Blomer)

1. The Postscript relocation approach
2. The rsync approach
3. The Gateway approach
4. The Container approach



The rsync approach:

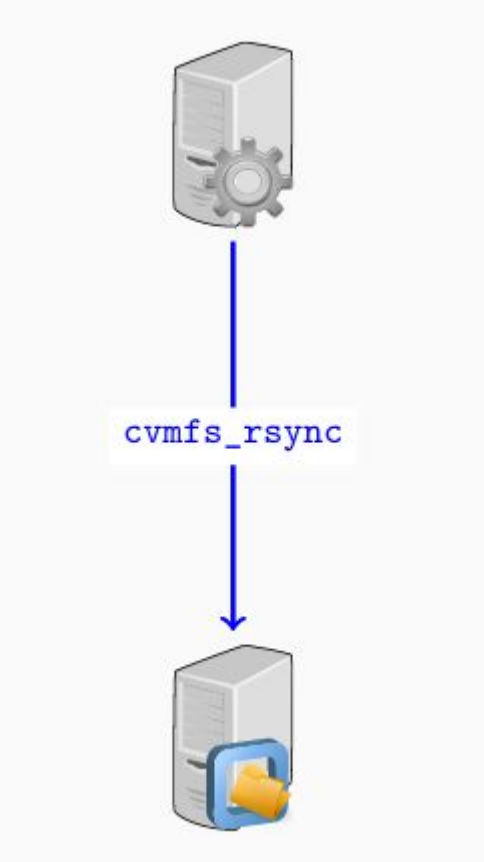
- Builder mounts a read/write copy of the /cvmfs tree
- Builder changes/installs software in place
- Publisher uses rsync to pull changes from the builder

Non-trivial to maintain multiple, synchronized publishers

CVMFS Deployment to /cvmfs/sw.hsf.org

Use Gitlab-CI with a few dedicated openstack build machines to automate deployment.

Ongoing developments in CVMFS like cvmfs_rsync are very helpful



The rsync approach:

- Builder mounts a read/write copy of the /cvmfs tree
- Builder changes/installs software in place
- Publisher uses rsync to pull changes from the builder

Non-trivial to maintain multiple, synchronized publishers

Spack: use for developers

Options to use spack for developing software:

- Spack can build from branches.
- Build can be done “as usual” after
  - `spack load`
  - `spack build-env`
- `spack dev-build` compiles local code according to the spack recipe
- `spack develop` allows dev-builds of several packages

Dedicated *development-environments* channel on slack.

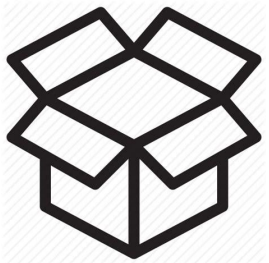
Recent Improvements

/cvmfs/sw.hsf.org/key4hep

- Spack distributable on read-only filesystem and installed on cvmfs
- Automated release workflow (gitlab-ci.yaml) with testing before deployment

- Build against Gentoo Prefix (usable on any* linux distribution)
- Declaration of geant4 data packages as “external” to allow sharing between installation
- Efficient nightly builds (use git commit as version)
- New concretizer already shows more robust concretization (variant forwarding)
- Automatic Generation of a setup.sh / setup.csh

Outlook and Open Issues



- “Spack buildcache” in principle allows to create “binary packages”
  - Mostly this is a tarball of the installation directory with tooling for relocation
  - For a lot of packages, relocation is fairly slow and introduces failure modes
- Spack-build Glibc and compiler dependencies
- Further improvements to the concretizer
- Work started on better `spack external` support for HEP packages
- Proper versioning of spack itself

Issues - Towards the LCG Releases

- Prototype of LCG_99 available
  - https://gitlab.cern.ch/sft/sft-spack-repo
  - /cvmfs/sw.hsf.org/sft-spack
- Validation required
  - Differences in the Build Configuration can lead to non-obvious errors, even at runtime
  - Key4hep already validated a good fraction
- Some packages still missing
  - … but continuously added
- Distribution of binaries (without cvmfs) still to be figured out, but no blockers.

Conclusion

- Spack is a viable solution for building HEP Software
  - Strong and growing community

    (check `spack list --tags hep`)
- Complete software stack for Key4hep:
  - ***STC/SCT, FCC, ILC/CLIC, CEPC***

  is already being build with spack and in use.

  - **`source /cvmfs/sw.hsf.org/key4hep/setup.sh`**
- Crucial Improvements (nightly builds, read-only spack) added and are being pushed upstream
- Last missing packages are being added for build of LCG releases
- Prototype build under /cvmfs/sw.hsf.org/sft-spack

Backup: HEP Packages in the Spack repository

```
spack list --tags hep
```

```
                    g4ensdfstate        k4marlinwrapper    pandoraanalysis
acts                g4incl              kaldet             pandorapfa
aida                g4ndl               kaltest            pandorasdk
aidatt              g4neutronxs         kassiopeia         photos
ccs-qcd             g4particlexs        key4hep-stack      physsim
ced                 g4photonevaporation kitrack                podio
cedviewer           g4pii               kitrackmarlin      py-gosam
cepcsw              g4radioactivedecay larcontent   py-hepdata-validator
clhep               g4realsurface       lccd               py-hepunits
clicperformance     g4saiddata          lccontent          py-particle
clupatra            g4tendl             lcfiplus           py-uproot4
```

collier	garlic	lcfivertex	pythia6
conddbmysql	gaudi	lcgeo	pythia8
conformaltracking	geant4	lcio	qd
dd4hep	geant4-data	lctuple	qgraf
ddkaltest	geant4-vmc	lhapdf	raida
ddmarlinpandora	gear	lich	relax
delphes	generalbrokenlines	madgraph5amc	rivet
dire	gosam-contrib	marlin	root
dual-readout	guinea-pig	marlindd4hep	syscalc
edm4hep	hepmc	marlinfastjet	tauola
evtgen	hepmc3	marlinkinfit	thepeg
fastjet	hepmcanalysis	marlinpandora	tricktrack
fcalclusterer	heppdt	marlinreco	vbfnlo

```
fcc-edm          heputils            marlintrkprocessors  vecgeom
fccsw            herwig3             marlinutil           vgm
fjcontrib        herwigpp            mcutils              whizard
forwardtrackingilcutil               njet                 yoda
g4abla           ildperformance      openloops
g4emlow          k4fwcore            overlay
```

CVMFS directory tree

Already mounted in most places -
Try it out on lxplus!

```
/cvmfs/sw.hsf.org/key4hep/
|-- spackages /  $platform / $compiler / $pkgname-$spackhash / (bin  … )
|-- views   / $K4_version  / $platform / (bin include share … init.sh)
|-- setup.sh
|-- contrib
|-- share/data

/cvmfs/sw-nightlies.hsf.org/key4hep/
|-- spackages/ $platform /  $compiler /  $pkgname-$spackhash / (bin  … )
|-- views   / $timestamp  / $platform / (bin include share … init.sh)
|-- setup.sh
```

Build Systems, CMake HSF Template etc

- Standardized and well configured package-level build tools like CMake help a lot to reduce maintenance burden
  - Good: explicitly declared build options instead of `if(PACKAGE_FOUND)`
  - Further HSF best practices:
    - **HSF-TN-2020-01**
- CMake is the clear favorite for modern C++ packages
- Only a few packages that have diy or exotic build systems
  - Openloops (scons), ...
- The HSF project template (https://github.com/HSF/tools/) should continue to be used!

Spack for Key4HEP

- Spack is a package manager
  - Does not replace CMake, Autotools, …
  - Comparable to apt, yum, homebrew, ...
    - But not tied to operating system
    - And no central repository for binaries!
- Originally written for/by Super-Computing (HPC) community
  - Emphasis on dealing with **multiple configurations** of the same packages
    - Different versions, compilers, external library versions …
    - … may coexist on the same system
  - Spread out to different communities, including HEP, which is well integrated by now
    - For a project besides Key4HEP, see W. Deconinck's work for eic: https://github.com/eic

Requirements for a Build System

[ ] Need to be able to scale to a typical experiment software stack

[ ] Combinatorics of multiple platforms, versions, Release/Debug ...

[ ] Easy deployment to CVMFS

[ ] Allow local builds independent of central CVMFS installations

[ ] Support software development usecases

The KEY4HEP stack contains some 300 packages

- 60 Experiment-specific
- 50 HEP-specific
- 200 System/General Purpose

14 GB install size, some 6h to build on single 4-core machine

Microarchitectures

- Package specs have “architecture” attribute: platform-os-target
  - E.g. linux-centos7-skylake
- Aware of both generic families, e.g. x86_64, and specific implementations, e.g. skylake (Intel), bulldozer (AMD)
  - https://spack.readthedocs.io/en/latest/basic_usage.html#support-for-specific-microarchitectures
- Architecture is used in install_path_scheme, the directory layout template for installed packages, so they can be distinguised

```
$ spack arch
linux-centos7-skylake
$ spack find
==> 2 installed packages
-- linux-centos7-ivybridge / gcc@8.3.0 ---------------------------
zlib@1.2.11

-- linux-centos7-skylake / gcc@8.3.0 -----------------------------
zlib@1.2.11
$
```

- In practice, less broadly compatible than lcg release, leading to possible “Illegal Instruction” errors on older CPUs
- See also: https://github.com/archspec/archspec

Spack Specs

- Core concept in Spack is that every package has a “spec” that describes how it should be, or was, built, including
  - Version, compiler-version built with, options used (“variants”)
  - Those parameters for each dependency used down the dependency graph
- Taking an example from the Spack docs
  - `spack install root@6.20.04:6.22.0%gcc@9.3.0 arch=linux-ubuntu20.04-broadwell +tmva ^python@3.8.2`
  - Means: *Install root at some version between 6.20.04 and 6.22.0 (inclusive) with the tmva build option enabled, built using gcc at version 9.3.0 for the broadwell architecture, Additionally, it says to use python version 3.8.2 to build root*
  - Variants and build options are defined in the package recipe (python file hosted in upstream spack or a dedicated repo)

Spack Specs

This is an abstract spec; spack concretizes it to obtain a hashable, concrete one:

```
root@6.22.0%gcc@9.3.0~aqua+davix~emacs+examples~fftw~fits~fo
rtran+gdml+gminimal~graphviz+gsl~http~ipo~jemalloc+math~mems
tat+minuit+mlp~mysql+opengl~postgres~pythia6+pythia8+python~
qt4+r+root7+rootfit+rpath~shadow+sqlite+ssl~table+tbb+thread
s+tmva+unuran+vc+vdt+vmc+x+xml+xrootd
build_type=RelWithDebInfo cxxstd=17
patches=22af3471f3fd87c0fe8917bf9c811c6d806de6c8b9867d30a1e3
d383a1b929d7 arch=linux-ubuntu20.04-broadwell

    ^cmake@3.18.4%gcc@9.3.0~doc+ncurses+openssl+ownlibs~qt
patches=bf695e3febb222da2ed94b3beea600650e4318975da90e4a71d6
f31a6d5d8c3d arch=linux-ubuntu20.04-broadwell

        ^ncurses@6.2%gcc@9.3.0~symlinks+termlib
arch=linux-ubuntu20.04-broadwell

...
```

Spack for Key4HEP

- Installation of the full software in 6 commands:

```
git clone https://github.com/spack/spack.git
source spack/share/spack/setup-env.sh

git clone https://github.com/key4hep/key4hep-spack.git
spack repo add key4hep4-spack

cp  key4hep-spack/config/packages.yaml spack/etc/spack

# install the meta-package for the key4hep-stack
spack install key4hep-stack
```

1. Setup upstream spack. The “installation” is just a git clone. Very limited dependencies (python, git, curl)

Spack for Key4HEP

- Installation of the full software in 6 commands:

```
git clone https://github.com/spack/spack.git
source spack/share/spack/setup-env.sh

git clone https://github.com/key4hep/key4hep-spack.git
spack repo add key4hep4-spack

cp  key4hep-spack/config/packages.yaml spack/etc/spack

# install the meta-package for the key4hep-stack
spack install key4hep-stack
```

2. Setup key4hep package recipes. Currently 44 packages - with standard build system fairly quick to create and maintain new ones.

Spack for Key4HEP

- Installation of the full software in 6 commands:

```
git clone https://github.com/spack/spack.git
source spack/share/spack/setup-env.sh

git clone https://github.com/key4hep/key4hep-spack.git
spack repo add key4hep4-spack

cp  key4hep-spack/config/packages.yaml spack/etc/spack

# install the meta-package for the key4hep-stack
spack install key4hep-stack
```

3. Set some preferred package configuration options(build type etc.), and give hints to the concretizer to avoid errors.

Spack for Key4HEP

- Installation of the full software in 6 commands:

```
git clone https://github.com/spack/spack.git
source spack/share/spack/setup-env.sh

git clone https://github.com/key4hep/key4hep-spack.git
spack repo add key4hep4-spack

cp  key4hep-spack/config/packages.yaml spack/etc/spack

# install the meta-package for the key4hep-stack
spack install key4hep-stack
```

4. Use a bundle package to install all packages needed. Environments can be used for installations with different build types or other specialisations.

Spack for Key4HEP

- Nightly and release builds on CVMFS:

  ```
  source /cvmfs/sw.hsf.org/key4hep/setup.sh
  ```

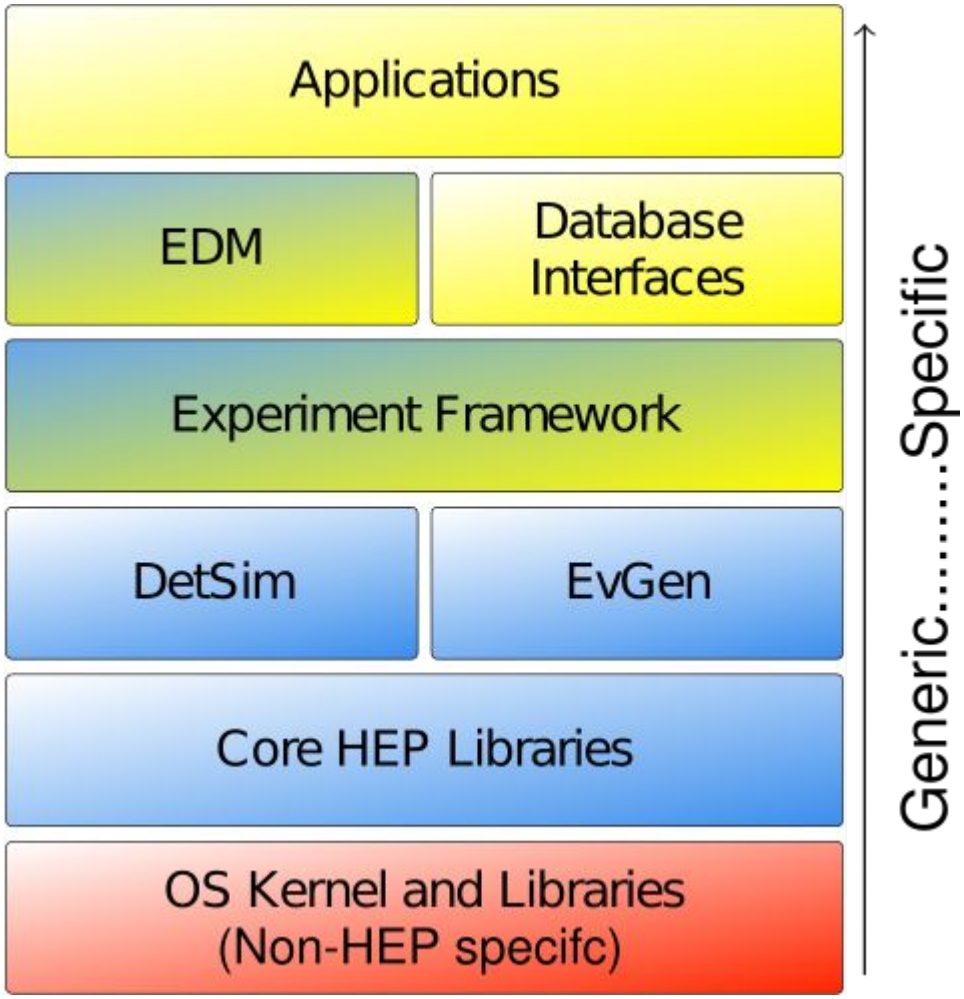
  ```
  source /cvmfs/sw-nightlies.hsf.org/key4hep/setup.sh
  ```

  Not yet using a filesystem view, instead setting PATH/LD_LIBRARY_PATH/… for each package.

- New developments in Key4HEP (to be pushed upstream):
  - Setup script prepends to the user environment and is installed automatically with the bundle package
  - For nightly builds, packages can be installed with commit hashes as versions and do not need to be rebuilt if nothing changes

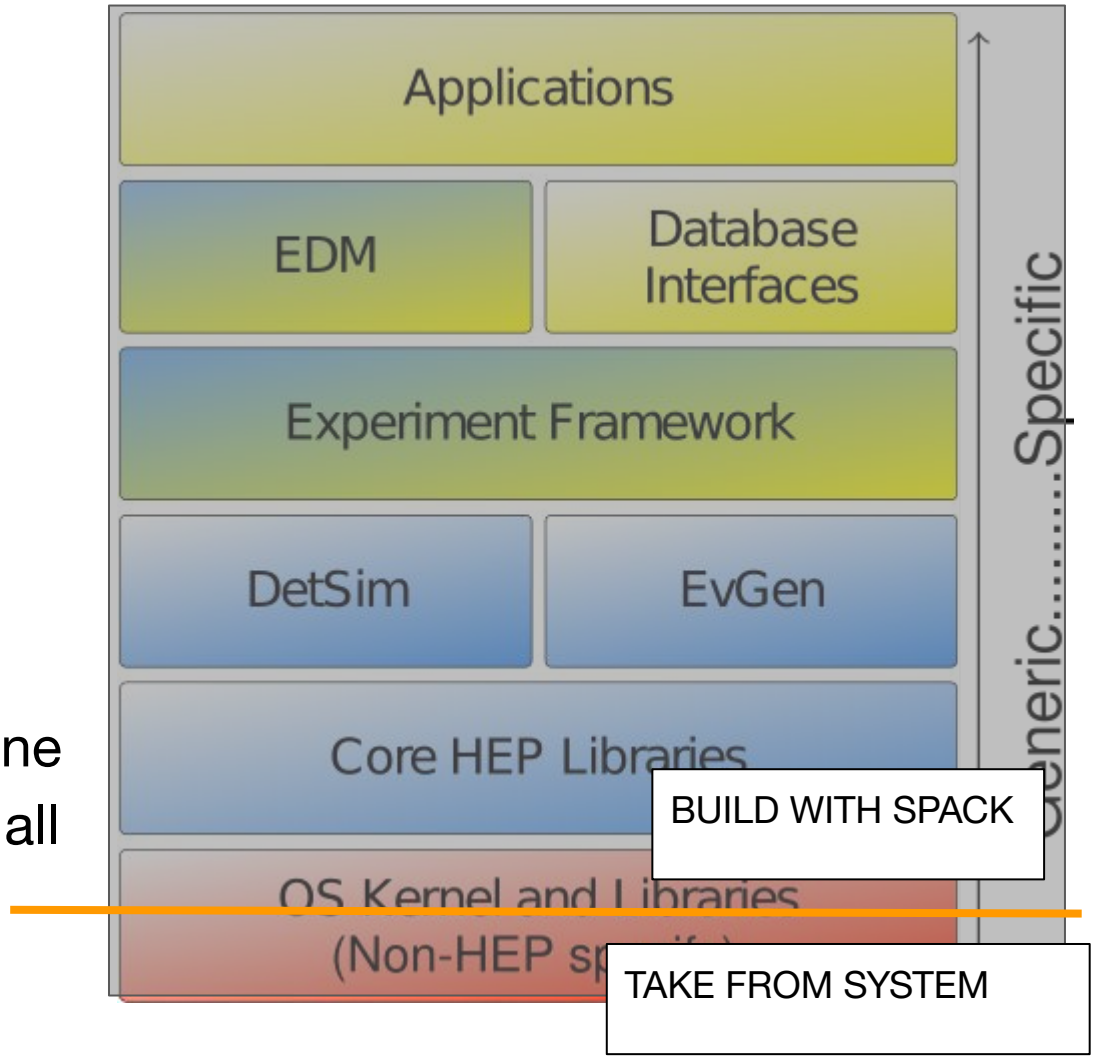
System packages

- Where to draw the line between system and experiment software?
- For the LCG releases, efforts have been made to formalise this (HEPOSLibs)
- For spack, we try to build as much as possible



System packages 2

- Where to draw the line between system and experiment software?
- In the future, it may be possible to even build glibc with spack.
  - “Containerization”, would allow to use one “distroless” build on all linux platforms



System packages 3

- Where to draw the line between system and experiment software?
- Tried an install that uses more external packages for arch linux - works, but will not support