

CERN R&D on Spack and the Turnkey Stack

HSF Software Workshop - 2020/11/19

Valentin Volkl (CERN), Key4hep Software Group

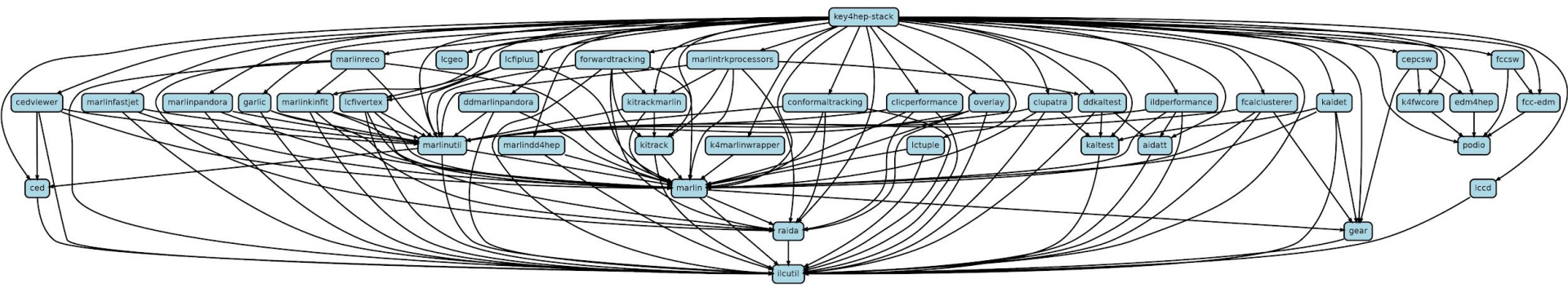
Thanks to Andre Sailer, Pere Mato, Ben Morgan for use of Slides

Table of Contents

- **Key4HEP - Introduction and motivation**
- **The Spack package manager**
- **Technical considerations**
- **Deployment to CVMFS**
- **Workflow -Spack for Development**

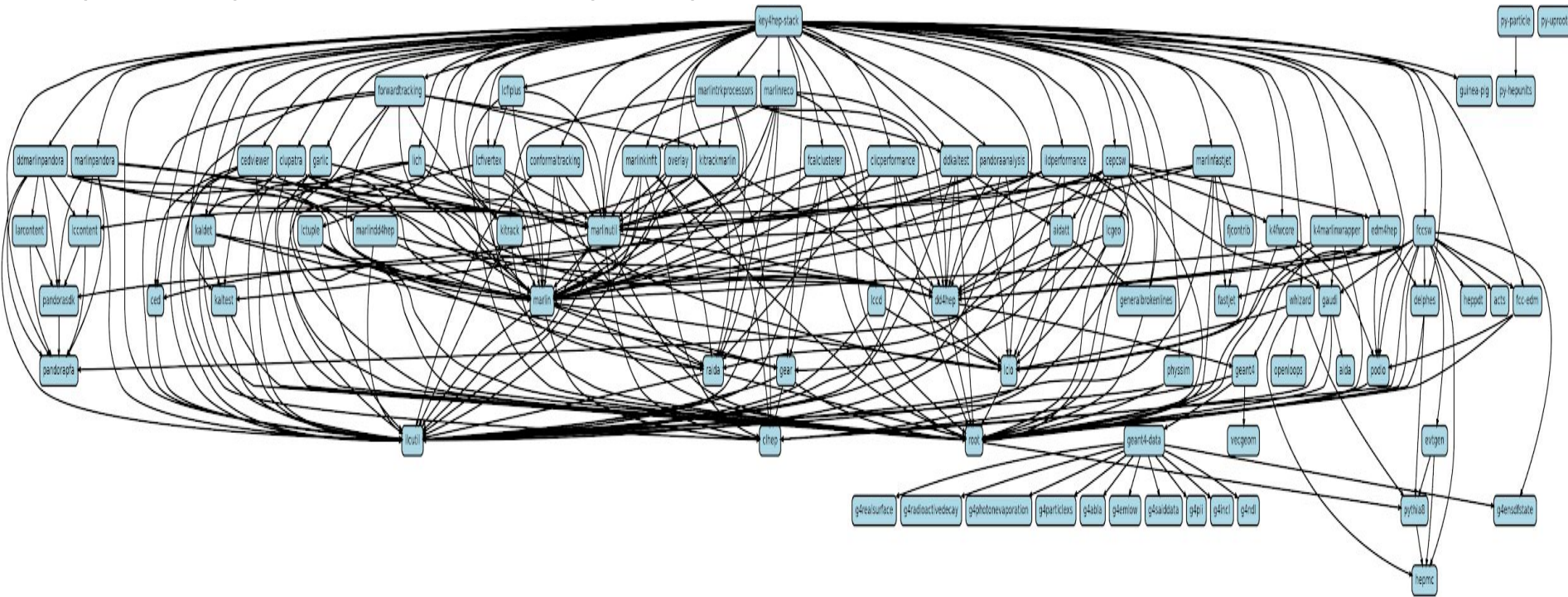
Dependency Graph of Key4HEP Stack

Experiment-specific Packages



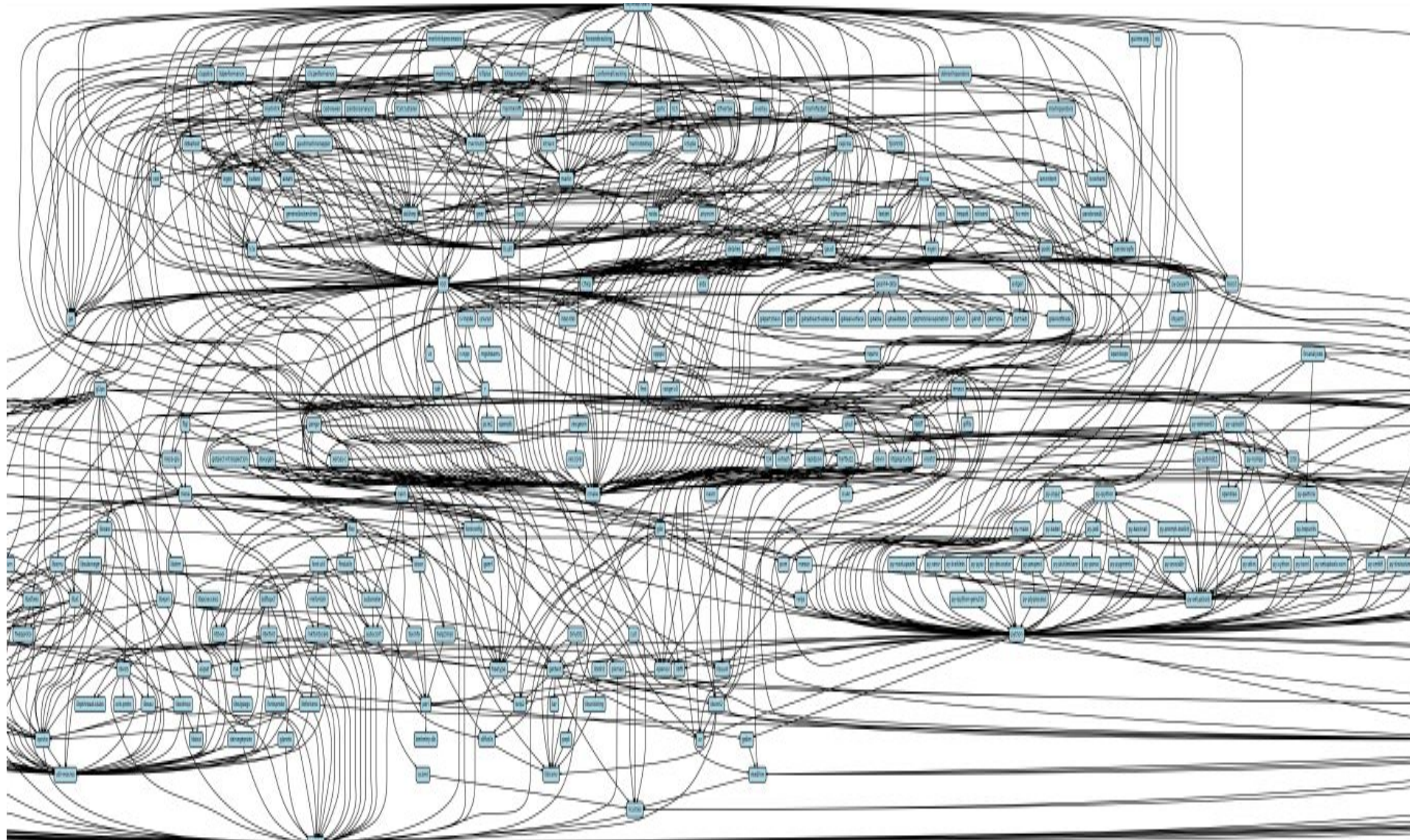
Dependency Graph of Key4HEP stack

Experiment-specific Packages + HEP-specific packages



Dependency Graph of Key4HEP stack

Experiment-specific Packages + HEP-specific Packages + General Purpose Libraries



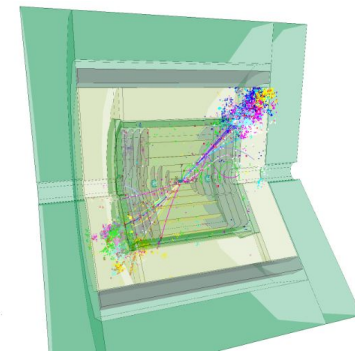
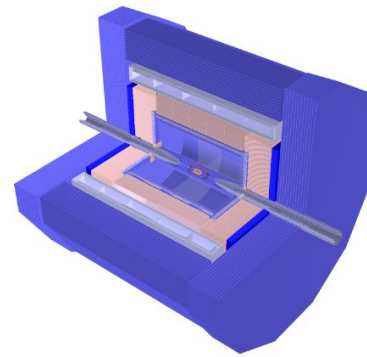
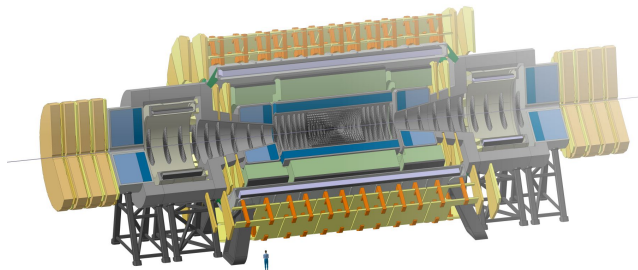
Requirements for a Build System

- [] 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



- Future detector studies critically rely on **well-maintained software stacks** to model detector concepts and to understand a detector's limitations and physics reach
- We have a scattered landscape of **specific software tools** on the one hand and **integrated frameworks** tailored for a specific experiment on the other hand
- Aim at a low-maintenance common stack for FCC, ILC/CLIC, CEPC with ready to use “plug-ins” to develop detector concepts
- Reached consensus among all communities for future colliders to develop a **common turnkey software stack** at recent [Future Collider Software Workshop](#)
- Identified as an important project in the CERN [EP R&D initiative](#)



- 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 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>



- 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



A flexible package manager supporting multiple versions, configurations, platforms, and compilers.

[GitHub](#)[Twitter](#)[Slack](#)[Docs](#)[Discussion](#)

Photo: LLNL Stern

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

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~fortran+gdm+gminimal~graphviz+gsl~http~ipo~jemalloc+math~memstat+minuit+mlp~mysql+opengl~postgres~pythia6+pythia8+python~qt4+r+root7+rootfit+rpath~shadow+sqlite+ssl~table+ttb+threads+tmva+unuran+vc+vdt+vmc+x+xml+xrootd
build_type=RelWithDebInfo cxxstd=17
patches=22af3471f3fd87c0fe8917bf9c811c6d806de6c8b9867d30a1e3d383a1b929d7 arch=linux-ubuntu20.04-broadwell
```

```
^cmake@3.18.4%gcc@9.3.0~doc+ncurses+openssl+ownlibs~qt
patches=bf695e3febb222da2ed94b3beea600650e4318975da90e4a71d6f31a6d5d8c3d arch=linux-ubuntu20.04-broadwell
```

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

...

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 distinguished

```
$ 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>



- 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)



- 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.



- 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.



- 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.



- 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.

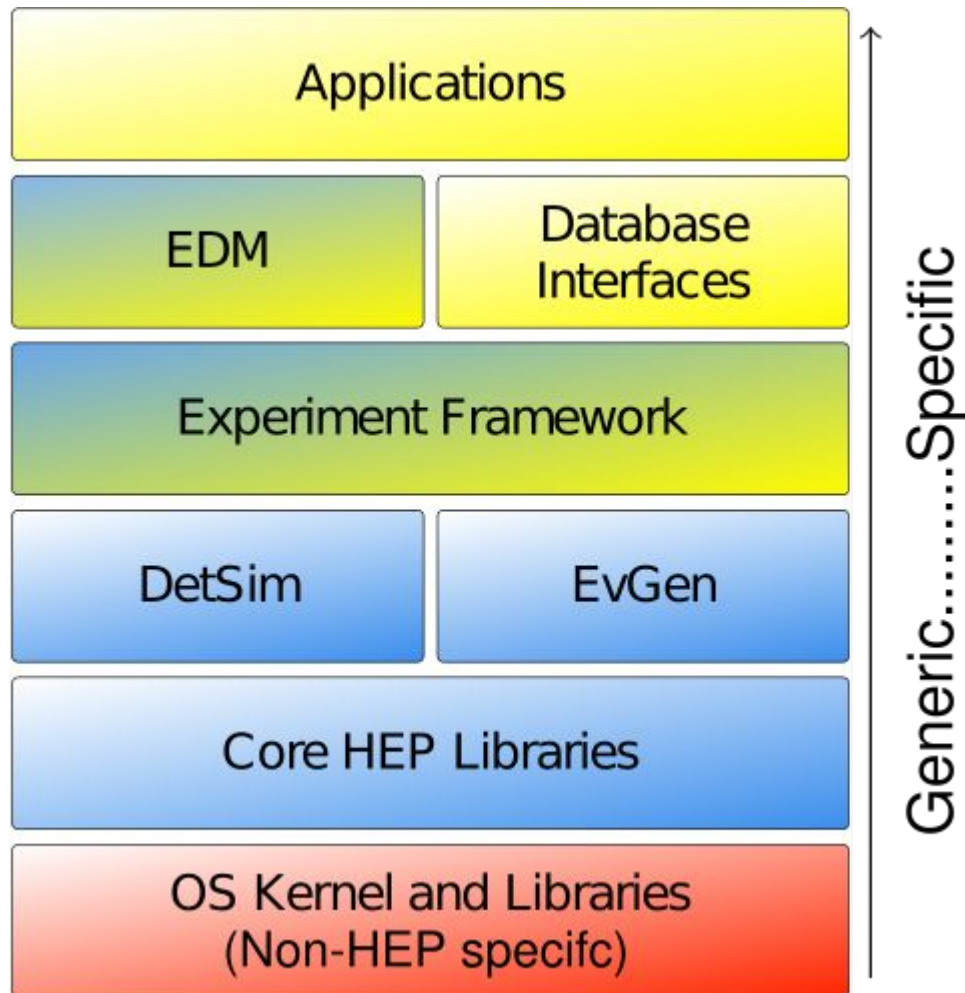
- New developments in Key4HEP (to be pushed upstream):
 - Setup script prepends to the user environment and is installed automatically with the bundle package
 - For nightly builds, packages can be installed with commit hashes as versions and do not need to be rebuilt if nothing changes

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!

System packages

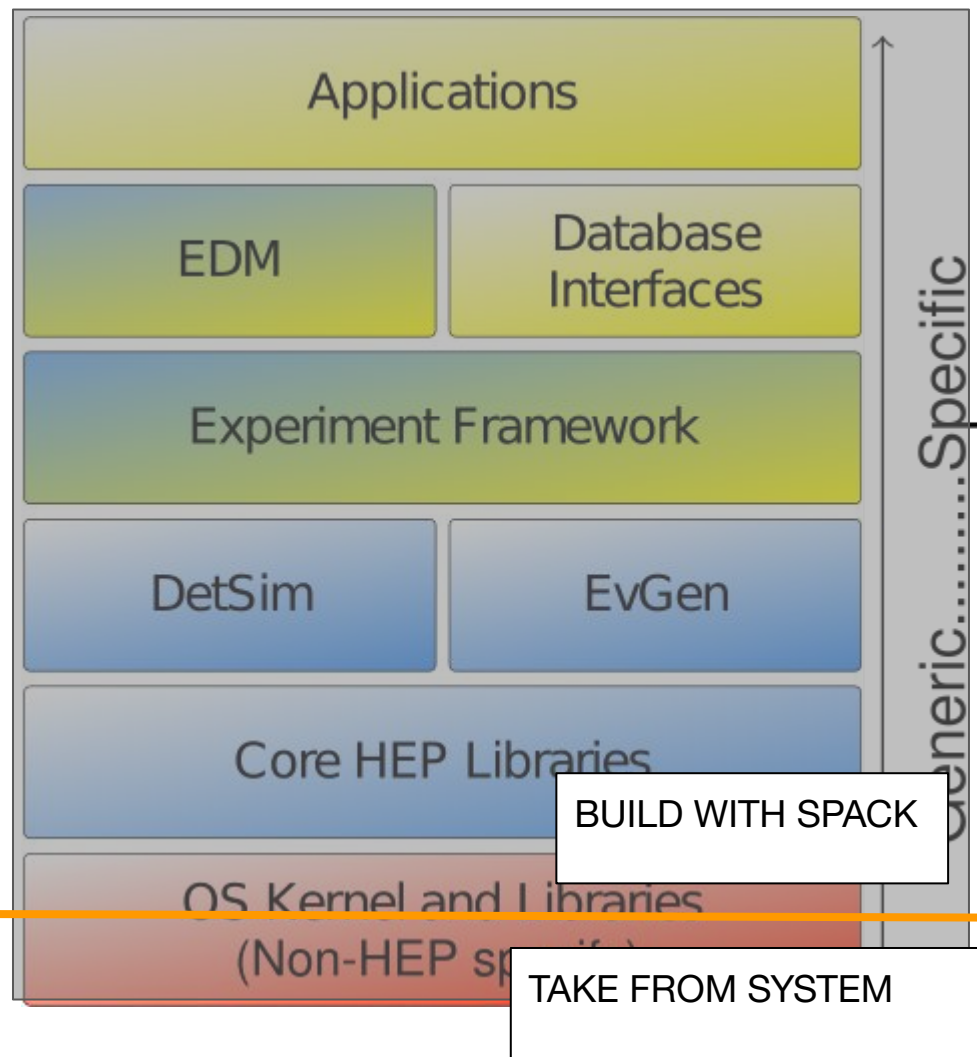
- Where to draw the line between system and experiment software?
- For the LCG releases, efforts have been made to formalise this ([HEPOS Libs](#))
- For spack, we try to build as much as possible



System packages 2

CENTOS7

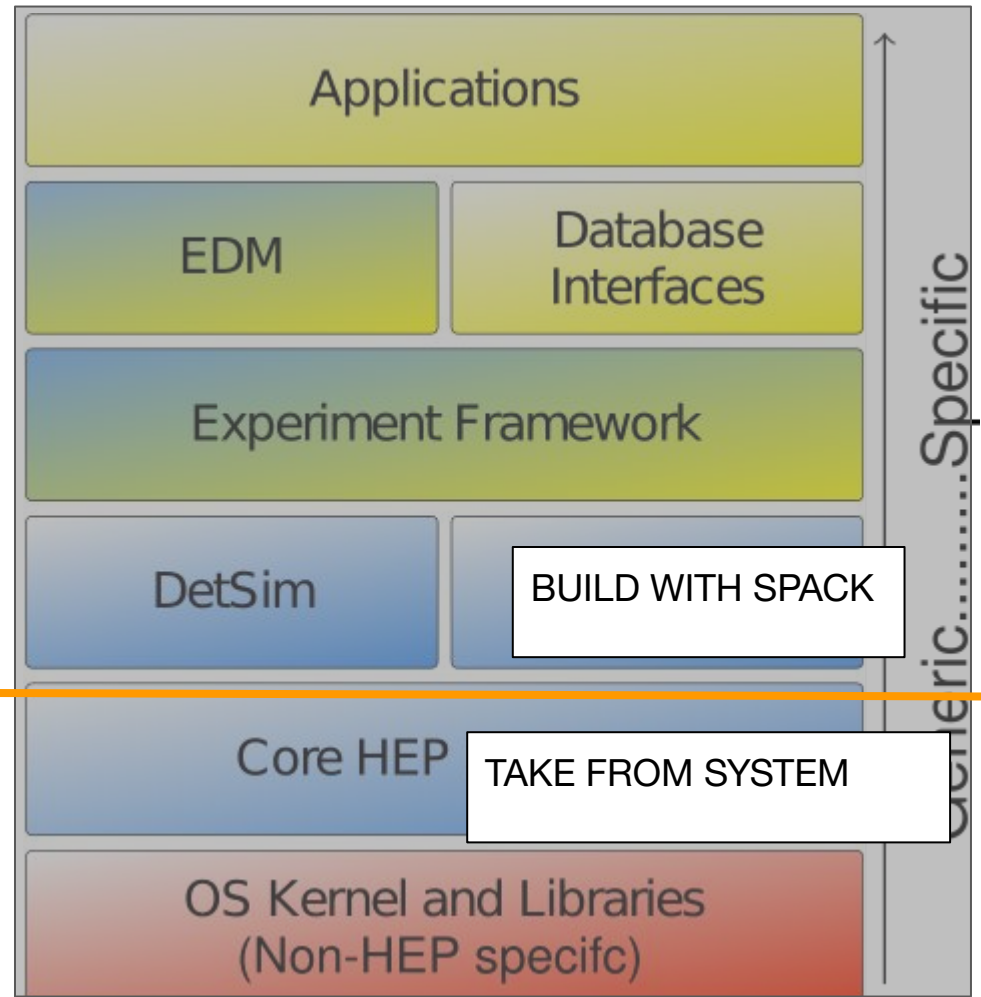
- Where to draw the line between system and experiment software?
- In the future, it may be possible to even build glibc with spack.
 - “Containerization”, would allow to use one “distroless” build on all linux platforms



System packages

Arch Linux

- 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



CVMFS Deployment

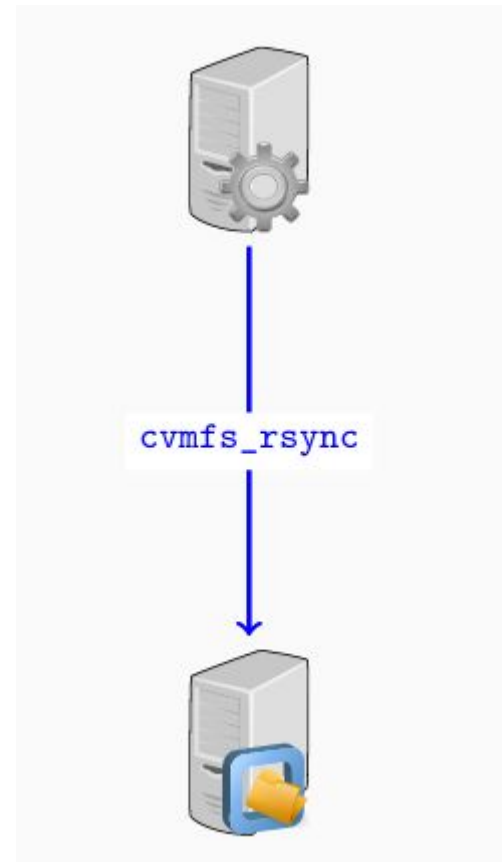
Several possible workflows (see [presentation by Jakob Blomer](#))

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

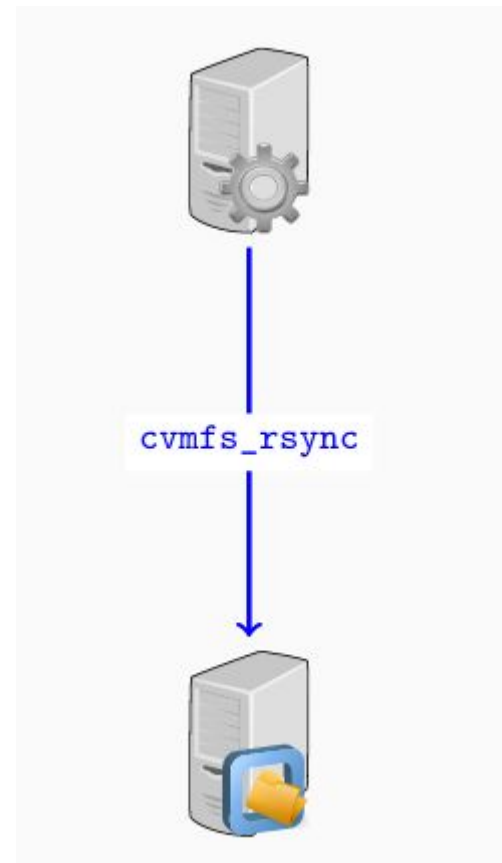
Use [Gitlab-CI](#) with a few dedicated openstack build machines to automate deployment.

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





CVMFS directory tree

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

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

Spack: use for developers



Use in developing software is pushing spack's intended purpose, but possible. Options:

- 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.

Issues - Towards the LCG Releases

- Ivan Razumov (SFT) has added a great amount of packages - maybe 70% of packages included in the lcg releases already available in spack
- Key4HEP installation can be used as a test-bed
- For both CLICSW and [FCCSW](#) there are now builds based both on LCG releases and on Spack
 - Lets us compare and validate the installation
- Differences in the Build Configuration can lead to non-obvious errors, even at runtime (all already fixed):
 - The default units for HEPMC are set at compile time, and can lead to wrong outputs if user code assumes one convention
 - If the “netaccessor” of Xerces-C (XML library) is disabled, certain Geant4 codes will throw a runtime exception because they cannot validate the gdml schema.
 - The Gaudi plugin system requires the runtime environment even at compile time.

Buildcaches, Relocation



- “Spack buildcache” in principle allows to create “binary packages”
 - Mostly this is a tarball of the installation directory with tooling for relocation
- A lot of HEP packages are, however, just not relocatable
- Even if they are, the relocation is fairly slow and introduces failure modes.
- Moving packages as tarballs without relocation (to the same path on a different machine), is definitely feasible and can replace rsync for cvmfs deployment



Personal Experiences

- Collaboration with Spack developers very smooth, and a growing community of HEP contributors
 - Some HEP colleagues have merging rights on the spack repo
 - Some HEP packages actively maintain their package recipes (ACTS!)
- Rapid pace of changes in upstream repository
 - Stable builds will need to pin the spack version used.
 - But miss out on the latest features.
- Spack developers very responsive, but roadmap sometimes a bit opaque:
 - The *concretizer* developments have been much delayed
- The recipes are very nice to persistify build system know-how

```
conflicts("%gcc@8.3.1",  
    msg="There are known issues with compilers from redhat's devtoolsets" \  
    "which are therefore not supported." \  
    "See https://root-forum.cern.ch/t/devtoolset-gcc-toolset-compatibility/38286")
```

Conclusion

- Key4hep: Joint software developments between *STC/SCT, FCC, ILC/CLIC, CEPC*
- Common software stack is already being build with spack and in use.
 - Still relatively “simple” application, but moving rapidly to more platforms, bigger releases
- Developments to watch
 - “Greedy” concretizer
 - Variant forwarding
 - Compilers as dependencies
 - glibc builds

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
cliperformanceg4saiddata          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
ddmarlinpandoragear		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	