# KEY4HEP - Plans for Deployment

preGDB Workshop - 05.05.2020

Valentin Volkl (CERN)

# The Key4HEP Project

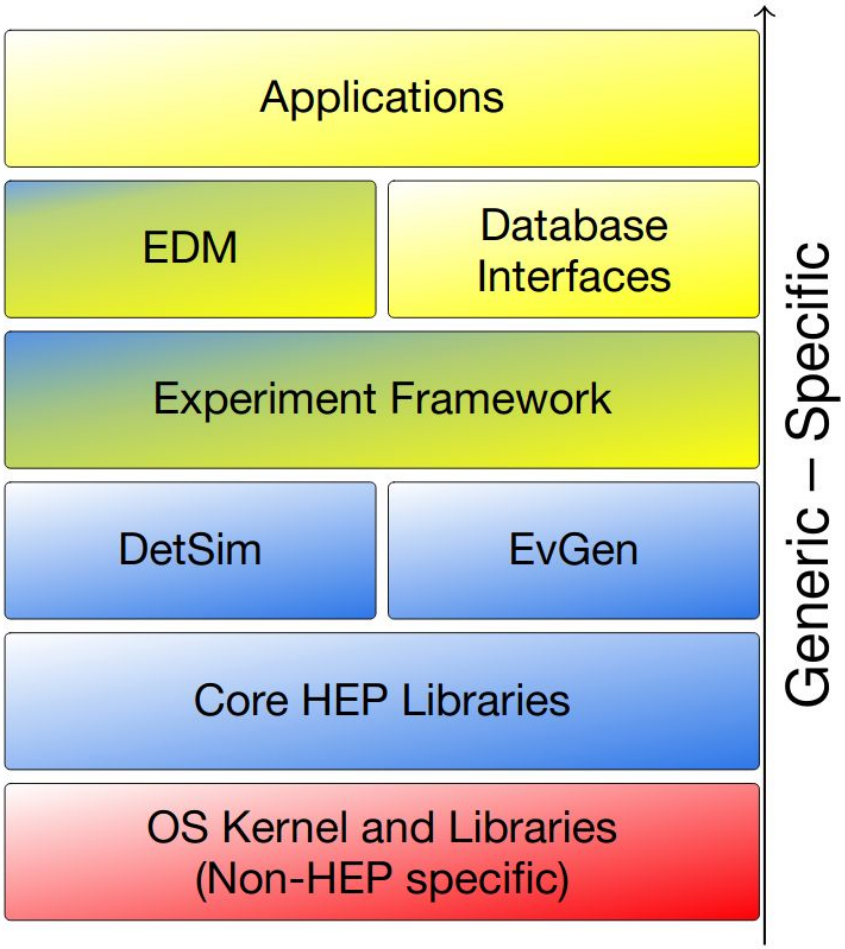
- Future detector studies critically rely on **well-maintained software stacks** to model detector concepts and to understand a detector's limitations and physics reach
- Aim at a low-maintenance common stack for FCC, ILC/CLIC, CEPC with ready to use "plug-ins" to develop detector concepts
- Reached consensus among all communities for future colliders to develop a **common turnkey software stack** at recent Future Collider Software Workshop (June 2019)
- Identified as an important project in the CERN EP R&D initiative
- Regular meetings
  - https://indico.cern.ch/category/11461/
- Docpages
  - https://cern.ch/key4hep (main documentation site)
  - https://cern.ch/edm4hep (doxygen code reference)

# Key4HEP Software Stack

- Should cover Detector and Physics Studies
  - Generation, Simulation, Reconstruction …
  - Includes most of the common libraries and projects: Geant4, ROOT, Delphes
- Where it will be used:
  - Locally
  - Batch farms
  - Grid
  - Opportunistic HPC
  - CI clouds (travis, github actions)
- Use CVMFS to deploy once and use as often as possible
- Containers currently used only to get a LCG-supported platform (currently Centos7) with CVMFS where not available
  - Especially the case on github / travis

# A typical HEP Software Stack

- Interfaces to tracking and reconstruction libraries (PandoraPFA, ACTS ...)
- (More or less) experiment specific event datamodel libraries
- Experiment core orchestration layer, which controls everything else: Marlin, Gaudi, CMSSW, AliRoot
- Packages used by many experiments: DD4hep, Pythia, ...
- Usual core libraries: ROOT, Geant4, CLHEP ...
- Non-HEP libraries: Boost, Python, CMake...



# Deployment Strategy

- Key4HEP Software is built with spack
  - Reference deployment to CVMFS
  - Additionally HTTP buildcache can be used to install binaries
  - System packages (HEPOSLibs) can be reused, but easier to use spack to build everything
- Completely independent installation possible as well
- Not clear yet if experiments want to build common packages themselves or use Key4HEP CVMFS space

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

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

# Deployment Strategy

- Key4HEP Software is built with spack
  - Reference deployment to CVMFS
  - Additionally HTTP buildcache can be used to install binaries
  - System packages (HEPOSLibs) can be reused, but easier to use spack to build everything
- Completely independent installation possible as well
- Not clear yet if experiments want to build common packages themselves or use Key4HEP CVMFS space

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

/cvmfs/sw-nightlies.hsf.org/key4hep/
|-- nightlies/ $timestamp / $platform /
|-- views    / $timestamp  / $platform /
|-- setup.sh
|-- contrib
```

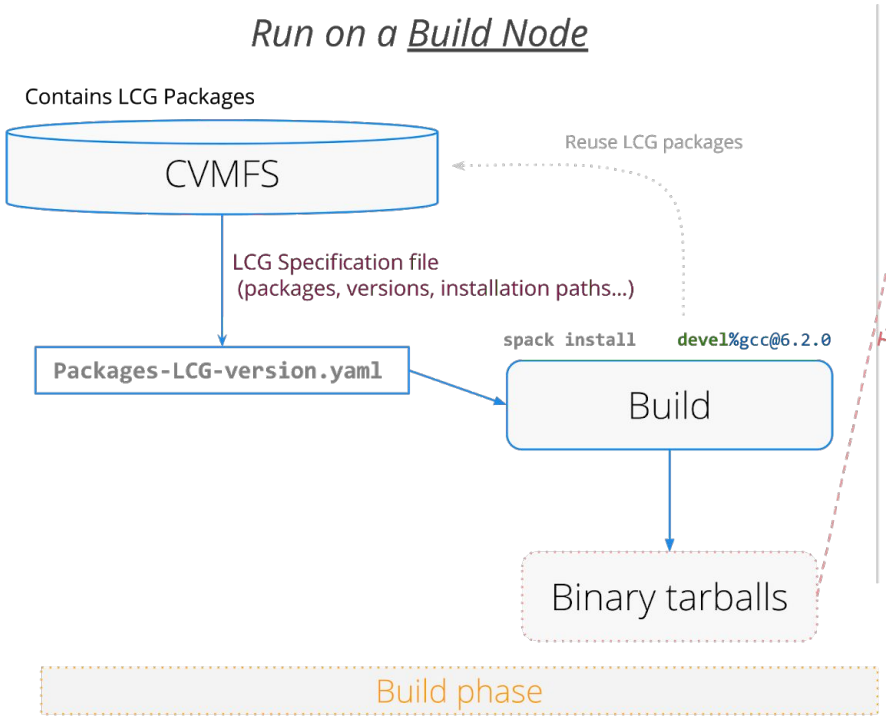
- Can use view without Spack
- Can use releases directory as a chained spack install dir
- Can use spack buildcache to locally install

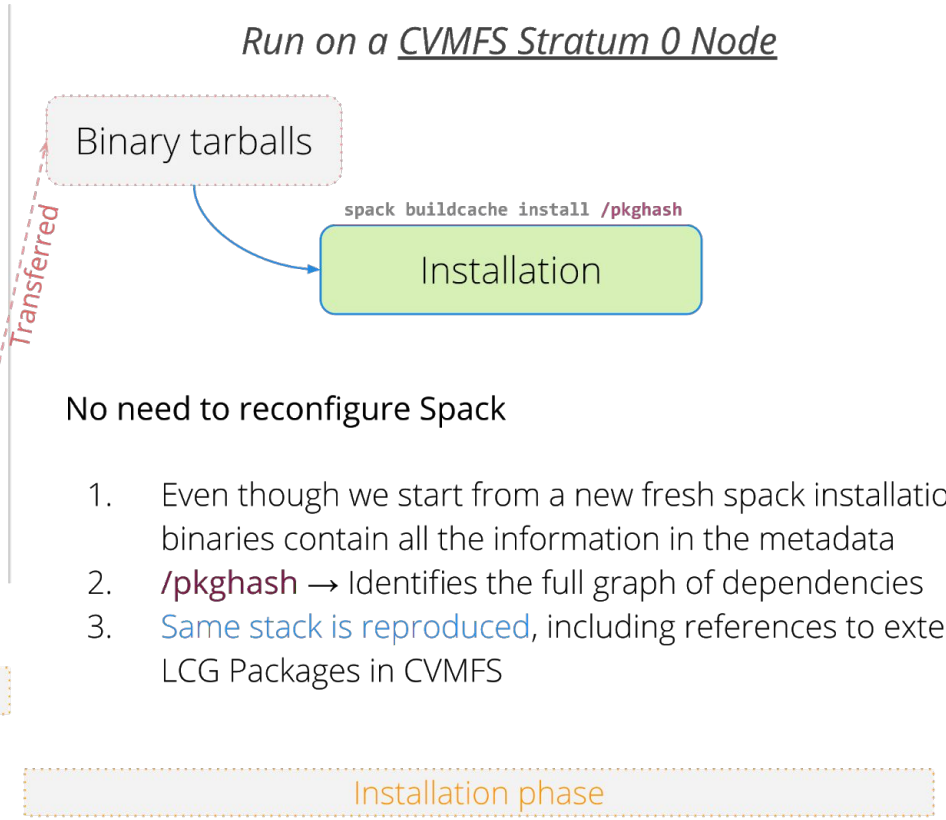
# Spack for Key4HEP

- 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 HPC community
  - Emphasis on dealing with **multiple configurations** of the same packages
    - Different versions, compilers, external library versions …
    - … may coexist on the same system
  - `Spec:` Syntax to describe package version  configuration and dependencies
- Repository added with Key4HEP package recipes

```
git clone https://github.com/spack/spack.git
git clone https://github.com/key4hep/k4-spack.git
alias spack='python $PWD/spack/bin/spack'
spack repo add k4-spack
# install the meta-package for the key4hep-stack
spack install key4hep-stack
```

# Build process




Adapted from: Javier Cervantes
CernVM Users Workshop 2019

# Production vs. Development Deployment

- CVMFS installations are great for production
- … but not ideally suited for production workflows
  - Requires internet connection
  - Difficult to change version of one dependency in views
  - No easy way to manually select packages
  - Development happens on different platform than production
- Key4HEP developers were asking for a way to build the stack without using LCG releases
- Obtaining dependencies and deployment should be the same for dev and prod

# Conclusion

- Key4HEP Software is built with spack, and binaries are distributed on CVMFS
- Covers common experiment software for physics/detector studies on top of projects/LCG releases.
- Should run on batch / grid for production and laptops / CI for development
  - CVMFS /w LCG releases on Grid sites
  - Spack allows us to be fairly flexible
- Use of Containers:
  - No fixed decision yet, will do whatever is necessary or convenient

# Interoperability

- Level 0 - Common Data Formats
  - Allows interoperability between different programs, even running on different hardware
  - E.g.: HepMC event records, LCIO, GDML, ALFA Messages
- Level 1 - Callable Interfaces
  - Basic calling interfaces defined by the programming languages, language calls possible
  - Can be dependent on the compiler and language version
  - Details are important: error/exception handling, thread safety, dependencies, runtime setup
- Level 2 - Introspection Capabilities
  - Software elements to facilitate the interaction of objects in a generic manner: Dictionaries, Scripting interfaces
  - E.g.: PyROOT to interact with any ROOT (C++) class via the python
- Level 3 - Software Frameworks
  - Software components of a common framework offer maximum re-use
  - Standard way to configure components, logging, object lifetime and

The right interoperability point between packages varies, but choosing it correctly provides great quality of life for developers and users