

Building the Key4hep Software Stack with Spack



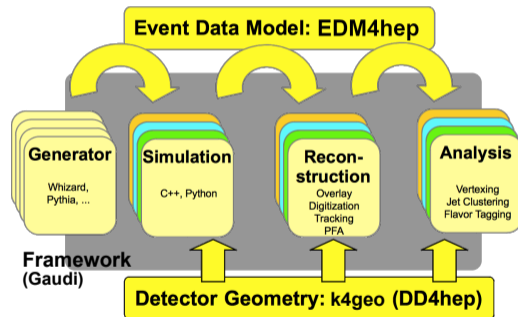
Juan Miguel Carceller j.m.carcell@cern.ch

CERN EP-SFT

October 22, 2024

Key4hep

- Turnkey software for future colliders
- Share components to reduce maintenance and development cost and allow everyone to benefit
- Complete data processing framework, from generation to data analysis
- Usable software stack, low threshold
- Community with people from future colliders: FCC, ILC, CLIC, CEPC, EIC, Muon Collider, etc.
- Open [biweekly](#) meetings



Spack

- Multi-platform package manager
- Designed to support multiple versions of packages, package configurations, and compilers
- Over 8000 recipes for packages are available, written in Python
- Configuration in yaml files
- Large community with many contributors
- Used in Key4hep for several years
- Used in some HEP experiments, for example DUNE, and HPCs



Spack

A screenshot of the Spack repository page on GitHub. The background is dark. At the top, there is a star icon followed by the text "4.3k stars". Below that is an eye icon followed by "99 watching". Then a fork icon followed by "2.3k forks". Next is a calendar icon followed by "12 years old". Below these is a green checkmark icon followed by the text "v0.22.2 (2024-09-21)" and a green pill-shaped button with the word "Latest" inside. Underneath that is the text "3 weeks ago". At the bottom of the screenshot, there is a section titled "Contributors" followed by a pill-shaped button containing the number "1,453".

Key4hep: Software

github.com/key4hep

- Many repositories under the Key4hep organization
 - [EDM4hep](#): the Event Data Model
 - [k4FWCore](#) provides the interface between EDM4hep and Gaudi
 - [k4MarlinWrapper](#) allows to call Marlin processors
 - [k4geo](#) for detector models
 - [key4hep-spack](#): for spack recipes
 - ...

The screenshot shows the GitHub organization page for Key4hep. At the top, there is a profile picture of a purple square with a white cross, the organization name "Key4hep: Turnkey Software for Future Colliders", and a link to the website "https://oem.ch/key4hep". Below this, there are three pinned repositories:

- key4hep-spack** (Public): A Spack recipe repository of Key4hep software. Language: Python. 8 stars, 19 forks.
- EDM4hep** (Public): Generic event data model for HEP collider experiments. Language: C++. 18 stars, 25 forks.
- k4FWCore** (Public): Core Components for the Gaudi-based Key4hep Framework. Language: C++. 6 stars, 21 forks.

Below the pinned repositories, there is a "Repositories" section with a search bar and filters for Type, Language, and Sort. The first repository listed is **k4EDM4hep2LcioConv** (Public): EDM4hep to LCIO Converter. Language: C++. 1 star, 9 forks, 3 releases, 1 issue. Updated 7 minutes ago. The second repository is **k4geo** (Public): DD4hep based geometry models for lepton collider detectors (Formerly known as l4geo). Language: C++. 8 stars, 36 forks, 5 releases, 2 issues. Updated 1 hour ago. The third repository is **key4hep-spack** (Public): A Spack recipe repository of Key4hep software.

Building Key4hep: Summary

- Builds are done using Spack
- Builds are deployed to CVMFS:
 - `/cvmfs/sw.hsf.org` for releases (every N months, on demand)
 - `/cvmfs/sw-nightlies.hsf.org` for nightly builds, every day
- Around 600 packages including dependencies
- Compilers are taken either from the system or installed on CVMFS
- Several flavours of builds: now AlmaLinux 9 and Ubuntu 22.04 (previously also CentOS 7) with an optimized and debug version of each build

What do we build?

- Starting point: github.com/key4hep/key4hep-spack
- About 75 packages that are not in upstream Spack
- Set of recipes for packages in Key4hep (from the repositories under github.com/key4hep) plus repositories under other organizations like [iLCSoft](#), [HEP-FCC](#) or [CEPC](#)
- Scripts for setting up the builds from CVMFS and for adding the latest version of each package to the nightlies environment
- Virtual package `key4hep-stack` that depends on the packages we want to build

- Depends on most of the Key4hep packages
- Pulls lots of package as dependencies
- Creates a `setup.sh` script that sets up the paths
- Users source this script (indirectly) to get the environment

```
depends_on("acts")
depends_on("babayaga")
depends_on("bdsim")
depends_on("bhlumi")
depends_on("cldconfig")
depends_on("dd4hep")
depends_on("delphes")
depends_on("edm4hep")
...
```

```
$ cat /cvmfs/sw.hsf.org/key4hep/releases/2024-10-03/x86_64-almalinux9-gcc14.2.0-opt/key4hep-stack/2024-10-08-k6xtr3/setup.sh
export ACLOCAL_PATH=/cvmfs/sw.hsf.org/key4hep/releases/2024-10-03...
export CC=/cvmfs/sw.hsf.org/contrib/x86_64-almalinux9-gcc11.4.1-o...
export CLDCONFIG=/cvmfs/sw.hsf.org/key4hep/releases/2024-10-03/x8...
export CMAKE_PREFIX_PATH=/cvmfs/sw.hsf.org/key4hep/releases/2024-...
...
```

How do we build?

- Builds are done in containers in Openstack instances provided by CERN
- Gitlab CI pipelines are used
- The configuration file `spack.yaml` for the nightlies (opt) looks like:

```
spack:
  view: false
  include:
  - ../key4hep-common/config-nightlies.yaml    # Paths for the nightlies
  - ./packages.yaml                            # Specific for nightly-opt
  - ../key4hep-common-opt/packages.yaml       # Configuration for opt builds
  - ../key4hep-common/packages.yaml           # Common for all builds
  - ../key4hep-common/compilers.yaml
  - ../key4hep-nightly-share/packages.yaml     # Location of packages that are already installed
  repos:
  - ../..   # Add all the packages in the key4hep-spack repository
  specs:
  - key4hep-stack+devtools
```

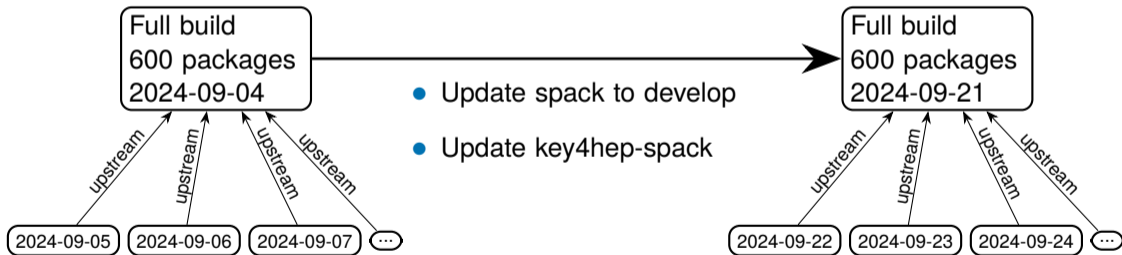

How do we build?

- Multiple parts of the configuration in different packages .yaml don't combine well
- An additional script is used to combine the included packages .yaml into a single one with the combined configuration
- If building the nightlies, it also adds the latest commit of some packages to the requirements
- Final packages .yaml:

```
packages:  
dd4hep:  
  require: '@18c76375c7d3ebc2b35b293e9985e2a1671aedef6=develop +edm4hep+lcio+xercesc+hepmc3 build_type=Release'  
edm4hep:  
  require: '@9d6325262422af907da41a0ede66d00371569bdc=develop build_type=Release cxxstd=20'  
...
```

How do we build?

- For the nightlies, every few weeks the latest commit of spack is picked and a full build is done
- The rest of the days only a subset of repositories are built (the ones that have changed). These use the builds from scratch as upstream



Debug builds

- Debug builds (with debug symbols and without compiler optimizations) share many packages with the opt builds
- They use as upstream the opt builds

Deployment

- After a build is completed and tested it is deployed to CVMFS by copying the build with rsync to the publisher node
- Several files are provided for each build for reproducibility:
 - `.spack-commit`: Which commit of spack was used to build this release
 - `.key4hep-spack-commit`: Which commit of key4hep-spack was used to build this release
 - `.cherry-pick`: A script that cherry picks the same commits as it happened during the build

Building Key4hep: User side

- We provide a script that checks the OS and sources the appropriate environment script
- With a `source /cvmfs/sw.hsf.org/key4hep/setup.sh` users get access to the stack

```
AlmaLinux/RockyLinux/RHEL 9 detected
Setting up the Key4hep software stack release latest-opt from CVMFS
Use the following command to reproduce the current environment:
```

```
source /cvmfs/sw.hsf.org/key4hep/setup.sh -r 2024-10-03
```

```
If you have any issues, comments or requests, open an issue at https://github.com/key4hep/key4hep-spacak/issues
Tip: A new -d flag can be used to access debug builds, otherwise the default is the optimized build
```

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

Building Key4hep: User side

- The setup script has several options
- Passing `--help` or `-h` displays the usage

```
Usage: source /cvmfs/sw.hsf.org/key4hep/setup.sh [-r <release>] [--list-releases [distribution]] [--list-packages [distribution]]
-d          : setup the debug version of the software stack
-r <release> : setup a specific release, if not specified the latest release will be used (also used for --list-packages)
--help, -h  : print this help message
--list-releases [distribution] : list available releases for the specified distribution (almalinux, centos, ubuntu).
--list-packages [distribution] : list available packages and their versions for the specified distribution (almalinux, centos, ubuntu).

In addition, after sourcing, the command k4_local_repo can be used to add the current repository to the environment
It will delete all the existing paths containing the repository name and add some predefined paths to the environment
```

- Example: how to see which version of EDM4hep will be picked up?

```
$ source /cvmfs/sw.hsf.org/key4hep/setup.sh --list-packages | grep edm4hep
edm4hep 0.99.1
```

Developing with Key4hep

- What is the workflow for developers?
- After sourcing, users are mostly on their own
- A function to setup packages locally was developed and is provided in the `setup.sh` script
- `k4_local_repo` will remove the paths from the stack for the current package and add the local ones
- Example workflow:

```
source /cvmfs/sw.hsf.org/key4hep/setup.sh
git clone https://github.com/key4hep/edm4hep
cd edm4hep
k4_local_repo
mkdir build; cd build
cmake ..
cmake --build .
```


Outlook

- Room for improvement
- From Spack:
 - Better CI - improved a lot recently
 - Separating core libraries and package recipes?
 - Concretization (resolving all the package versions, variants and dependencies) is slow
- Cache usage to avoid redundant building
 - Nightlies are built in different directories every day so it's not trivial to set up ccache
 - Using the Spack buildcache was studied and the issues found were reported in a [HSF meeting](#)
 - 8 hours for a full build, 1 hour for nightly for each OS and compiler combination (16 threads)
- Development with Spack?

Summary

- We use successfully Spack in Key4hep to build our software stacks
 - Benefit from maintenance of the recipes from the community
- System to build releases and nightlies automatically
 - Support multiple OSes and build types
 - Doesn't need human supervision except when new versions are built and builds fail
- Overall happy with Spack

Backup

Building Key4hep: Result

```
$ ls /cvmfs/sw-nightlies.hsf.org/Key4hep/releases/2023-06-24/x86_64-almalinux9-gcc11.3.1-opt
acts                pmix
aida                podio
aidatt              popt
alpaka              protobuf
assimp              psimd
...                 ...
```

Inside each package there is a folder containing the version and the hash

```
$ cd acts
$ ls
23.2.1-hicgwc
```

And then the actual package is inside this folder:

```
$ cd 23.2.1-hicgwc
$ ls
bin include lib64
```

Building Key4hep: Build workflow

- When doing a full build and updating the spack commit, building never works
- Fixes are found and either a PR upstream or a workaround is made
- If a PR upstream, then the commit in the PR is cherry picked for the build
- It's not too bad since we update frequently, there can't be that many failures in two weeks worth of spack commits, but there is always at least one

Gitlab Pipeline example

- Example of the pipeline to build and deploy the nightlies

Passed Juan Miguel Carceller created pipeline for commit 3eb668b5 16 hours ago, finished 14 hours ago

For main

Scheduled latest 12 jobs 74 minutes 54 seconds, queued for 2 seconds

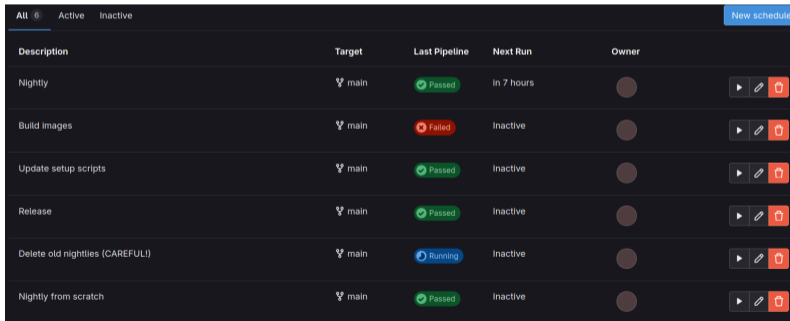
Pipeline Needs Jobs 12 Tests 0

Group jobs by Stage Job dependencies Show dependencies

Job Name	Stage	Dependencies	Status
prepare-spack- prepare-spack- alma9	prepare-spack		Passed
prepare-spack- prepare-spack- alma9-gcc14	prepare-spack		Passed
prepare-spack- prepare-spack- ubuntu22	prepare-spack		Passed
build-nightly-opt- compilation-and-test- alma9	compilation-and-test	prepare-spack-*	Passed
build-nightly-opt- compilation-and-test- alma9-gcc14	compilation-and-test	prepare-spack-*	Passed
build-nightly-opt- compilation-and-test- ubuntu22	compilation-and-test	prepare-spack-*	Passed
build-nightly-dbg- compilation-and-test- alma9	compilation-and-test	prepare-spack-*	Passed
build-nightly-dbg- compilation-and-test- alma9-gcc14	compilation-and-test	prepare-spack-*	Passed
build-nightly-dbg- compilation-and-test- ubuntu22	compilation-and-test	prepare-spack-*	Passed
deploy-nightly- deployment- alma9	deployment	build-nightly-*	Passed
deploy-nightly- deployment- alma9-gcc14	deployment	build-nightly-*	Passed
deploy-nightly- deployment- ubuntu22	deployment	build-nightly-*	Passed

Gitlab Schedules

- Nightly and Release build the nightlies and the stable releases
- Build images: builds the docker images for the containers
- Update setup scripts: updates the `setup.sh` scripts to their latest version in the `key4hep-spac` repository
- Delete old nightlies: deletes the old nightlies from the `cvmfs`
- Nightly from scratch: Builds all the packages for the next nightly



The screenshot shows the GitLab Schedules page with a table of scheduled jobs. The table has columns for Description, Target, Last Pipeline, Next Run, and Owner. There are also tabs for 'All', 'Active', and 'Inactive', and a 'New schedule' button.

Description	Target	Last Pipeline	Next Run	Owner
Nightly	main	Passed	In 7 hours	
Build Images	main	Failed	Inactive	
Update setup scripts	main	Passed	Inactive	
Release	main	Passed	Inactive	
Delete old nightlies (CAREFUL!)	main	Running	Inactive	
Nightly from scratch	main	Passed	Inactive	

Slide for HSF meeting (1): Issues: Build Caches

- We have been testing build caches
- A build was done for one day (for example 2022-05-20) and we tried to install in the location corresponding to a different day (2022-05-30)
- Relocation failed in several occasions:
 - git has binaries in non standard locations, but the RPATHS inside those were still pointing to the build location and not to the one where they were being installed
 - At least one package with symbolic links didn't have the symbolic links updated to the new locations
- At that point the build was broken and no more research was done, but it's possible there are more issues

Slide for HSF meeting (2): Issues: Build Caches, Examples

- Git: installed on 2023-05-30, rpath pointing to where the build cache was originally built

```
$ pwd
/cvmfs/sw-nightlies.hsf.org/Key4hep/releases/2023-05-30/x86_64-almalinux9-gcc11.3.1-opt/git/2.40.0-bwhwxr/libexec/git-core
$ objdump -p git
...
RPATH                /cvmfs/sw-nightlies.hsf.org/Key4hep/releases/2023-05-20/x86_64-almalinux9-gcc11.3.1-opt/git/2.40.0-bwhwxr/lib...
...
```

- Python example, broken symlinks:

```
$ ls -lah /cvmfs/sw-nightlies.hsf.org/Key4hep/releases/2023-05-30/x86_64-almalinux9-gcc11.3.1-opt/python/3.10.10-plst3j/bin
total 102K
drwxr-sr-x. 2 cvmfs cvmfs 235 Jan  1 1970 .
drwxr-sr-x. 7 cvmfs cvmfs  70 Jan  1 1970 ..
lrwxrwxrwx. 1 cvmfs cvmfs   9 May 30 08:27 2to3 -> 2to3-3.10
-rwxr-xr-x. 1 cvmfs cvmfs 201 May 30 08:27 2to3-3.10
lrwxrwxrwx. 1 cvmfs cvmfs   8 May 30 08:27 idle3 -> idle3.10
-rwxr-xr-x. 1 cvmfs cvmfs 199 May 30 08:27 idle3.10
lrwxrwxrwx. 1 cvmfs cvmfs   9 May 30 08:27 pydoc3 -> pydoc3.10
-rwxr-xr-x. 1 cvmfs cvmfs 184 May 30 08:27 pydoc3.10
lrwxrwxrwx. 1 cvmfs cvmfs 121 May 30 08:27 python -> /cvmfs/sw-nightlies.hsf.org/Key4hep/releases/2023-05-20/x86_64-almalinux9-gcc11.3.1-opt/python/3.10.10-plst3j/bin
lrwxrwxrwx. 1 cvmfs cvmfs  10 May 30 08:27 python3 -> python3.10
-rwxr-xr-x. 1 cvmfs cvmfs 27K May 30 08:27 python3.10
-rwxr-xr-x. 1 cvmfs cvmfs 3.1K May 30 08:27 python3.10-config
-rwxr-xr-x. 1 cvmfs cvmfs 65K Jan  1 1970 python3.10-gdb.py
lrwxrwxrwx. 1 cvmfs cvmfs  17 May 30 08:27 python3-config -> python3.10-config
lrwxrwxrwx. 1 cvmfs cvmfs 128 May 30 08:27 python-config -> /cvmfs/sw-nightlies.hsf.org/Key4hep/releases/2023-05-20/x86_64-almalinux9-gcc11.3.1-opt/python/3.10.10-plst3j/bin
```