

Software Ecosystem (and MC generators)

ECFA Higgs Factories: 1st Topical Meeting on Generators

November 10, 2021
G Ganis, CERN-EP

Outline

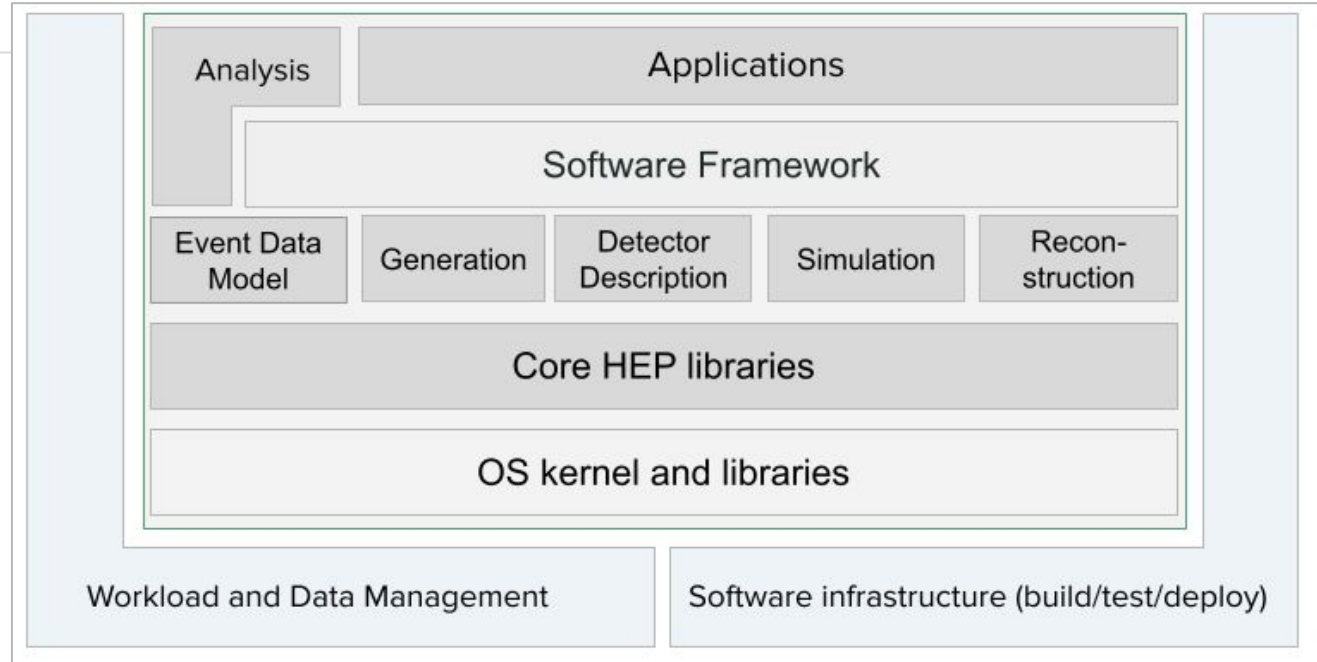
- From software ecosystems to a common software system for future projects
 - Key4hep
- Monte Carlo generators @ key4hep
 - Availability
 - Interoperability levels and managements
 - Role and current status of k4Gen
- Final remarks and Summary

From software ecosystems to key4hep

HEP Software Ecosystem

See [G Ganis @ ECFA Kick-off, June 2021](#)

Seamless integration and optimization between various networks of devices, software and services aimed to facilitate data processing for High Energy Physics experiments



Software Ecosystems for future experiments

- Each experiment needs and has a software ecosystem to satisfy S&C needs
 - Number of common tools increased with time
 - Historically Cernlib, PAW, ROOT, ... now machine learning tools, frameworks, detector description, etc
- Future projects feel ready for a common software ecosystem
 - Similar software needs: support for physics and detector studies
 - Flexible detector description, open to evolution
 - Completeness: Generation, Simulation, Reconstruction, Analysis, MDI support, ...
 - Ease of use: low usability threshold and fast learning curve
 - Extensive documentation, regular training
 - Similar needs for software tools to manage computing
 - Tools to facilitate effective access to CPU / storage resources

The common software vision (key4hep)

Create a software ecosystem integrating in optimal way various software components to provide a ready-to-use full-fledged solution for data processing of HEP experiments

Complete set of tools for

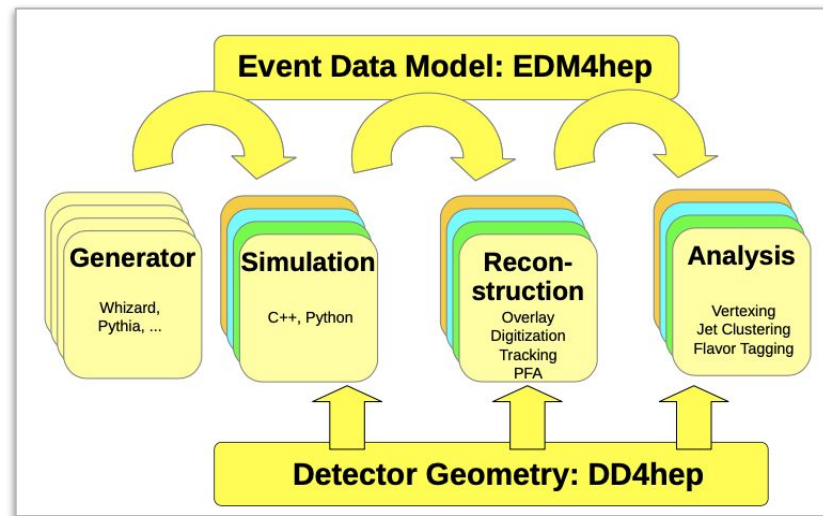
- Generation, simulation, reconstruction, analysis
- Build, package, test, deploy, run

Core Ingredients of current key4hep

- PoDIO for **EDM4hep**, based on LCIO and FCC-edm
- **Gaudi** framework, devel/used for (HL-)LHC
- **DD4hep** for geometry, adopted at LHC
- **Spack** package manager, lot of interest from LHC

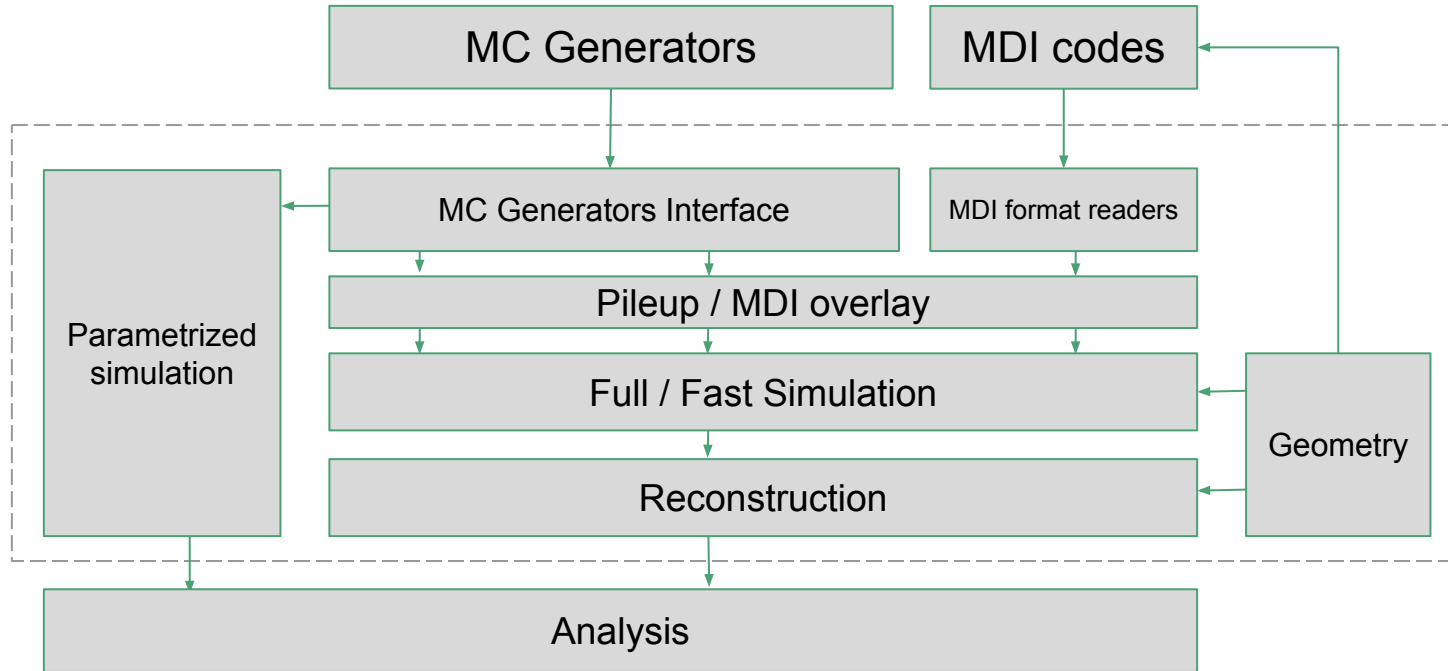
Community project, unifying efforts

- Contributions from CLIC, ILC, FCC, CEPC
- Support from major R&D programs



Kick-off meetings in [Bologna](#), [Hong Kong](#)

Typical workflows to support



Build on the experience of running experiments

- Requirements of running experiments, in particular LHC, in terms of software and computing, are unprecedented. Working solutions exist for
 - Frameworks
 - Reconstruction techniques / algorithms
 - ML techniques development / deployment
 - MT, heterogenous (GPU, FPGA, ...)
 - Workload and Data Management
 - Software build / packaging / test / deployment
 - Analysis tools
 - ...

New developments should focus on what is uncovered / specific to EW/Higgs factories e.g. e+e- MC, flexible geometry, specific detector technologies, reconstruction/analysis tools, ... possibly generic and re-usable.

Role of R&D programs: AIDAs, CERN EP R&D, ...

- **AIDA: Joint European effort for detector R&D**

- Successful software packages (WP2@AIDA, WP3@AIDA2020)
 - Core software, Simulation
 - VecGeom (vectorized geometry), **DD4hep** (geometry description, ...)
 - **PoDIO: EDM toolkit**
 - Advanced reconstruction
 - Tracking (converged to ACTS), Particle Flow (PandoraPFA)
- Approved follow-up: AIDA-Innova 2021-2025
 - **Turnkey Software Stack**, Fast Simulation, Track Reconstruction (ACTS), Particle Flow (PFA)
 - Focus on: parallelisation, acceleration, machine learning



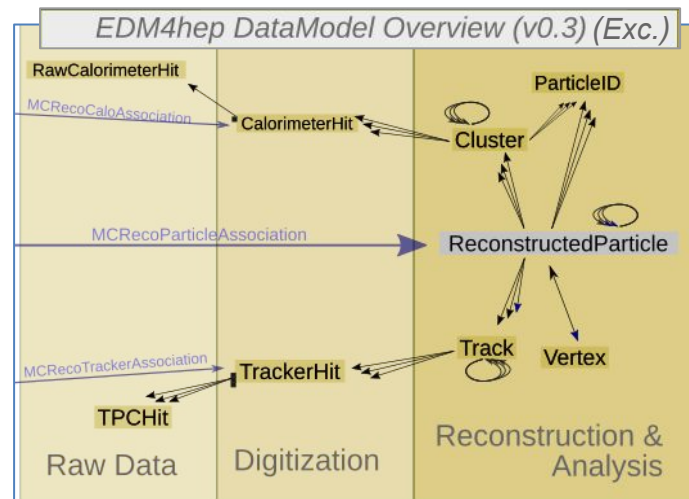
- **CERN EP R&D software work package #7**

- **Turnkey Software Stack**, Faster Simulation, reconstruction at High Pile-Up, Efficient Analysis

The common event data model: the challenges

EDM provides common language for exchange among framework components

- **Challenge 1: efficient support different collision environments (e+e-, pp, ...)**
 - Positive first experiences with FCC-hh components
- **Challenge 2: keep I/O efficient**
 - PoDIO: separate definition from implementation, facilitate optimal adaptation to backend
 - POD layer designed for efficient I/O, simple memory layout
 - Flat data support (RNTuple) will provide insight
- **Challenge 3: efficient support for schema evolution**
 - Requires schema evolution in PoDIO, planned
- **Challenge 4: efficient support for detector needs**
 - Interaction w/ detector teams from the start
 - Eg. cluster counting for IDEA Drift Chamber



The key4hep project today

See [P Declara at ILCX workshop](#)

- **Key4hep actively developed**

- Regular (weekly) meetings
- O(10) people attending, mostly from CERN, DESY, CEPC
- *Deliver early, deliver often* approach to foster adoption

- **Key4hep adoption status**

- FCC: physics performance studies fully based on key4hep
 - FCCSW re-arranged into key4hep components and adapted to EDM4hep
 - Planning to use iLCSoft reconstruction algorithms through Marlin wrapper
- CEPC: well advanced
 - Gaudi based, several components from key4hep, developed components to facilitate transition
- LC community: keep existing software chains
 - Planned full test of key4hep-based workflow in AIDAInnova
- SCT: started adapting to EDM4hep

Key4hep common tooling: not only HEP specific

- Key4hep provides also general software tools
 - For configuring and building: CMake, autoconf, ...
 - Compilers: Gcc, Clang, gfortran, ...
 - Sanitizers, code style checking: clang-format, clang-tidy
- Common testing tools: catch2, pytest, unittest, ...
- Templates for new packages
- Continuous Integration infrastructure
 - Help identify regressions and consistency with dependencies
- Package manager for unified building and deployment
 - **Spack**, lot of interest in HEP and LHC
 - Several packages already available

Where can key4hep be found?

- Key4hep is available on the **CernVM-FS** shared distributed file system

```
$ source /cvmfs/sw.hsf.org/key4hep/setup.sh
Setting up the latest Key4HEP software stack from CVMFS ...
$ which k4run
/cvmfs/sw.hsf.org/spackages3/k4fwcore/1.0pre13/x86_64-centos7-gcc8.3.0-opt/k7.../bin/k4run
$ which mg5
/cvmfs/sw.hsf.org/spackages3/madgraph5amc/2.8.1/x86_64-centos7-gcc8.3.0-opt/5o.../bin/mg5
$ pythia8-config --libdir
/cvmfs/sw.hsf.org/spackages3/pythia8/8.306/x86_64-centos7-gcc8.3.0-opt/2c.../lib
```

- CernVM-FS optimizes remote access
 - LHC driven solution adopted widely
 - Available for any unix-like system
 - Requires internet connection (but solutions to work off-line available)

Monte Carlo generators @ key4hep

MC Generators

See also [W. Kilian's, ECFA kick-off, June 2021](#)

Needs

- High energy ($\sqrt{s} > hZ$ threshold) e+e- generators
- Generators at Z peak, WW
- Heavy Flavour decays, including taus

Examples of heavily used codes for LC

- Whizard, MadGraph5_aMC, PhysSim, Pythia6, ...

Areas of work (not exhaustive)

- Recovery of LEP generators, but still state of art for Z peak, WW
 - KKMC family, BHLUMI, BHWIDE, Babayaga, ... interfacing work ~~in progress~~ **mostly done**
- Hadronization “tunes” for e+e- (Pythia, Herwig, ...) **solutions possible**
 - Eg. cannot import Pythia6 tune (from LEP) to Pythia8
- Interfaces with up-to-date decay codes (EvtGen, ...) **work needed**

Monte Carlo Generators in key4hep

- A Monte Carlo generator is a **package**
- Key4hep includes already many generators as packages
 - Initial list derived from **LCG stacks**, so sort of LHC oriented
 - But several e+e- additions available: Whizard, KKMCEE, BabaYaga, BHLUMI, ...
 - Including wrappers for better user experience
- What does it mean “adding a generator to key4hep”?
 - Required information for inclusion in the package manager
 - **Source** location, minimal **documentation on how to build and** required **dependencies**, default configuration files, tests, ...
 - Key4hep infrastructure will
 - Build in **shared installation** mode
 - Run built-in tests, if any
 - Install in **distributed shared file system**

List of MC Generators currently available in key4hep

From spack upstream

form 4.2.1	vbfno 2.71	collier 1.2.5
crmc 1.6.0	syscalc 1.1.7	gosam 2.0
photos 3.64	thepeg 2.2.1	herwig3 7.2.1
qd 2.3.13	chaplin 1.2	madgraph5amc 2.8.1
evtgen 2.1.0	heputils 1.3.2	openloops 2.1.2
lhpdf 6.3.0	looptools 2.15	pythia8 8.306
mcutils 1.3.5	njet 2.1.1	recola 2.2.3
qgraf 3.4.2	pythia6 6.4.28	yoda 1.9.0
rivet 3.1.4	sherpa 2.2.11	
hepmc 2.06.11	hepmc3 3.2.4	

From key4hep-spack

guinea-pig 1.2.2rc
whizard 3.0.1
KKMCee 4.32.01
BHLUMI 4.04-linuxLHE
Babayaga fcc-1.0.0

Levels of interoperability

- Level 0 - *Common Data Formats*
 - Maximal interoperability, even on different hardware
- Level 1 - *Callable Interfaces*
 - Defined for one or more programming languages
 - Implementation quality of interfaced components important
 - Required to define plugins
- Level 2 - *Introspection Capabilities*
 - Software elements to facilitate the interaction of objects in a generic manner such as Dictionaries and Scripting interfaces
 - Language bindings, e.g. PyROOT
- Level 3 - *Component Level*
 - Software components are part of a *common framework*, optimal interplay
 - Common configuration, log and error reporting, plug-in management, ...

Common Data Formats

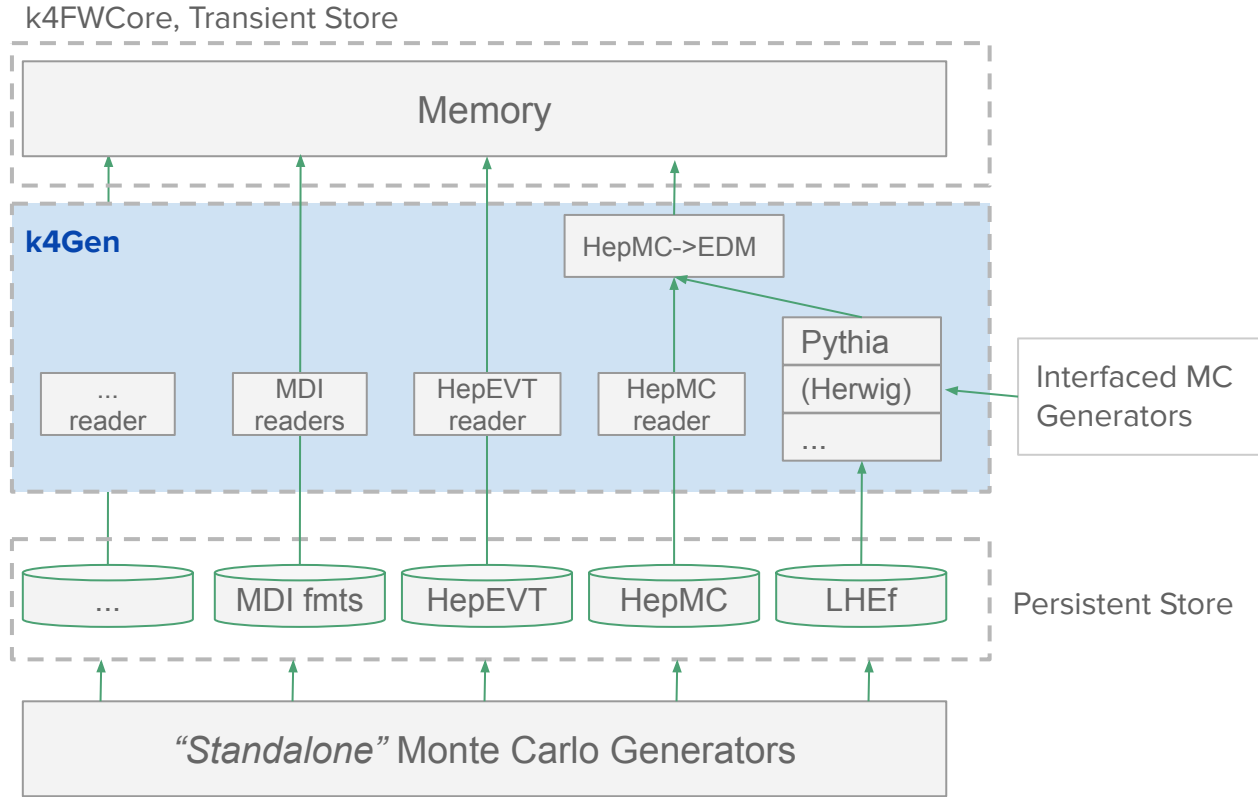
- LHEf
 - Format agreed for LHC, generally OK for e+e-
 - Some issues for FCC (see FCC presentation)
- HepEVT
 - Format used by old fortran generators
- HEPMC
 - Generic framework for MC generator event record encoding and manipulation
 - Version 3 complete re-write of previous version and generally adopted
 - Provide tools for managing LHEf files
- Potentially any format completely documented

Managing interoperability through Gaudi

- Components tailored for specific tasks
 - Tools for inputting and managing MC outputs
- What is required
 - Readers for data formats
 - Currently available: HepEVT, HepMC, LHEf, ...
 - Level-1 interfaces for MC
 - Only currently available Pythia8
 - Place where to put an interface to Herwig

The repository is [k4Gen](#), and includes also modules to perform actions on the MC events, such as vertex smearing, particle filters, or basic MC tools, such particle guns

Managing interoperability



Final remarks

Possible future evolution

- As Repository

- Generator content evolution
 - Currently mostly driven by FCC-ee@Z: what about other {Ecms, env contexts}?
 - What about packages used inside other packages (Dizet, Tauola, ...)?
 - Should an effort be done to have a single version in the stack used by wherever needed?
- What role authors see for this repository?
 - Can it somehow be the technical contact point among authors and users, at least for e+e-?
 - Requires some level of engagement and setup of an efficient communication channel

- As Framework interface

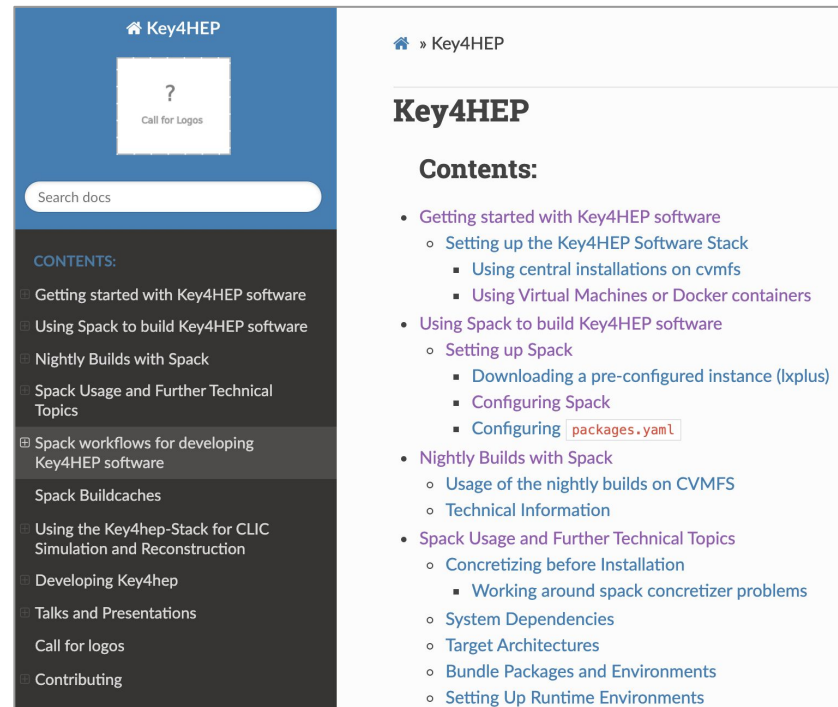
- Missing components
 - Herwig interface: enable switching hadronization/fragmentation technology. Others?
- Data formats and related libraries
 - HepMC3 can handle all relevant data formats (HepMC, LHEf, HepEvt, ascii/binary): should it get an even more central role?

Summary

- Key4hep is taking shape and increases is offering following the needs of its immediate stakeholders: **EW/Higgs factories**
- Key4hep provides
 - A coherent software stack
 - Increasingly complete and centrally distributed
 - A sanitized environment, w/ internal consistency tests
- For MC Generators, a 2-level interoperability interface is provided
 - Matching current needs of projects
 - But ready to be adapt to evolving needs, exploiting the flexibility of the underlying framework

Useful links

- Key4hep GitHub Project
<https://github.com/key4hep>
- Main documentation page
<https://key4hep.github.io/key4hep-doc/>
- Doxygen available., e.g. for EDM4hep
<https://edm4hep.web.cern.ch/>



The screenshot displays the Key4HEP documentation website. The top navigation bar is blue with the text 'Key4HEP' and a home icon. Below the navigation bar is a white box with a question mark and the text 'Call for Logos'. A search bar labeled 'Search docs' is positioned below the navigation bar. The main content area is dark grey and features a 'CONTENTS:' section with a list of links: 'Getting started with Key4HEP software', 'Using Spack to build Key4HEP software', 'Nightly Builds with Spack', 'Spack Usage and Further Technical Topics', 'Spack workflows for developing Key4HEP software', 'Spack Buildcaches', 'Using the Key4hep-Stack for CLIC Simulation and Reconstruction', 'Developing Key4hep', 'Talks and Presentations', 'Call for logos', and 'Contributing'. The right sidebar is white and contains the text 'Key4HEP' with a home icon, followed by a 'Contents:' section with a list of links: 'Getting started with Key4HEP software' (with sub-links 'Setting up the Key4HEP Software Stack' and 'Using Virtual Machines or Docker containers'), 'Using Spack to build Key4HEP software' (with sub-links 'Setting up Spack', 'Downloading a pre-configured instance (Ixplus)', and 'Configuring Spack'), 'Nightly Builds with Spack' (with sub-links 'Usage of the nightly builds on CVMFS' and 'Technical Information'), and 'Spack Usage and Further Technical Topics' (with sub-links 'Concretizing before Installation', 'Working around spack concretizer problems', 'System Dependencies', 'Target Architectures', 'Bundle Packages and Environments', and 'Setting Up Runtime Environments').

Additional material

Adding a new package (MC) to key4hep

- Basic requirements for inclusion
 - Source publicly available (ideally GitHub/GitLab), proper versioning
 - Support for central installation
 - Minimal test suite
 - Minimal documentation
- The key4hep development team can provide some support and guidance
 - Including access to infrastructure to run Continuous Integration tests