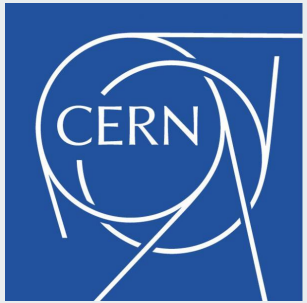
Building Key4HEP with Spack

Summer project with Hobbs Willet(t)
hobbs.arthur.willett@cern.ch

Supervisors: Graeme Stewart, Javier Cervantes Villanueva



Key4HEP

Provide one large, ready to use stack of software to future experiments, as a modernisation of the LCG releases.

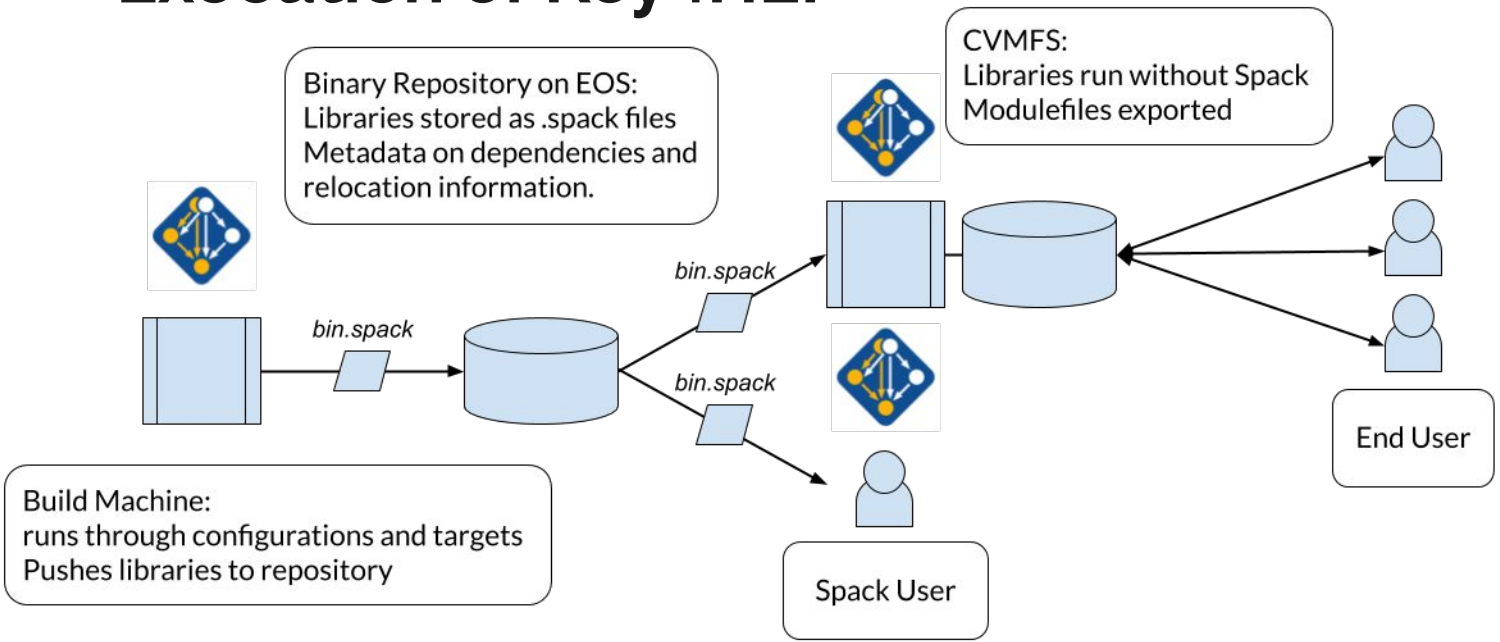
Easier to maintain, improves reproducibility and allows the whole community to share the progress made to its construction.

Minimal setup by the end user and maximum amount of software ready to run ‘out of the box’.

This system uses current infrastructure.

Doesn't require end users to know about Spack.

Execution of Key4HEP

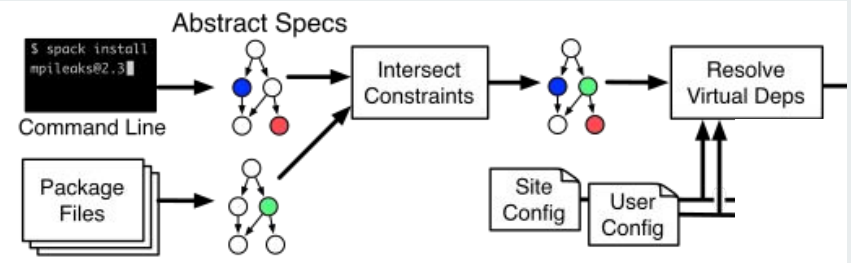


My Project

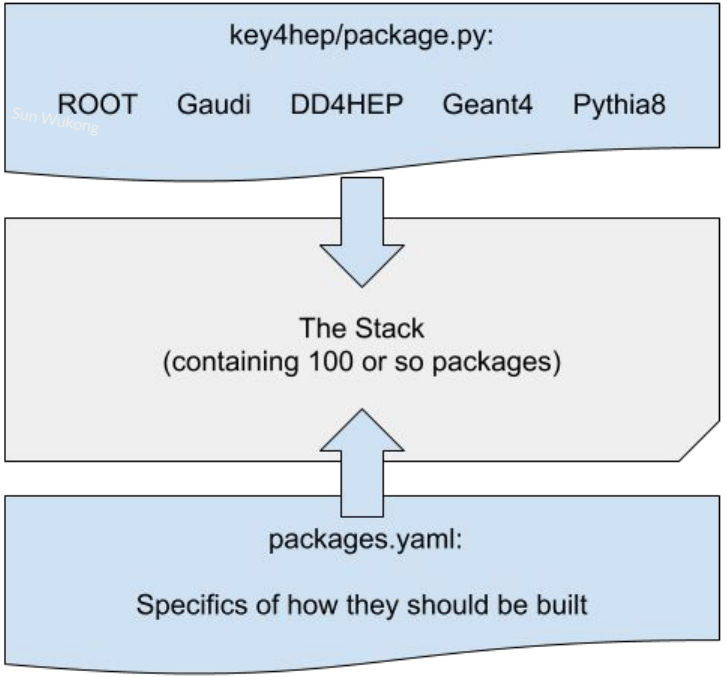
- Reproduce a prototype LCG stack with Spack
- Investigate the stack at runtime
- Distribute the stack

The Stack Prototype

Constructed from above and below.



$ spack install key4hep



key4hep installs nothing but lists dependencies

Combined they replicate the behaviour of an LCG release

Internal or External compilers

Internal

- Configuration file specifies a compiler.
- If compiler not available in the system spack will install it (from scratch or from an available binary repository).

-> Went with this one in the stack, as it is ‘closed’ system and does not have LCG as a dependency.

External

- An LCG compiler used from CVMFS.
- Requires manual configuration to find all necessary libraries.

Runtime environments

Filesystem Views:

A directory structure with (sym/hard)links to the package and its dependencies.

Manually prepend to your environment variables.

Common in HEP

Modulefiles:

Adaptable and scriptable system for dynamically loading and unloading environment variables.

Spack can generate these automatically.

Common in HPC

Filesystem View Issues

1. Merge conflicts:

Spack attempts to merge all the expectations of where packages look for dependents. For a view of ROOT, it fails on `libffi/share/info/dir`

2. Manual setup

All environment variables (e.g PATH, ROOTSYS, PYTHONPATH) will have to be set manually (or by a script)

Modulefile Issues

Leads to long environment variables ~ e.g. 61 directories for Key4HEP. These load all necessary environment variables using the recipe

Requires users to use modulefile infrastructure e.g Lmod, environment-modules

Use Cases

Experiment runtime:

A small number of software applications will be run by each experiment (simulation,reconstruction, analysis) using the packages.

A short PATH needed for large scale efficiency.

Unclear how to consistently set up all variables

Package development:

Developers need very specific control over loaded environment.

→ Modulefiles

These are complementary approaches, we are evaluating which one is the best technical solution.

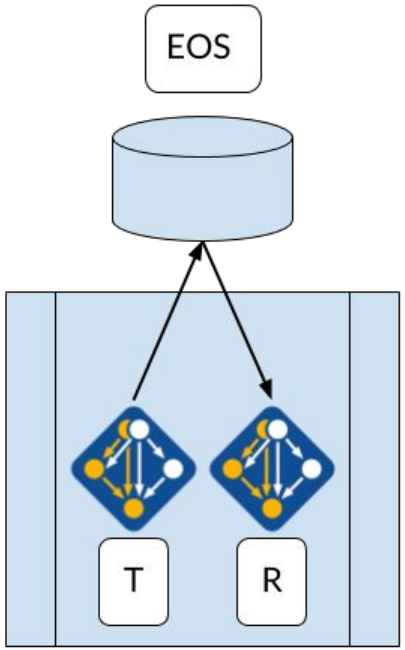
Distribution Test Suite

To examine Spacks behaviour when making, pushing to and pulling from binary repositories.

To do this we have two Spack instances on one machine that do not know of each other:

1. Transmitter
2. Receiver

And a binary repository that they share.



Spack-environments

Spack-environments are relatively new, and work like conda environments. The allow a subset of the packages to be seen.

In addition a spack-env can be replicated exactly (or approximately) in Spack using a environment file in a non-human readable format. For transfer or preservation.

The spack-env file also replicates the behaviour of a packages.yaml, it will search the buildcache by the correctly concretised packages.

Entire stack can be replicated between machines using single small file.

Work to do

- Put the stack on CVMFS for experiments to use
- Construct complementary version of the stack between FS views and Modulefiles