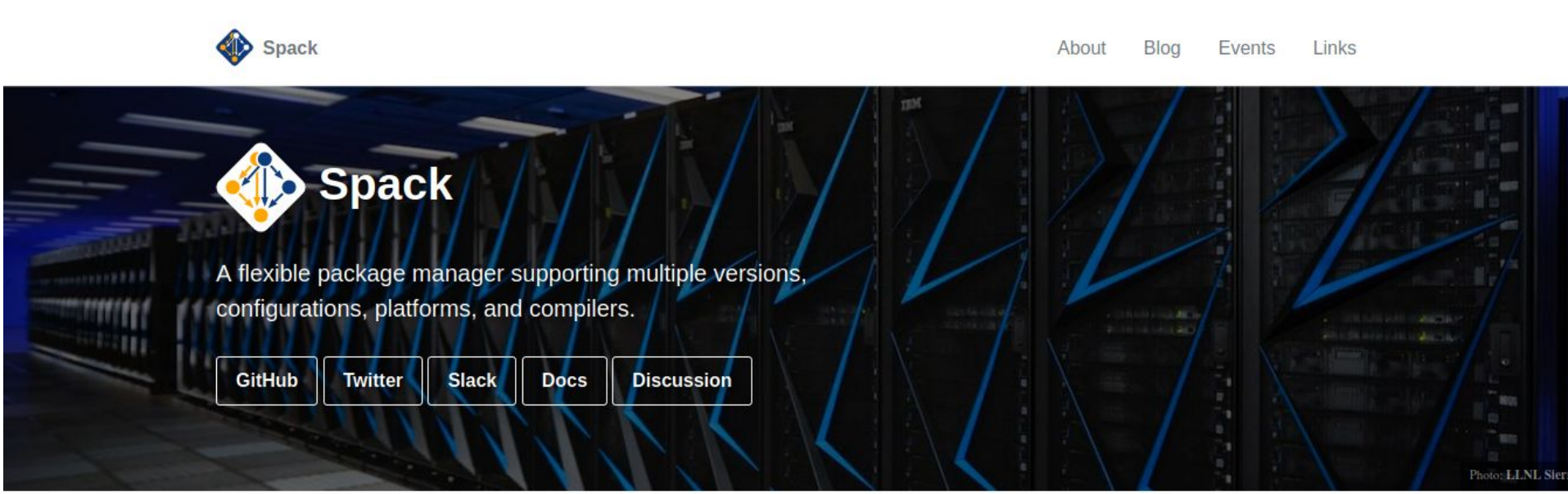
The spack installation on /cvmfs/sw.hsf.org/ - HOWTO for Librarians and Lessons from Key4hep

LiM- 2021/03/15

Valentin Volkl (CERN), Key4hep Software Group

Spack Jargon

- Spec
- Concretized Spec
- Environment
- Recipe/repository
- Upstream
- Archspec
- variant



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

Using the installation - without spack

- The switch to spack is (hopefully) transparent.
  - After setup, you should be able to use the installation like other LCG releases

- No setup script available at the moment
  - Need Spack to setup the installation
- Key4hep has a solution for creating setup scripts!

```
<Setup spack (see later slides)>
spack load --first lcg-release@98

# Use as you would lcg release
cmake ..
root ..
```

Spack Upstreams

- Adding packages from another spack installation is in principle extremely easy:

```
spack/etc/spack/upstreams.yaml

upstreams:
  spack-instance-1:
    install_tree: /cvmfs/sw.hsf.org/sft-spack/
  spack-instance-2:
    install_tree: /path/to/another/spack/opt/spack
```

1. Note: once you use a package as a dependency, its paths are basically set in stone. Putting it in another location essentially means re-installing all dependents

Spack for Key4HEP

- Complete setup instructions:

```
git clone https://github.com/spack/spack.git
source spack/share/spack/setup-env.sh

# register sft packages
git clone https://gitlab.cern.ch/sft/sft-spack-repo.git
spack repo add sft-spack-repo --scope-site

# setup spack environment
cd sft-spack-repo/config/LCG_98python3_ATLAS_5
spack env activate .

# install your own packages
spack add podio
spack concretize
spack install
```

What's on CVMFS?

- /cvmfs/sw.hsf.org/sft-spack/linux-centos7-haswell/gcc-8.3.0/

  Default install prefix, But basically arbitrary! All important data is inside the package install dir, p.ex. .../root-6.22.00-okmrmwxhkn72rws5rcj456wnkzmdocq3/.spack

- /cvmfs/sw.hsf.org/sft-spack/.spack
  - Is a cache of the package database, can be regenerated
- /cvmfs/sw.hsf.org/sft-spack/bin/sbang
  - Technicality to deal with long packagenames
- /cvmfs/sw.hsf.org/sft-spack/modules
- /cvmfs/sw.hsf.org/sft-spack/views

What's on CVMFS?

- /cvmfs/sw.hsf.org/spackages
  - Default installation prefix used for key4hep packages
- /cvmfs/sw.hsf.org/share/data
  - Installation prefix for architecture-independent packages (geant4-data)
- /cvmfs/sw.hsf.org/contrib/spack
  - Installation of the key4hep fork of spack
- /cvmfs/sw.hsf.org/key4hep
  - Setup scripts for key4hep

Personal Opinion 1: Views

- With Spack, Views don't offer much benefit
- What is really needed is a simple, quick, bash-only way to setup the installations
- Key4hep has this:

```
source /cvmfs/sw.hsf.org/key4hep/setup.sh

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

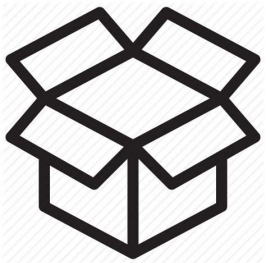
- Explicitly adds all package dirs to all relevant environment variables
  - Views only serve to have a more compact, human-readable environment
  - … but spack load can be used for these purposes

Personal Opinion 2: Distribution of Spack itself

- If a common installation is used, Users should use the same spack distribution that is used to install it (or something very close to it)
- In practice means working on a common fork of spack and maybe installing spack itself on cvmfs
  - .. current practice for key4hep, works well!

- Very difficult to get spack to pick up installed packages otherwise!

Personal Opinon3: 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 package are, however, just not relocatable
- Even if they are, the relocation if 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 Opinion 4: Concretization Time for ROOT

- ROOT should be treated a bit like a compiler
  - Even if just for concretization times
- Concretization times:
  - `spack spec podio: 27 seconds`
  - `Spack spec podio ^ /sldf3234: 2 seconds`
- Mark ROOT as an external install already in the LCG install workflow?
  - If only done by users, will re-install ROOT

Final Comment: Validation

When switching an installation, validation is required - not only for the build, but also for runtime errors!

- Key4hep did a lot of the heavy lifting here already
- Runtime errors encountered:
  - Unit definitions in HepMC
  - XML parser errors
  - Pythia Environment variable runtime error

Backup

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")
```

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
clicperformance     g4saiddata          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
ddmarlinpandora	gear	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
```

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~fo
rtran+gdml+gminimal~graphviz+gsl~http~ipo~jemalloc+math~mems
tat+minuit+mlp~mysql+opengl~postgres~pythia6+pythia8+python~
qt4+r+root7+rootfit+rpath~shadow+sqlite+ssl~table+tbb+thread
s+tmva+unuran+vc+vdt+vmc+x+xml+xrootd
build_type=RelWithDebInfo cxxstd=17
patches=22af3471f3fd87c0fe8917bf9c811c6d806de6c8b9867d30a1e3
d383a1b929d7 arch=linux-ubuntu20.04-broadwell

    ^cmake@3.18.4%gcc@9.3.0~doc+ncurses+openssl+ownlibs~qt
patches=bf695e3febb222da2ed94b3beea600650e4318975da90e4a71d6
f31a6d5d8c3d 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 distinguised

```
$ 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

CVMFS directory tree

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

```
/cvmfs/sw.hsf.org/sft-spack/
|-- sft-spack/ $platform / $compiler / $pkgname-$spackhash / (bin  … )
|-- sft-spack / views   / $K4_version  / $platform / (bin include share … init.sh)
|-- setup.sh
|-- contrib
|-- share/data

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