Managing HPC Application Software with SPACK@LRZ

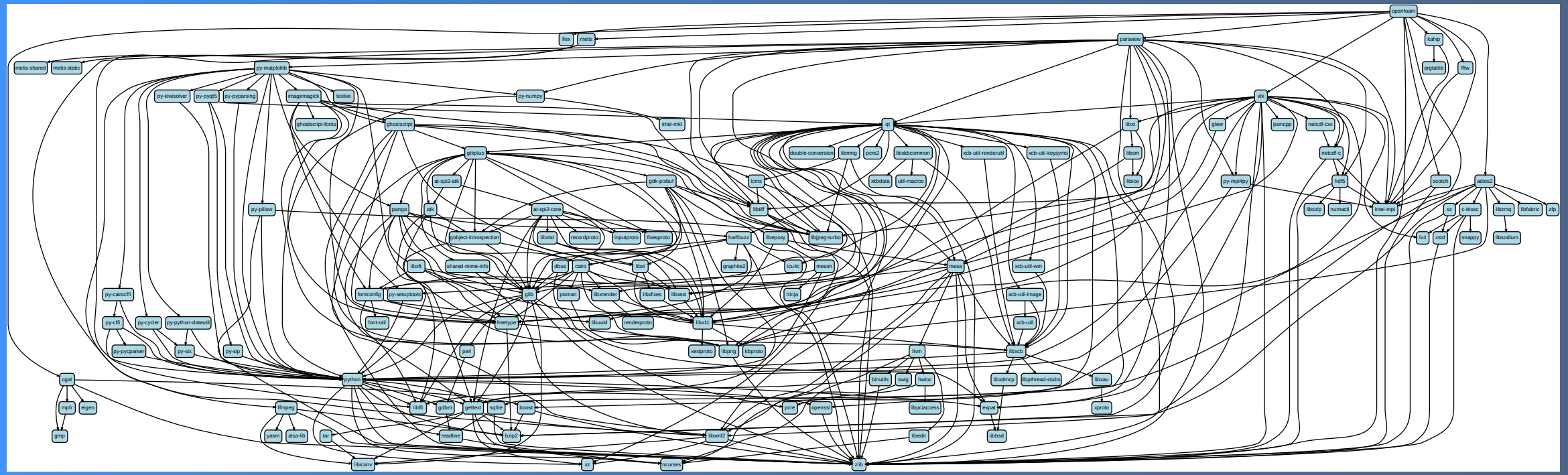
Leibniz-Rechenzentrum | 09.06.2021 | Gilbert Brietzke

Motivation: How to manage the dependency-hell?

- A high-level application may just be the „tip of an iceberg“ when considering a feature-rich configuration of the software with all it‘s dependencies
- Example: OpenSource CFD-Package OpenFOAM

e.g.: feature-rich OpenFOAM incl. vtk & paraview

140 dependencies



Spack is a flexible package manager targeted at HPC-systems

- Spack available at github ‚ready to use'
  few prerequisits only:
  - a basic python,
  - make and a c/c++-compiler
  - tar/gzip/bzip2/xz
  - patch + git + curl
  - pgp (for gnupg2 commands only)

In principle it may be as simple as:

```
git clone https://github.com/spack/spack.
. spack/share/spack/setup.env.sh
spack install <package-spec>
```

NOTE: doing like this is fine but you are on your own … needs some experience with configuring and compiling software in many circumstances

However - we do support spack in user-space via a module called **user_spack**
(with pre-installed and -configured spack)

```
spack install <package-spec>
# e.g.:
spack install hdf5
spack install hdf5%gcc@9.3.0+fortran+hl
spack install hdf5 ^openmpi
```

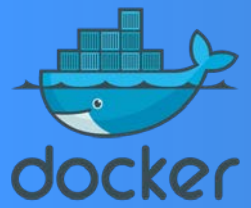
- Spack may install many different variants of the same package:
  - Built different package-versions
  - Built with different compilers
  - Built with different MPI-implementations
  - Built with different build-options

- Installation locations are seperated via unique hashes
  -> installations may peacefully coexist

Spack is one of the many package-managers

- Functional Cross-Platform Package Managers:
  e.g Nix (NixOs), Gnu Guix (Gnu Guix Linux) … use hashes in install-dirs
- Build-from-source Package Managers
  e.g. HomeBrew/LinuxBrew
- Package Managers for specific scripting languages
  e.g. Pip (Python), NPM (Javascript)
- Easy Build:
  installation framework for managing scientific software on HPC-systems: less flexible for experimental build-combinations
- Conda:
  popluar binary package managers for Python and R (but also for other rpm–like packaging in user-space). Easy to use.
  In general no architecture optimized binaries, not targeted at HPC

Containers



- Containers provide a great way to reproduce and distribute an already-built software stack
- In cloud Machines this may be a good choice
- But: who builds the container for your project on a specific HPC-machine?
  - What about MPI?
  - What about well performing parallel-IO
- ! For sofisticated scientific projects on specific HPC-machines this often isn‘t a trivial task!
- `spack containerize` -command may help you with building containers using spack
- This short-talk is not on container. But just to mention:
  We do support Containers at LRZ (at present mostly by Charliecloud-containers)

From manual single package installations to automated stack builds

In the past at LRZ …

- Software stack on LRZ HPC-systems used to be provided via the module system in a non-orchestrated way with hand-written TCL-files to make installations available: applications/libraries/tools /compilers

Limitations:

- Non-transparent or oblique conflicts and/or dependencies of packages
- Non-transparent package-configs and build-variants
- Builds often not reproducible (documentation issue)

Since recently at LRZ …

- Spack compiled software provided for many open-source packages

Advantages:

- **Spack Builds** are **self-documenting:**

  -> Package-builds are typically **reproducible**

- Spack-compiler wrappers inject compiler-flags for the target-architecture -> **optimized** software stack
- Installation of many package-variants do not disturb each other -> many packages may **peacefully coexist**
- Installation (fetch/configure/build/install/module-create) of the software is **automized**

Spack in user-space:
chaining existing installations into your own Spack environment

1. We do privide compiled software with support via environment–modules (the classical way ~>300 modules)

2. NEW + Experimental (work in progress):
module load user_spack
We provide compiled software via spack-chaining

- For experienced users:
  - may use spack via `module load user_spack` that provides a preconfigured spack
  - making use of already installed packages via spack chaining of upstream-location (lrzs/sys/spack/x/y)

-> avoids recompiling low level packages in many situations
-> has working defaults configurated for some essential dependencies (e.g. MPI)

- **Simple Example 1 – install (missing) package libvdwxc:**

```
cm2login3~>module list
Currently Loaded Modulefiles:
 1) admin/1.0    2) tempdir/1.0    3) lrz/1.0    4) spack/21.1.1
cm2login3~>module load user_spack
executing /lrz/sys/spack/user/release/21.1.1/bin/../spack/share/spack/setup-env.sh
cm2login3~>spack spec -lI libvdwxc
Input spec
--------------------------------
 -   libvdwxc

Concretized
--------------------------------
 -   duakorn  libvdwxc@0.4.0%gcc@8.4.0+mpi~pfft arch=linux-sles15-haswell
[^]  ve6ybks      ^fftw@3.3.8%gcc@8.4.0+mpi~openmp~pfft_patches precision=double,f
loat arch=linux-sles15-haswell
[^]  cyojcvv          ^intel-mpi@2019.8.254%gcc@8.4.0 arch=linux-sles15-haswell

cm2login3~>spack install libvdwxc
[+] /dss/dsshome1/lrz/sys/spack/release/21.1.1/opt/haswell/intel-mpi/2019.8.254-gc
c-cyojcvv
[+] /dss/dsshome1/lrz/sys/spack/release/21.1.1/opt/haswell/intel-mpi/2019.8.254-gc
c-cyojcvv
[+] /dss/dsshome1/lrz/sys/spack/release/21.1.1/opt/haswell/fftw/3.3.8-gcc-ve6ybks
[+] /dss/dsshome1/lrz/sys/spack/release/21.1.1/opt/haswell/fftw/3.3.8-gcc-ve6ybks
==> Installing libvdwxc
==> No binary for libvdwxc found: installing from source
==> libvdwxc: Executing phase: 'autoreconf'
==> libvdwxc: Executing phase: 'configure'
==> libvdwxc: Executing phase: 'build'
==> libvdwxc: Executing phase: 'install'
[+] /dss/dsshome1/          /spack/opt/linux-sles15-haswell/libvdwxc/0.4.0-gcc-8.4
.0-duakorn
```

Spack in user-space: chaining existing installations into your own Spack environment

NEW + Experimental (work in progress):
module load user_spack

Example 2: create your own new package inside your own repository.

e.g. libgeotiff
Recently moved to github, version that comes built-in-spack is too old for your purpose

```
cm2login3~>spack create -n libgeotiff -N mine-15.4 https://github.
com/OSGeo/libgeotiff/releases/download/1.6.0/libgeotiff-1.6.0.tar.
gz
==> Using specified package name: 'libgeotiff'

==> This package looks like it uses the cmake build system
==> Created template for libgeotiff package
==> Created package file: /dss/dsshome1/(          /spack/repos/min
e-15.4/packages/libgeotiff/package.py
```

```
# See the Spack documentation for more information on packaging.
# ----------------------------------------------------------------------------

from spack import *


class Libgeotiff(CMakePackage):
    """FIXME: Put a proper description of your package here."""

    # FIXME: Add a proper url for your package's homepage here.
    homepage = "https://www.example.com"
    url      = "https://github.com/OSGeo/libgeotiff/releases/download/1.6.0/libgeotiff-1.6.0.tar.gz"

    # FIXME: Add a list of GitHub accounts to
    # notify when the package is updated.
    # maintainers = ['github_user1', 'github_user2']

    version('1.6.0', sha256='9311017e5284cffb86f2c7b7a9df1fb5ebcdc61c30468fb2e6bca36e4272ebca')
    version('1.5.1', sha256='f9e99733c170d11052f562bcd2c7cb4de53ed405f7acdde4f16195cd3ead612c')
    version('1.4.3', sha256='b8510d9b968b5ee899282cdd5bef13fd02d5a4c19f664553f81e31127bc47265')

    # FIXME: Add dependencies if required.
    # depends_on('foo')

    def cmake_args(self):
        # FIXME: Add arguments other than
        # FIXME: CMAKE_INSTALL_PREFIX and CMAKE_BUILD_TYPE
        # FIXME: If not needed delete this function
        args = []
        return args


-UU-:----F1  package.py     All L1     (Python) ----------------------------------------------------
```

Add the missing stuff: here at least the dependencies need to be specified

Spack in user-space: chaining existing installations into your own Spack environment

NEW + Experimental (work in progress):
module load user_spack

Example 2: create your own new package inside your own repository.

E.g. libgeotiff
Recently moved to github, version that comes built in spack is too old for your purpose

```
cm2login3~>spack create -n libgeotiff -N mine-15.4 https://github.
com/OSGeo/libgeotiff/releases/download/1.6.0/libgeotiff-1.6.0.tar.
gz
==> Using specified package name: 'libgeotiff'

==> This package looks like it uses the cmake build system
==> Created template for libgeotiff package
==> Created package file: /dss/dsshome1/(          /spack/repos/min
e-15.4/packages/libgeotiff/package.py
```

```
# See the Spack documentation for more information on packaging.
# ----------------------------------------------------------------------------

from spack import *


class Libgeotiff(CMakePackage):
    """FIXME: Put a proper description of your package here."""

    # FIXME: Add a proper url for your package's homepage here.
    homepage = "https://www.example.com"
    url      = "https://github.com/OSGeo/libgeotiff/releases/download/1.6.0/libgeotiff-1.6.0.tar.gz"

    # FIXME: Add a list of GitHub accounts to
    # notify when the package is updated.
    # maintainers = ['github_user1', 'github_user2']

    version('1.6.0', sha256='9311017e5284cffb86f2c7b7a9df1fb5ebcdc61c30468fb2e6bca36e4272ebca')
    version('1.5.1', sha256='f9e99733c170d11052f562bcd2c7cb4de53ed405f7acdde4f16195cd3ead612c')
    version('1.4.3', sha256='b8510d9b968b5ee899282cdd5bef13fd02d5a4c19f664553f81e31127bc47265')

    depends_on('jpeg')
    depends_on('libtiff')
    depends_on('proj')
    depends_on('zlib')

    def cmake_args(self):
        # FIXME: Add arguments other than
        # FIXME: CMAKE_INSTALL_PREFIX and CMAKE_BUILD_TYPE
        # FIXME: If not needed delete this function
        args = []
        return args
-UU-:----F1  package.py     All L1     (Python) ----------------------------------------------------
```

Add the missing stuff: here at least the dependencies need to be specified

Spack in user-space: chaining existing installations into your own Spack environment

NEW + Experimental (work in progress):
module load user_spack

Example 2: create your own new package inside your own repository.

E.g. libgeotiff
Recently moved to github, version that comes built in spack is too old for your purpose

Depending on the complexity the package Implementing package.py

- may be very easy
- may become more difficult

But in many cases it is doable

```
cm2login3~>spack spec -lINt libgeotiff
Input spec
--------------------------------
 -   [    ]  .libgeotiff

Concretized
--------------------------------
 -   cprryjt  [    ]  mine-15.4.libgeotiff@1.6.0%gcc@8.4.0 build_type=RelWithDebInfo arch=linux-sles15-haswell
[^]  fi3lvva  [b   ]      ^builtin.cmake@3.16.5%gcc@8.4.0~doc+ncurses+openssl+ownlibs~qt patches=1c540040c7e203
dd8e27aa20345ecb07fe06570d56410a24a266ae570b1c4c39 arch=linux-sles15-haswell
[^]  6qhv5ta  [bl  ]          ^fixes015x.ncurses@6.2%gcc@8.4.0~symlinks+termlib arch=linux-sles15-haswell
[^]  cfijkws  [b   ]              ^builtin.pkgconf@1.7.3%gcc@8.4.0 arch=linux-sles15-haswell
[^]  jpko756  [bl  ]          ^builtin.openssl@1.1.1g%gcc@8.4.0+systemcerts arch=linux-sles15-haswell
[^]  bhpjih4  [b  t]              ^builtin.perl@5.30.3%gcc@8.4.0+cpanm+shared+threads arch=linux-sles15-haswell
[^]  szzheyp  [bl  ]                  ^builtin.gdbm@1.18.1%gcc@8.4.0 arch=linux-sles15-haswell
[^]  3kfx6pu  [bl  ]                      ^builtin.readline@8.0%gcc@8.4.0 arch=linux-sles15-haswell
[^]  m2bfsoy  [bl  ]              ^builtin.zlib@1.2.11%gcc@8.4.0+optimize+pic+shared arch=linux-sles15-haswell
[^]  fvj645l  [bl  ]      ^builtin.libjpeg-turbo@2.0.4%gcc@8.4.0 arch=linux-sles15-haswell
[^]  q7vii4v  [b   ]          ^builtin.nasm@2.14.02%gcc@8.4.0 arch=linux-sles15-haswell
[^]  zltgjjg  [bl  ]      ^builtin.libtiff@4.0.10%gcc@8.4.0 arch=linux-sles15-haswell
[^]  mz5q6pl  [bl  ]          ^builtin.xz@5.2.5%gcc@8.4.0 arch=linux-sles15-haswell
[^]  qrrrav5  [bl  ]      ^builtin.proj@6.3.1%gcc@8.4.0 arch=linux-sles15-haswell
[^]  ophpcos  [bl  ]          ^builtin.sqlite@3.31.1%gcc@8.4.0+column_metadata+fts~functions~rtree arch=linux-s
les15-haswell

cm2login3~>spack install libgeotiff
[+] /dss/dsshome1/lrz/sys/spack/release/21.1.1/opt/haswell/pkgconf/1.7.3-gcc-cfijkws
```

```
[+] /dss/dsshome1/lrz/sys/spack/release/21.1.1/opt/haswell/libtiff/4.0.10-gcc-zltgjjg
==> Installing libgeotiff
==> No binary for libgeotiff found: installing from source
==> libgeotiff: Executing phase: 'cmake'
==> libgeotiff: Executing phase: 'build'
==> libgeotiff: Executing phase: 'install'
[+] /dss/dsshome1/          spack/opt/linux-sles15-haswell/libgeotiff/1.6.0-gcc-8.4.0-cprryjt
```

Spack: a few words on dynamic linking

Priority-ordering of dynamic linking:

1. LD_PRELOAD
2. RPATH
3. LD_LIBRARY_PATH
4. RUNPATH

Spack uses RPATH as default:

- pathes where to find libraries are coded into the executables & libraries
- executables and libraries are functional without setting up einvironment:
  - -> the binaries know where to look for their dependency-libraries

installed libgeotiff as example here:

```
cm2devel~>ldd $HOME/spack/opt/linux-sles15-haswell/libgeotiff/1.6.0-gcc-8.4.0-cprryjt/bin/makegeo
        linux-vdso.so.1 (0x00007fffa35da000)
        libtiff.so.5 => /dss/dsshome1/lrz/sys/spack/release/21.1.1/opt/haswell/libtiff/4.0.10-gcc-zltgjjg/lib/libtiff.so.5 (0x00007f301adcd000)
        libproj.so.15 => /dss/dsshome1/lrz/sys/spack/release/21.1.1/opt/haswell/proj/6.3.1-gcc-qrrrav5/lib/libproj.so.15 (0x00007f301a910000)
        libc.so.6 => /lib64/libc.so.6 (0x00007f301a555000)
        libwebp.so.6 => /usr/lib64/libwebp.so.6 (0x00007f301a2f7000)
        liblzma.so.5 => /dss/dsshome1/lrz/sys/spack/release/21.1.1/opt/haswell/xz/5.2.5-gcc-mz5q6pl/lib/liblzma.so.5 (0x00007f301a0d1000)
        libjpeg.so.62 => /dss/dsshome1/lrz/sys/spack/release/21.1.1/opt/haswell/libjpeg-turbo/2.0.4-gcc-fvj645l/lib64/libjpeg.so.62 (0x00007f30
        libz.so.1 => /dss/dsshome1/lrz/sys/spack/release/21.1.1/opt/haswell/zlib/1.2.11-gcc-m2bfsoy/lib/libz.so.1 (0x00007f3019c23000)
        libm.so.6 => /lib64/libm.so.6 (0x00007f30198eb000)
        libsqlite3.so.0 => /dss/dsshome1/lrz/sys/spack/release/21.1.1/opt/haswell/sqlite/3.31.1-gcc-ophpcos/lib/libsqlite3.so.0 (0x00007f30195d
        libdl.so.2 => /lib64/libdl.so.2 (0x00007f30193cf000)
        libpthread.so.0 => /lib64/libpthread.so.0 (0x00007f30191b0000)
        libstdc++.so.6 => /dss/dsshome1/lrz/sys/spack/release/21.1.0/opt/x86_64/gcc/8.4.0-gcc-656wch7/lib64/libstdc++.so.6 (0x00007f3018e27000)
        libgcc_s.so.1 => /dss/dsshome1/lrz/sys/spack/release/21.1.0/opt/x86_64/gcc/8.4.0-gcc-656wch7/lib64/libgcc_s.so.1 (0x00007f3018c0f000)
        /lib64/ld-linux-x86-64.so.2 (0x00007f301b047000)
cm2devel~>readelf -d $HOME/spack/opt/linux-sles15-haswell/libgeotiff/1.6.0-gcc-8.4.0-cprryjt/bin/makegeo

Dynamic section at offset 0x4dc8 contains 28 entries:
  Tag        Type                         Name/Value
 0x0000000000000001 (NEEDED)             Shared library: [libtiff.so.5]
 0x0000000000000001 (NEEDED)             Shared library: [libproj.so.15]
 0x0000000000000001 (NEEDED)             Shared library: [libc.so.6]
 0x000000000000000f (RPATH)              Library rpath: [/dss/dsshome1/lrz/sys/spack/release/21.1.0/opt/x86_64/gcc/8.4.0-gcc-656wch7/lib:/dss/d
rz/sys/spack/release/21.1.0/opt/x86_64/gcc/8.4.0-gcc-656wch7/lib64:/dss/dsshome1/0D/di34faf/spack/opt/linux-sles15-haswell/libgeotiff/1.6.0-gcc
rryjt/lib:/dss/dsshome1/0D/di34faf/spack/opt/linux-sles15-haswell/libgeotiff/1.6.0-gcc-8.4.0-cprryjt/lib64:/dss/dsshome1/lrz/sys/spack/release/
t/haswell/libtiff/4.0.10-gcc-zltgjjg/lib:/dss/dsshome1/lrz/sys/spack/release/21.1.1/opt/haswell/xz/5.2.5-gcc-mz5q6pl/lib:/dss/dsshome1/lrz/sys/
ease/21.1.1/opt/haswell/zlib/1.2.11-gcc-m2bfsoy/lib:/dss/dsshome1/lrz/sys/spack/release/21.1.1/opt/haswell/proj/6.3.1-gcc-qrrrav5/lib:/dss/dssh
sys/spack/release/21.1.1/opt/haswell/sqlite/3.31.1-gcc-ophpcos/lib:/dss/dsshome1/lrz/sys/spack/release/21.1.1/opt/haswell/readline/8.0-gcc-3kfx
dss/dsshome1/lrz/sys/spack/release/21.1.1/opt/haswell/ncurses/6.2-gcc-6qhv5ta/lib:/dss/dsshome1/lrz/sys/spack/release/21.1.1/opt/haswell/libjpe
.0.4-gcc-fvj645l/lib64:/dss/dsshome1/0D/di34faf/spack/opt/linux-sles15-haswell/libgeotiff/1.6.0-gcc-8.4.0-cprryjt/lib:/dss/dsshome1/0D/di34faf/
/linux-sles15-haswell/libgeotiff/1.6.0-gcc-8.4.0-cprryjt/lib64:/dss/dsshome1/lrz/sys/spack/release/21.1.1/opt/haswell/readline/8.0-gcc-3kfx6pu/
dsshome1/lrz/sys/spack/release/21.1.1/opt/haswell/sqlite/3.31.1-gcc-ophpcos/lib:/dss/dsshome1/lrz/sys/spack/release/21.1.1/opt/haswell/libjpeg-
.4-gcc-fvj645l/lib64:/dss/dsshome1/lrz/sys/spack/release/21.1.1/opt/haswell/zlib/1.2.11-gcc-m2bfsoy/lib:/dss/dsshome1/lrz/sys/spack/release/21.
aswell/ncurses/6.2-gcc-6qhv5ta/lib:/dss/dsshome1/lrz/sys/spack/release/21.1.1/opt/haswell/xz/5.2.5-gcc-mz5q6pl/lib:/dss/dsshome1/lrz/sys/spack/
1.1.0/opt/x86_64/gcc/8.4.0-gcc-656wch7/lib/gcc/x86_64-pc-linux-gnu/8.4.0]
 0x000000000000000c (INIT)               0x401aa8
 0x000000000000000d (FINI)               0x403844
 0x0000000000000019 (INIT_ARRAY)         0x604d10
 0x000000000000001b (INIT_ARRAYSZ)       8 (bytes)
 0x000000000000001a (FINI_ARRAY)         0x604d18
 0x000000000000001c (FINI_ARRAYSZ)       8 (bytes)
 0x0000000000000004 (HASH)               0x400278
```

Spack in user-space:
chaining existing installation into your own spack environment

```
cm2login3~>module load hdf5/1.8.22-gcc8-impi
Autoloading numactl/2.0.12-gcc8

Loading hdf5/1.8.22-gcc8-impi
  Loading requirement: numactl/2.0.12-gcc8
cm2login3~>spack spec -lI $HDF5_SPEC
Input spec
--------------------------------
[^]  hdf5@1.8.22%gcc@8.4.0+cxx~debug+fortran+hl~java+mpi+pic+shared+szip+threadsafe api=none a
[^]      ^intel-mpi@2019.8.254%gcc@8.4.0 arch=linux-sles15-haswell
[^]      ^libszip@2.1.1%gcc@8.4.0 arch=linux-sles15-haswell
[^]      ^numactl@2.0.12%gcc@8.4.0 arch=linux-sles15-haswell
[^]      ^zlib@1.2.11%gcc@8.4.0+optimize+pic+shared arch=linux-sles15-haswell

Concretized
--------------------------------
[^]  3lmvxrf  hdf5@1.8.22%gcc@8.4.0+cxx~debug+fortran+hl~java+mpi+pic+shared+szip+threadsafe a
[^]  cyojcvv      ^intel-mpi@2019.8.254%gcc@8.4.0 arch=linux-sles15-haswell
[^]  o62frdt      ^libszip@2.1.1%gcc@8.4.0 arch=linux-sles15-haswell
[^]  wz47lgr      ^numactl@2.0.12%gcc@8.4.0 arch=linux-sles15-haswell
[^]  m2bfsoy      ^zlib@1.2.11%gcc@8.4.0+optimize+pic+shared arch=linux-sles15-haswell

cm2login3~>spack spec -lI hdf5@1.8.22%gcc@8.4.0+cxx+debug+fortran+hl~java+mpi+pic+shared+szip+
Input spec
--------------------------------
 -   hdf5@1.8.22%gcc@8.4.0+cxx+debug+fortran+hl~java+mpi+pic+shared+szip+threadsafe

Concretized
--------------------------------
 -   4exl2a5  hdf5@1.8.22%gcc@8.4.0+cxx+debug+fortran+hl~java+mpi+pic+shared+szip+threadsafe a
[^]  cyojcvv      ^intel-mpi@2019.8.254%gcc@8.4.0 arch=linux-sles15-haswell
[^]  o62frdt      ^libszip@2.1.1%gcc@8.4.0 arch=linux-sles15-haswell
[^]  wz47lgr      ^numactl@2.0.12%gcc@8.4.0 arch=linux-sles15-haswell
[^]  jns7liw          ^autoconf@2.69%gcc@8.4.0 arch=linux-sles15-haswell
[^]  6vxvnrt              ^m4@1.4.18%gcc@8.4.0+sigsegv patches=3877ab548f88597ab2327a2230ee048
aswell
[^]  gv36h32                  ^libsigsegv@2.12%gcc@8.4.0 arch=linux-sles15-haswell
[^]  bhpjih4              ^perl@5.30.3%gcc@8.4.0+cpanm+shared+threads arch=linux-sles15-haswel
[^]  szzheyp                  ^gdbm@1.18.1%gcc@8.4.0 arch=linux-sles15-haswell
[^]  3kfx6pu                      ^readline@8.0%gcc@8.4.0 arch=linux-sles15-haswell
[^]  6qhv5ta                          ^ncurses@6.2%gcc@8.4.0~symlinks+termlib arch=linux-sles1
[^]  cfijkws                              ^pkgconf@1.7.3%gcc@8.4.0 arch=linux-sles15-haswell
[^]  zzoup2h          ^automake@1.16.2%gcc@8.4.0 arch=linux-sles15-haswell
[^]  4nya677          ^libtool@2.4.6%gcc@8.4.0 arch=linux-sles15-haswell
[^]  m2bfsoy      ^zlib@1.2.11%gcc@8.4.0+optimize+pic+shared arch=linux-sles15-haswell
```

NEW + Experimental (work in progress):
module load user_spack

Example 3:

Install existing installation in a different variant: here -- with debug-option: +debug

Spack-generated environment modules at LRZ provide a variable <package>_SPEC that holds location of the input/concretized spack-spec dumped in a yaml-file: spec.yaml

One may use this to see details of the installation behind the module: via the spack spec -command

Spack in user-space: chaining existing installation into your own spack environment

```
cm2login3~>module unload hdf5/1.8.22-gcc8-impi numactl/2.0.12-gcc8
cm2login3~>spack install hdf5@1.8.22%gcc@8.4.0+cxx+debug+fortran+hl~java+mpi+pic+shared+szip+threadsafe
[+] /dss/dsshome1/lrz/sys/spack/release/21.1.1/opt/haswell/intel-mpi/2019.8.254-gcc-cyojcvv
[+] /dss/dsshome1/lrz/sys/spack/release/21.1.1/opt/haswell/intel-mpi/2019.8.254-gcc-cyojcvv
[+] /dss/dsshome1/lrz/sys/spack/release/21.1.1/opt/haswell/libszip/2.1.1-gcc-o62frdt
```

```
[+] /dss/dsshome1/lrz/sys/spack/release/21.1.1/opt/haswell/numactl/2.0.12-gcc-wz47lgr
==> Installing hdf5
==> No binary for hdf5 found: installing from source
==> hdf5: Executing phase: 'autoreconf'
==> hdf5: Executing phase: 'configure'
==> hdf5: Executing phase: 'build'
==> hdf5: Executing phase: 'install'
[+] /dss/dsshome1/        /spack/opt/linux-sles15-haswell/hdf5/1.8.22-gcc-8.4.0-4exl2a5
```

```
cm2login3~>ls spack/opt/linux-sles15-haswell/hdf5/1.8.22-gcc-8.4.0-4exl2a5/lib*
libhdf5.a               libhdf5_fortran.so.10      libhdf5hl_fortran.a          libhdf5_hl.so.10.2.3
libhdf5_cpp.a           libhdf5_fortran.so.10.0.7  libhdf5hl_fortran.la         libhdf5.la
libhdf5_cpp.la          libhdf5_hl.a               libhdf5_hl_fortran.so        libhdf5.settings
libhdf5_cpp.so          libhdf5_hl_cpp.a           libhdf5hl_fortran.so         libhdf5.so
libhdf5_cpp.so.16       libhdf5_hl_cpp.la          libhdf5hl_fortran.so.10      libhdf5.so.10
libhdf5_cpp.so.16.0.1   libhdf5_hl_cpp.so          libhdf5hl_fortran.so.10.0.6  libhdf5.so.10.4.0
libhdf5_fortran.a       libhdf5_hl_cpp.so.11       libhdf5_hl.la
libhdf5_fortran.la      libhdf5_hl_cpp.so.11.1.3   libhdf5_hl.so
libhdf5_fortran.so      libhdf5_hl_fortran.a       libhdf5_hl.so.10
```

NEW + Experimental (work in progress):
module load user_spack

Example 3 from previous slide continued

Spack self documenting artifacts

`.spack` directory in all installation-paths:
-> usefull information from installation process is available

Lets inspect this for our own hdf5 installation :

```
cm2login3~>ls spack/opt/linux-sles15-haswell/hdf5/1.8.22-gcc-8.4.0-4exl2a5/.spack/
```

```
archived-files
install_manifest.json
repos
spack-build-env.txt
spack-build-out.txt
spack-configure-args.txt
spec.yaml
cm2login3~>
```

- `archived-files`contains log of configure-phase (if avail)
- `repos` contains all procedures (package.py‘s) used for installation (package + all deps)
- `spack-build-env.txt` -- dump of environment during installation
- `spack-build-out.txt` -- dump of output-stream from installation
- `spack-configure-args` -- dump of configure arguments
- `spec.yaml` -- dictionary with input and concretized spack-specs

Spack commands (subset) that may be usefull for your work

```
query packages:
  list            list and search available packages
  info            get detailed information on a
                  particular package
  find            list and search installed packages

build packages:
  install         build and install packages
  uninstall       remove installed packages
  dev-build       developer build: build from code
                  in current working directory
  spec            show what would be installed,
                  given a spec

container:
  containerize    creates recipes to build images
                  for different container runtimes
environments:
  env               manage virtual environments
```

```
create packages:
  create          create a new package file
  edit            open package files in $EDITOR

system:
  compilers       list available compilers

user environment:
  load            add package to the user
                  environment
  module          manipulate module files
  unload          remove package from the user
                  environment
configuration:
  config          get and set configuration options
  repo            manage package source repositories
```

Automated stack builds with Spack: Improving our documentation

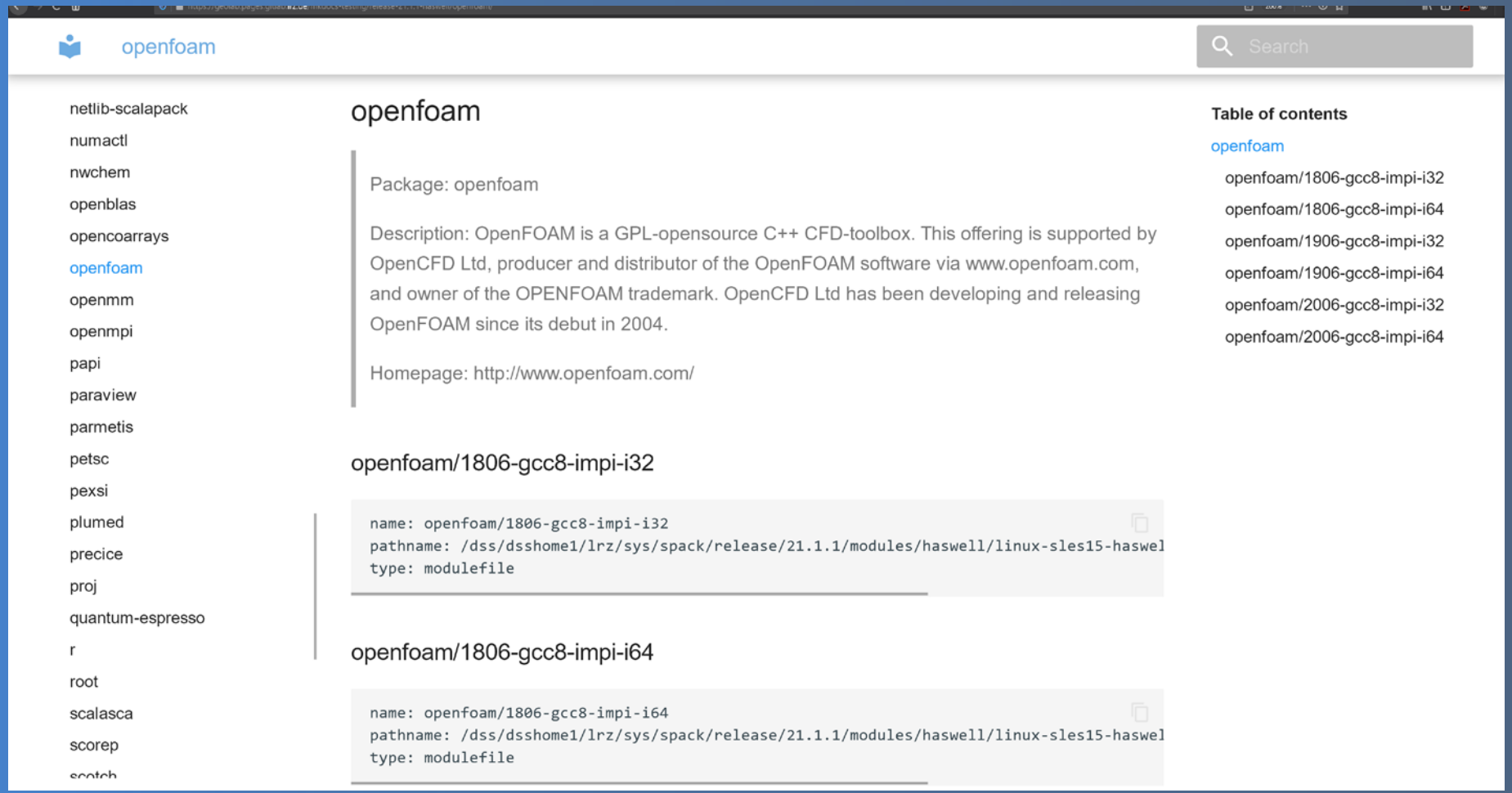
Work in Progress:

- evaluating auto-generated documentation for a **basic** or an advanced view

  Basic

… with basic info on what‘s installed

  - Package info
  - installed versions
  - url to homepage

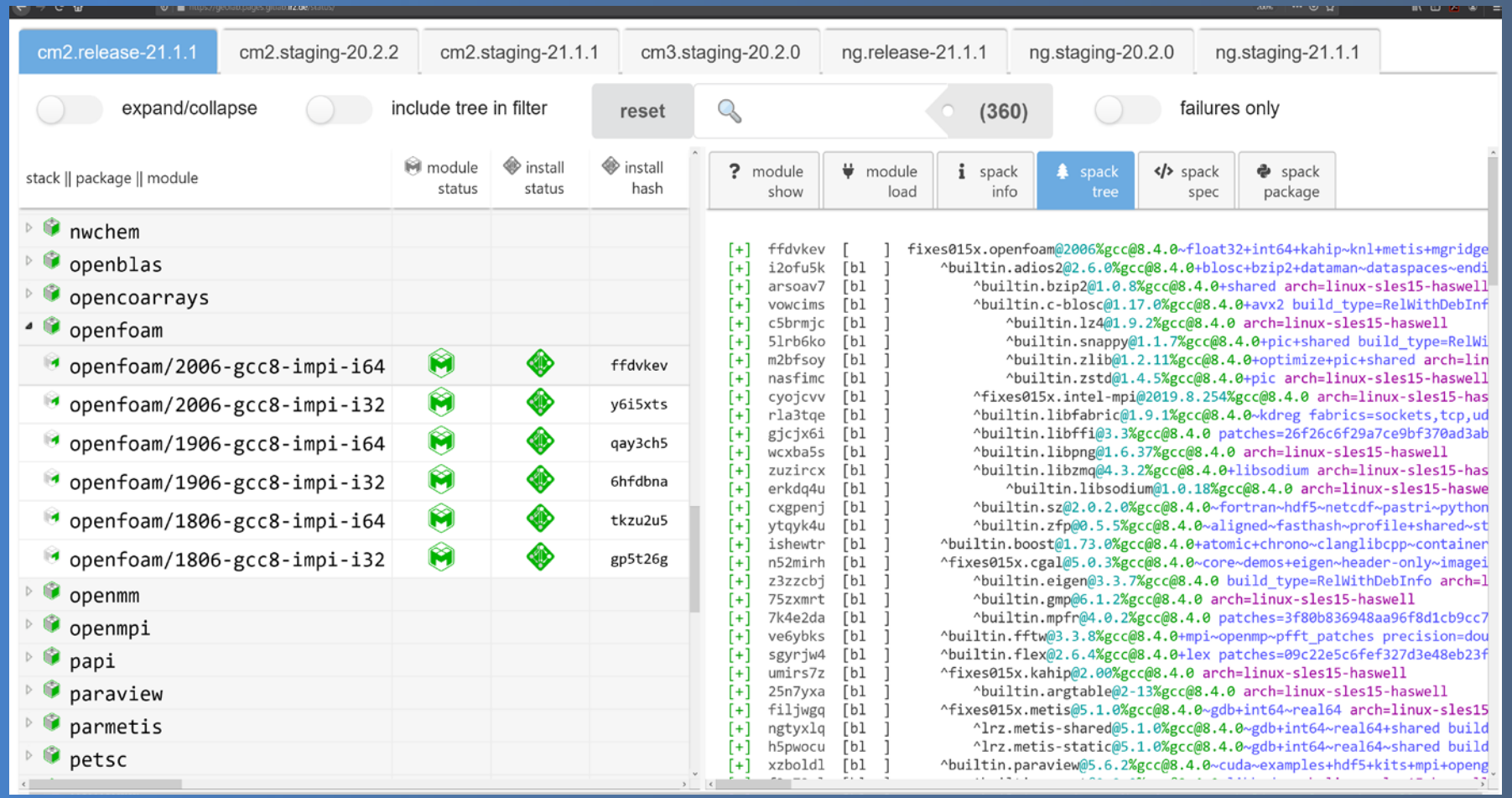


Automated stack builds with Spack: Improving Transparency

Work in Progress:

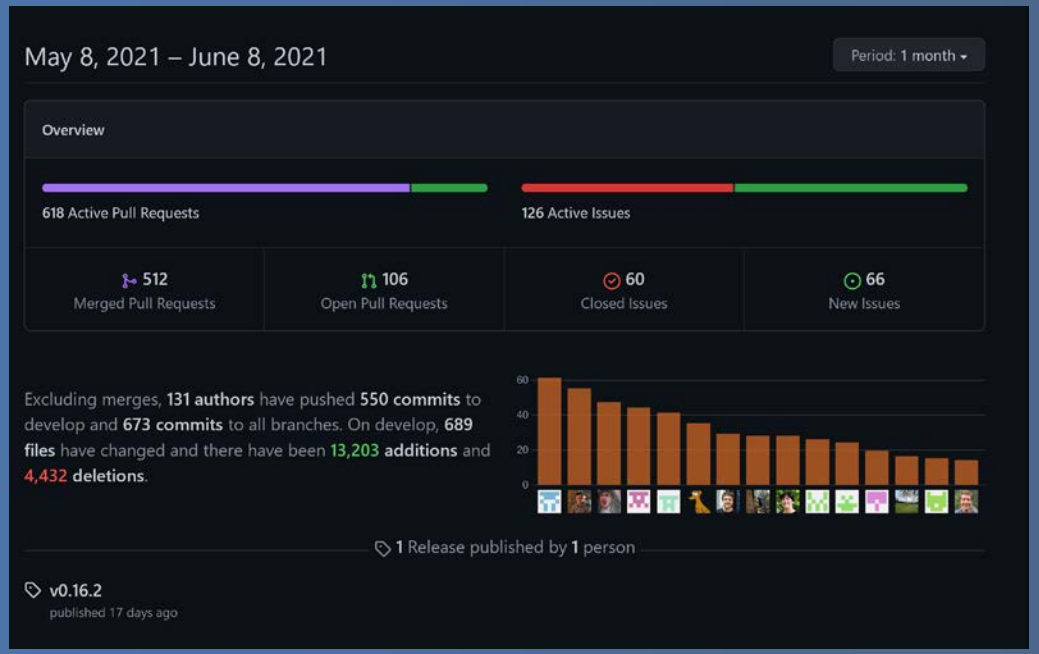
- evaluating auto-generated documentation for a basic or an **advanced** view
- Advanced:

  … with details on installations:
  - installed package
  - dependencies
  - package configuration
  - reference to external docs
  - reference to source origin



Spack is open-source with many community contributions

- Spack has excellent documentation: https://spack.readthedocs.io/en/v0.15.4/
- Spack community gives strong support via slack https://slack.spack.io/
- Spack repository is hosted on github: https://github.com/spack/spack
  - Spack is under heavy development
    - spack-developers
    - application-developers
    - domain-scientists
    - HPC-support-staff
    - hardware-vendors
- Consider yourself becoming part of the community:
  - Contributing and benefitting from



- LRZ Documentation on spack in user-space (updates pending) https://doku.lrz.de/display/PUBLIC/Building+software+in+user+space+with+spack