ALCF DEVELOPER SESSIONS

Argonne
NATIONAL LABORATORY

SOFTWARE DEPLOYMENT WITH SPACK

DR FRANK T WILLMORE
Software Integration Admin
ALCF Operations
willmore@anl.gov

U.S. DEPARTMENT OF ENERGY
Argonne National Laboratory is a
U.S. Department of Energy laboratory
managed by UChicago Argonne, LLC.

29 April 2020

SOFTWARE DEPLOYMENT

HOW DO I INSTALL SOFTWARE ON MY LAPTOP?

yum
rpm
apt
zypper
brew

configure
make install

cmake
make install

```
willmore:~$ 
```

WE WANT TO INSTALL ON SUPERCOMPUTERS (IN ADDITION TO LAPTOPS)

Summit Supercomputer
At Oak Ridge National Lab

200 PetaFLOPs Peak

4,608 nodes, each with:

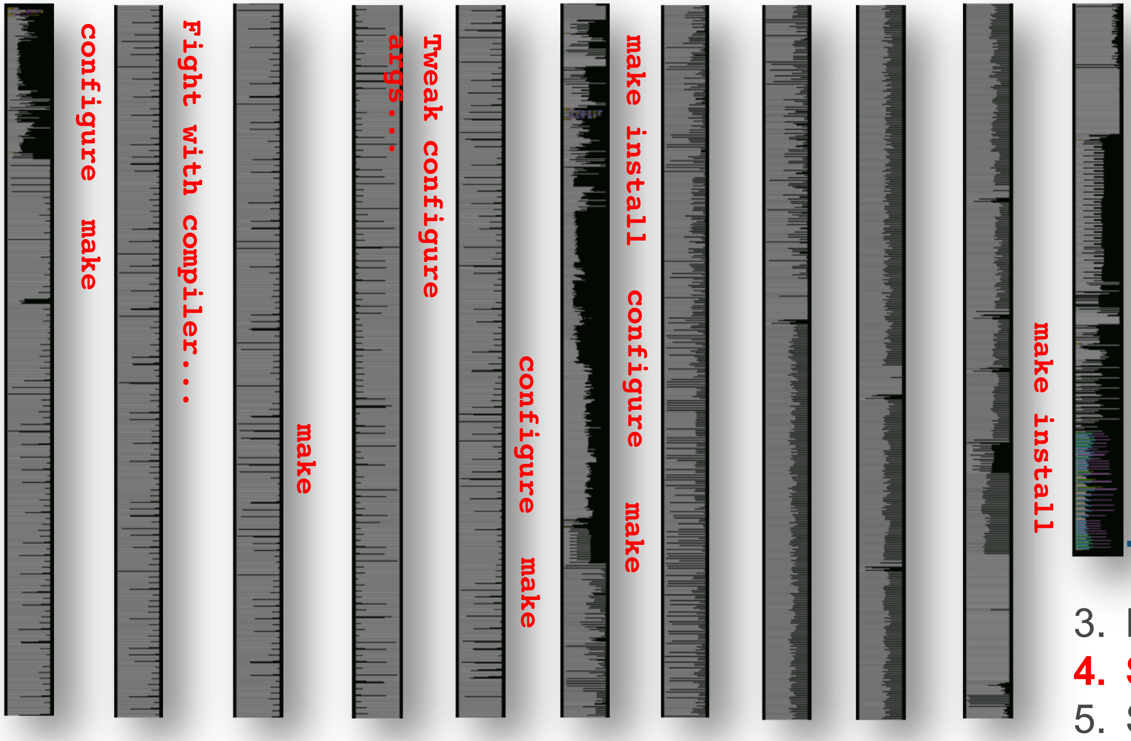
- **48 Power9 cores**
- **6 NVIDIA Volta GPUs**

- large shared system
- multi-user
- multi-architecture
- specialized hardware

HOW TO INSTALL SOFTWARE ON A SUPERCOMPUTER

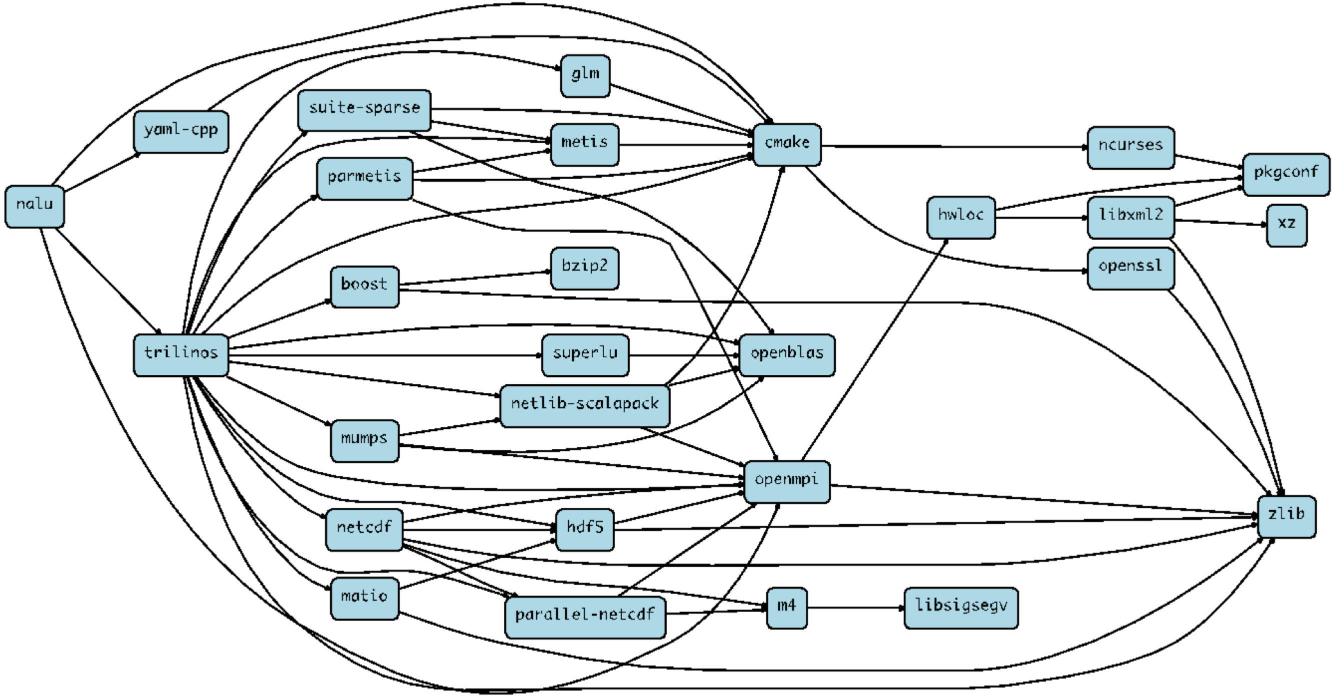
1. Download all 10 (100?) tarballs you need
2. Start building!

- edit configuration files
- edit some system files and hope it doesn't break something else
- update something you don't want to update and hope it doesn't break something else
- re-edit whatever settings you changed for your dependencies, if you can find your notes and remember what you did
- pester a colleague
- hope that software provider didn't change the software without bumping version.



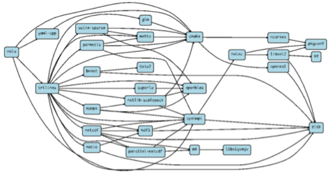
3. Run code
4. **Segfault!?**
5. Start over…

SOFTWARE COMPLEXITY IN HPC IS GROWING

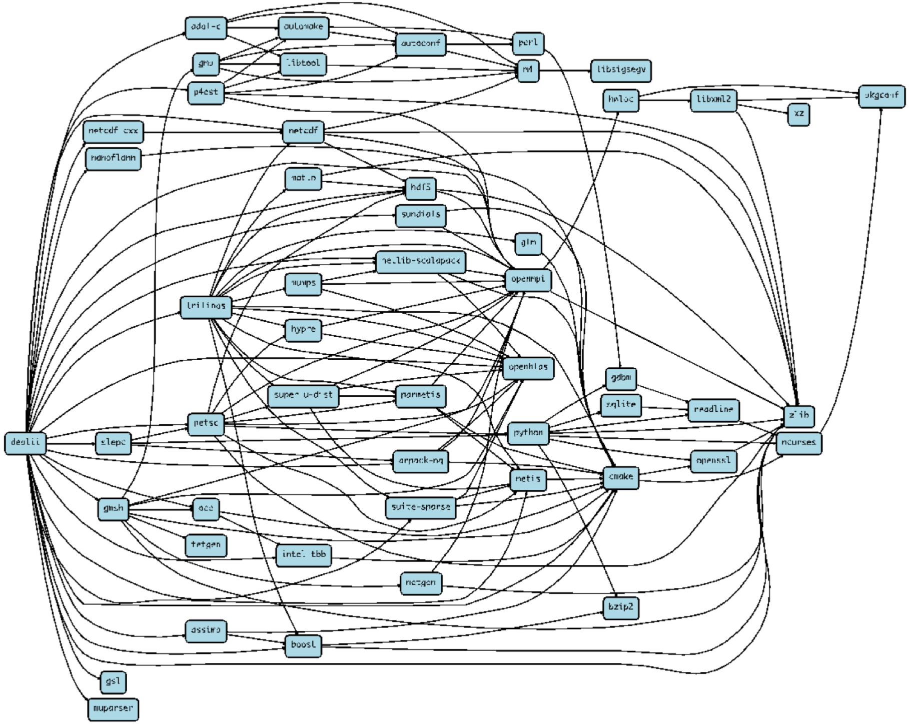


Nalu: Generalized Unstructured Massively Parallel Low Mach Flow

SOFTWARE COMPLEXITY IN HPC IS GROWING

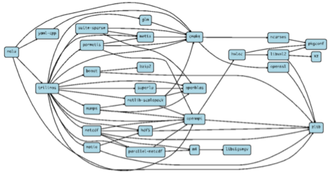


Nalu: Generalized Unstructured Massively Parallel Low Mach Flow

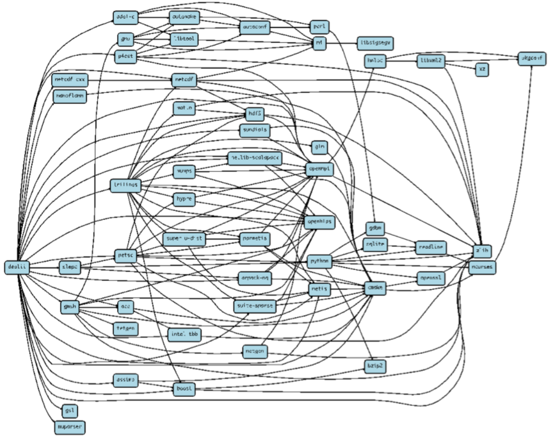


dealii: C++ Finite Element Library

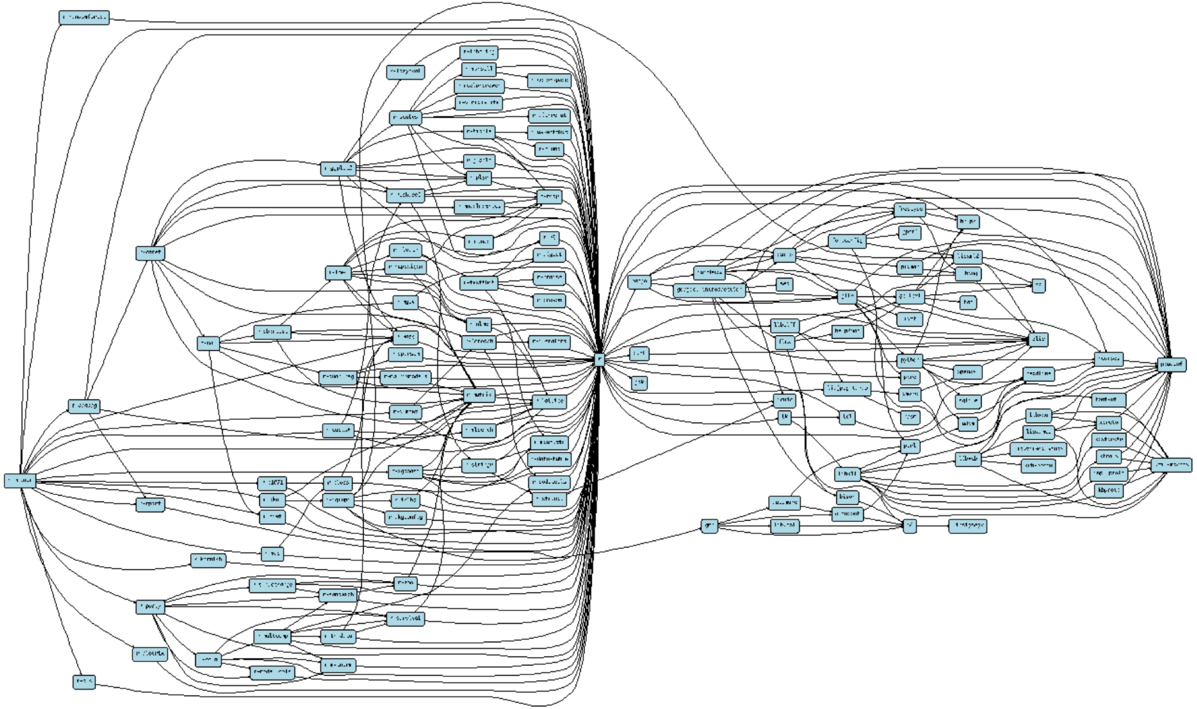
SOFTWARE COMPLEXITY IN HPC IS GROWING



Nalu: Generalized Unstructured Massively Parallel Low Mach Flow



dealii: C++ Finite Element Library



R Miner: R Data Mining Library

WHAT ABOUT MODULES?

- Most supercomputers deploy some form of *environment modules*
  - TCL modules (dates back to 1995) and Lmod (from TACC) are the most popular

```
$ gcc
- bash: gcc: command not found

$ module load gcc/7.0.1
$ gcc –dumpversion
7.0.1
```

- Modules don’t handle installation!
  - They only modify your environment (things like PATH, LD_LIBRARY_PATH, etc.)
- Someone (likely a team of people) has already installed gcc for you!
  - Also, you can *only* `module load` the things they’ve installed

WHAT ABOUT CONTAINERS?

- **Containers provide a great way to reproduce and distribute an already-built software stack**
  - Promises layer of abstraction at kernel level
  - OK, well, maybe higher up the stack, like above the MPI layer?
- **Someone needs to build the container!**
  - This isn't trivial
  - Containerized applications still have hundreds of dependencies
- **Using the OS package manager inside a container is insufficient**
  - Most binaries are built unoptimized
  - Generic binaries, not optimized for specific architectures (e.g. no AVX-512 vectorization)
- **Developing with an OS software stack can be painful**
  - Little freedom to choose versions
  - Little freedom to choose compiler options, build options, etc. for packages

We need something more flexible to **build** the containers

SPACK

WHAT IS SPACK?

- *Spack is a package manager that enables and automates the deployment of multiple versions of a software package to a single installation, including builds for different microprocessor architectures, different numbered versions, and builds with different options. Spack is developed as open-source software with support from the US Department of Energy and is adopted as the primary scientific software deployment tool in the Exascale Computing Project. It also has a rapidly growing base of users outside of the DOE, ranging from large HPC installations worldwide, vendor adopters, and a committed base of users needing newer versions of software than are provided with typical OS distributions.*

SPACK IS A FLEXIBLE PACKAGE MANAGER FOR HPC

- How to install Spack:

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

- How to install a package:

```
$ spack install hdf5
```

- HDF5 and its dependencies are installed within the Spack directory.
- Unlike typical package managers, Spack can also install many variants of the same build.
  - Different compilers
  - Different MPI implementations
  - Different build options

Get Spack!

http://github.com/spack/spack

@spackpm

WHO CAN USE SPACK?

People who want to use or distribute software for HPC!

1. **End Users of HPC Software**
   - Install and run HPC applications and tools
2. **HPC Application Teams**
   - Manage third-party dependency libraries
3. **Package Developers**
   - People who want to package their own software for distribution
4. **User support teams at HPC Centers**
   - People who deploy software for users at large HPC sites

SPACK PROVIDES A *SPEC* SYNTAX TO DESCRIBE CUSTOMIZED CONFIGURATIONS

```
$ spack install mpileaks                          unconstrained
$ spack install mpileaks@3.3                        @ custom version
$ spack install mpileaks@3.3 %gcc@4.7.3             % custom compiler
$ spack install mpileaks@3.3 %gcc@4.7.3 +threads     +/- build option
$ spack install mpileaks@3.3 cppflags="-O3 –g3"      set compiler flags
$ spack install mpileaks@3.3 target=skylake         set target microarchitecture
$ spack install mpileaks@3.3 ^mpich@3.2 %gcc@4.9.3   ^ dependency information
```

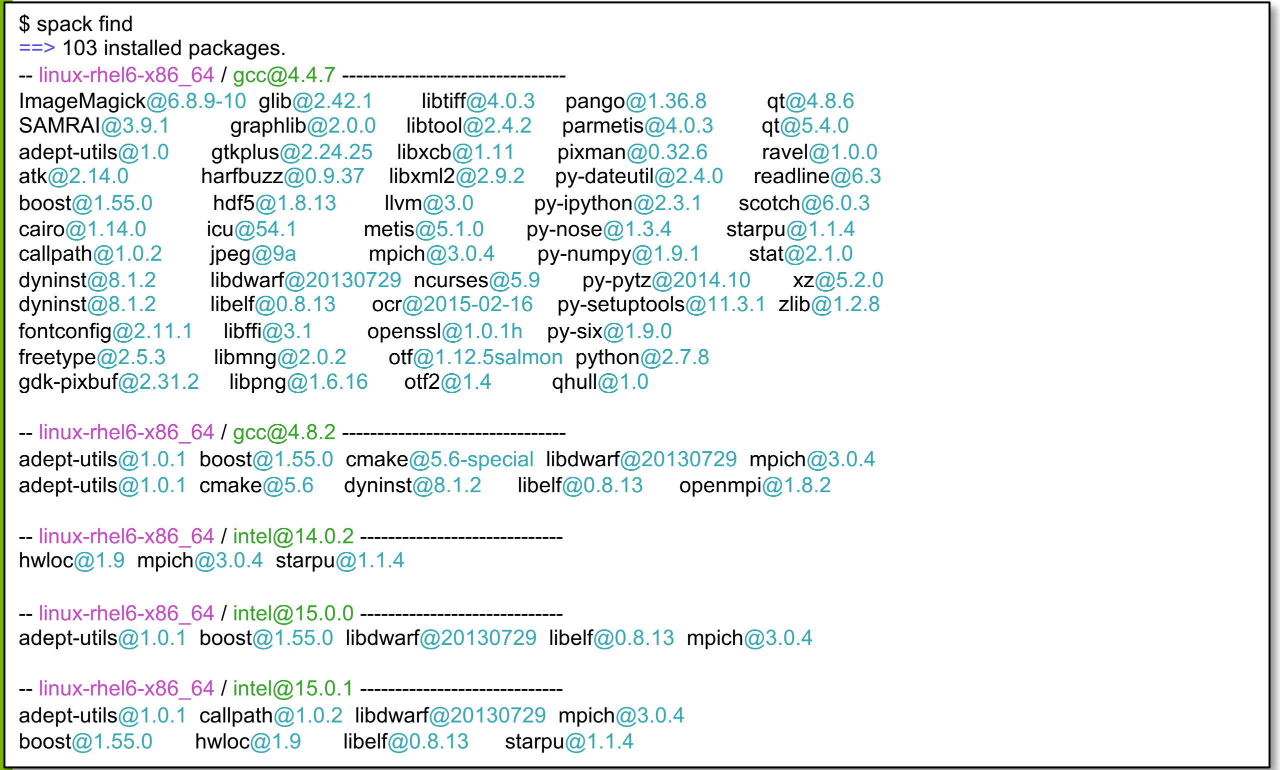
- Each expression is a ***spec*** for a particular configuration
  - Each clause adds a constraint to the spec
  - Constraints are optional – specify only what you need.
  - Customize install on the command line!
- Spec syntax is recursive
  - Full control over the combinatorial build space

`SPACK LIST` SHOWS WHAT PACKAGES ARE AVAILABLE

```
$ spack list
==> 303 packages.
activeharmony   cgal             fish         gtkplus      libgd          mesa              openmpi          py-coverage        py-pycparser          qt         tcl
adept-utils     cgm              flex         harfbuzz     libgpg-error   metis             openspeedshop    py-cython          py-pyelftools         qthreads   texinfo
apex            cityhash         fltk         hdf          libjpeg-turbo  Mitos             openssl          py-dateutil        py-pygments           R          the_silver_searcher
arpack          cleverleaf       flux         hdf5         libjson-c      mpc               otf              py-epydoc          py-pylint             ravel      thrift
asciidoc        cloog            fontconfig   hwloc        libmng         mpe2              otf2             py-funcsigs        py-pypar              readline   tk
atk             cmake            freetype     hypre        libmonitor     mpfr              pango            py-genders         py-pyparsing          rose       tmux
atlas           cmocka           gasnet       icu          libNBC         mpibash           papi             py-gnuplot         py-pyqt               rsync      tmuxinator
atop            coreutils        gcc          icu4c        libpciaccess   mpich             paraver          py-h5py            py-pyside             ruby       trilinos
autoconf        cppcheck         gdb          ImageMagick  libpng         mpileaks          paraview         py-ipython         py-pytables           SAMRAI     uncrustify
automaded       cram             gdk-pixbuf   isl          libsodium      mrnet             parmetis         py-libxml2         py-python-daemon      samtools   util-linux
automake        cscope           geos         jdk          libtiff        mumps             parpack          py-lockfile        py-pytz               scalasca   valgrind
bear            cube             gflags       jemalloc     libtool        munge             patchelf         py-mako            py-rpy2               scorep     vim
bib2xhtml       curl             ghostscript  jpeg         libunwind      muster            pcre             py-matplotlib      py-scientificpython   scotch     vtk
binutils        czmq             git          judy         libuuid        mvapich2          pcre2            py-mock            py-scikit-learn       scr        wget
bison           damselfly        glib         julia        libxcb         nasm              pdt              py-mpi4py          py-scipy              silo       wx
boost           dbus             glm          launchmon    libxml2        ncdu              petsc            py-mx              py-setuptools         snappy     wxpropgrid
bowtie2         docbook-xml      global       lcms         libxshmfence   ncurses           pidx             py-mysqldb1        py-shiboken           sparsehash xcb-proto
boxlib          doxygen          glog         leveldb      libxslt        netcdf            pixman           py-nose            py-sip                spindle    xerces-c
bzip2           dri2proto        glpk         libarchive   llvm           netgauge          pkg-config       py-numexpr         py-six                spot       xz
cairo           dtcmp            gmp          libcerf      llvm-lld       netlib-blas       pmgr_collective  py-numpy           py-sphinx             sqlite     yasm
callpath        dyninst          gmsh         libcircle    lmdb           netlib-lapack     postgresql       py-pandas          py-sympy              stat       zeromq
cblas           eigen            gnuplot      libdrm       lmod           netlib-scalapack  ppl              py-pbr             py-tappy              sundials   zlib
cbtf            elfutils         gnutls       libdwarf     lua            nettle            protobuf         py-periodictable   py-twisted            swig       zsh
cbtf-argonavis  elpa             gperf        libedit      lwgrp          ninja             py-astropy       py-pexpect         py-urwid              szip
cbtf-krell      expat            gperftools   libelf       lwm2           ompss             py-basemap       py-pil             py-virtualenv         tar
cbtf-lanl       extrae           graphlib     libevent     matio          ompt-openmp       py-biopython     py-pillow          py-yapf               task
cereal          exuberant-ctags  graphviz     libffi       mbedtls        opari2            py-blessings     py-pmw             python                taskd
cfitsio         fftw             gsl          libgcrypt    memaxes        openblas          py-cffi          py-pychecker       qhull                 tau
```

- Spack has over 3,900 packages now.

`SPACK FIND` SHOWS WHAT IS INSTALLED



```
$ spack find
==> 103 installed packages.
-- linux-rhel6-x86_64 / gcc@4.4.7 --------------------------------
ImageMagick@6.8.9-10  glib@2.42.1       libtiff@4.0.3     pango@1.36.8         qt@4.8.6
SAMRAI@3.9.1          graphlib@2.0.0    libtool@2.4.2     parmetis@4.0.3       qt@5.4.0
adept-utils@1.0       gtkplus@2.24.25   libxcb@1.11       pixman@0.32.6        ravel@1.0.0
atk@2.14.0            harfbuzz@0.9.37   libxml2@2.9.2     py-dateutil@2.4.0    readline@6.3
boost@1.55.0          hdf5@1.8.13       llvm@3.0          py-ipython@2.3.1     scotch@6.0.3
cairo@1.14.0          icu@54.1          metis@5.1.0       py-nose@1.3.4        starpu@1.1.4
callpath@1.0.2        jpeg@9a           mpich@3.0.4       py-numpy@1.9.1       stat@2.1.0
dyninst@8.1.2         libdwarf@20130729 ncurses@5.9       py-pytz@2014.10      xz@5.2.0
dyninst@8.1.2         libelf@0.8.13     ocr@2015-02-16    py-setuptools@11.3.1 zlib@1.2.8
fontconfig@2.11.1     libffi@3.1        openssl@1.0.1h    py-six@1.9.0
freetype@2.5.3        libmng@2.0.2      otf@1.12.5salmon  python@2.7.8
gdk-pixbuf@2.31.2     libpng@1.6.16     otf2@1.4          qhull@1.0

-- linux-rhel6-x86_64 / gcc@4.8.2 --------------------------------
adept-utils@1.0.1  boost@1.55.0  cmake@5.6-special  libdwarf@20130729  mpich@3.0.4
adept-utils@1.0.1  cmake@5.6     dyninst@8.1.2      libelf@0.8.13      openmpi@1.8.2

-- linux-rhel6-x86_64 / intel@14.0.2 ------------------------------
hwloc@1.9  mpich@3.0.4  starpu@1.1.4

-- linux-rhel6-x86_64 / intel@15.0.0 ------------------------------
adept-utils@1.0.1  boost@1.55.0  libdwarf@20130729  libelf@0.8.13  mpich@3.0.4

-- linux-rhel6-x86_64 / intel@15.0.1 ------------------------------
adept-utils@1.0.1  callpath@1.0.2  libdwarf@20130729  mpich@3.0.4
boost@1.55.0       hwloc@1.9       libelf@0.8.13      starpu@1.1.4
```

- All the versions coexist!
  - Multiple versions of same package are ok.
- Packages are installed to automatically find correct dependencies.
- Binaries work *regardless of user's environment*.
- Spack also generates module files.
  - Don't *have* to use them.

USERS CAN QUERY THE FULL DEPENDENCY CONFIGURATION OF INSTALLED PACKAGES.

```
$ spack find callpath
==> 2 installed packages.
-- linux-rhel6-x86_64 / clang@3.4 ——      -- linux-rhel6-x86_64 / gcc@4.9.2 -------
callpath@1.0.2                            callpath@1.0.2
```

Expand dependencies with spack find -d

```
$ spack find -dl callpath
==> 2 installed packages.
-- linux-rhel6-x86_64 / clang@3.4 -----      -- linux-rhel6-x86_64 / gcc@4.9.2 -----
xv2clz2    callpath@1.0.2                    udltshs    callpath@1.0.2
ckjazss        ^adept-utils@1.0.1            rfsu7fb        ^adept-utils@1.0.1
3ws43m4            ^boost@1.59.0             ybet64y            ^boost@1.55.0
ft7znm6            ^mpich@3.1.4              aa4ar6i            ^mpich@3.1.4
qqnuet3        ^dyninst@8.2.1                tmnnge5        ^dyninst@8.2.1
3ws43m4            ^boost@1.59.0             ybet64y            ^boost@1.55.0
g65rdud            ^libdwarf@20130729        g2mxrl2            ^libdwarf@20130729
cj5p5fk                ^libelf@0.8.13        ynpai3j                ^libelf@0.8.13
cj5p5fk            ^libelf@0.8.13            ynpai3j            ^libelf@0.8.13
g65rdud        ^libdwarf@20130729            g2mxrl2        ^libdwarf@20130729
cj5p5fk            ^libelf@0.8.13            ynpai3j            ^libelf@0.8.13
cj5p5fk        ^libelf@0.8.13                ynpai3j        ^libelf@0.8.13
ft7znm6        ^mpich@3.1.4                  aa4ar6i        ^mpich@3.1.4
```

- Architecture, compiler, versions, and variants may differ between builds.

Not shown: **patches**, **resources**, **conflicts**, other directives.

SPACK PACKAGES ARE *TEMPLATES*
THEY USE A SIMPLE PYTHON DSL TO DEFINE HOW TO BUILD

```
from spack import *

class Kripke(CMakePackage):
    """Kripke is a simple, scalable, 3D Sn deterministic particle
      transport proxy/mini app.
    """

    homepage = "https://computation.llnl.gov/projects/co-design/kripke"
    url      = "https://computation.llnl.gov/projects/co-design/download/kripke-openmp-1.1.tar.gz"

    version(‘1.2.3’, sha256='3f7f2eef0d1ba5825780d626741eb0b3f026a096048d7ec4794d2a7dfbe2b8a6’)
    version(‘1.2.2’, sha256='eaf9ddf562416974157b34d00c3a1c880fc5296fce2aa2efa039a86e0976f3a3’)
    version('1.1’, sha256='232d74072fc7b848fa2adc8a1bc839ae8fb5f96d50224186601f55554a25f64a’)

    variant('mpi',    default=True, description='Build with MPI.’)
    variant('openmp', default=True, description='Build with OpenMP enabled.’)

    depends_on('mpi', when='+mpi’)
    depends_on('cmake@3.0:', type='build’)

    def cmake_args(self):
        return [
            '-DENABLE_OPENMP=%s’ % ('+openmp’ in self.spec),
            '-DENABLE_MPI=%s' % ('+mpi’ in self.spec),
        ]

    def install(self, spec, prefix):
        # Kripke does not provide install target, so we have to copy
        # things into place.
        mkdirp(prefix.bin)
        install('../spack-build/kripke', prefix.bin)
```

Base package (CMake support)

Metadata at the class level

Versions

Variants (build options)

Dependencies (same spec syntax)

Install logic in instance methods

Don’t typically need install() for CMakePackage, but we can work around codes that don’t have it.

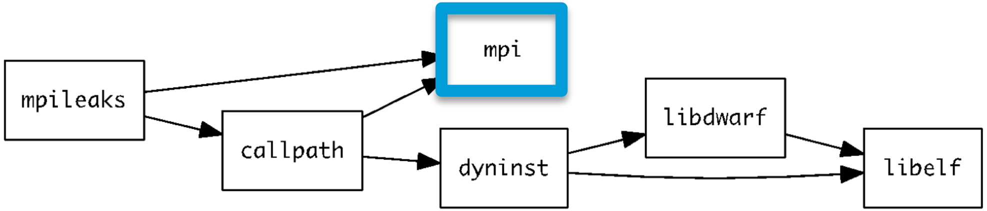
DEPEND ON INTERFACES (NOT IMPLEMENTATIONS) WITH VIRTUAL DEPENDENCIES

- mpi is a *virtual dependency*
- Install the same package built with two different MPI implementations:

```
$ spack install mpileaks ^mvapich
```

```
$ spack install mpileaks ^openmpi@1.4:
```

- Virtual deps are replaced with a valid implementation at resolution time.
  - If the user didn't pick something and there are multiple options, Spack picks.



Virtual dependencies can be versioned:

```
class Mpileaks(Package):
    depends_on("mpi@2:")
```

dependent

```
class Mvapich(Package):
    provides("mpi@1” when="@:1.8")
    provides("mpi@2” when="@1.9:")
```

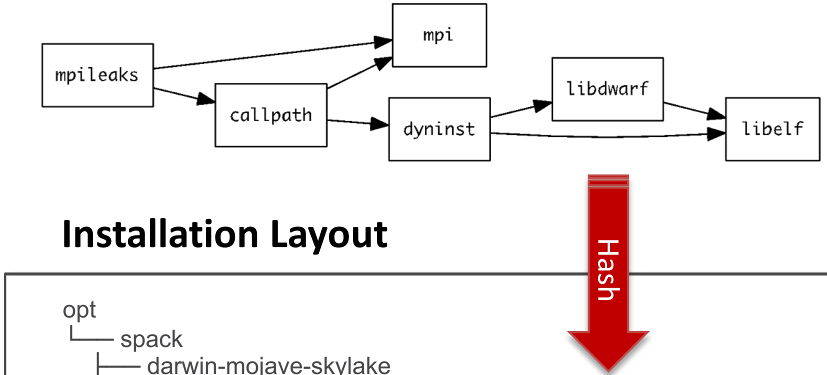
provider

```
class Openmpi(Package):
    provides("mpi@:2.2" when="@1.6.5:")
```

provider

Spack handles combinatorial software complexity

Dependency DAG



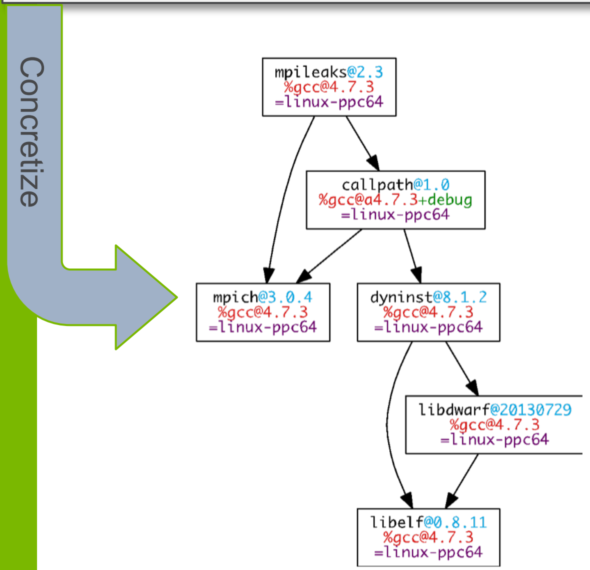
Installation Layout

```
opt
└── spack
    ├── darwin-mojave-skylake
    │   └── clang-10.0.0-apple
    │       ├── bzip2-1.0.8-hc4sm4vuzpm4znmvrfzri4ow2mkphe2e
    │       ├── python-3.7.6-daqqpssxb6qbfrztsezkmhus3xoflbsy
    │       ├── sqlite-3.30.1-u64v26igxvxyn23hysmklfums6tgjv5r
    │       ├── xz-5.2.4-u5eawkvaoc7vonabe6nndkcfwuv233cj
    │       └── zlib-1.2.11-x46q4wm46ay4pltriijbgizxjrhbaka6
    ├── darwin-mojave-x86_64
    │   └── clang-10.0.0-apple
    │       └── coreutils-8.29-pl2kcytejqcys5dzecfrtjqxfdssvnob
```

- Each unique dependency graph is a unique ***configuration***.
- Each configuration in a unique directory.
  - Multiple configurations of the same package can coexist.
- **Hash** of entire directed acyclic graph (DAG) is appended to each prefix.
- Installed packages automatically find dependencies
  - Spack embeds RPATHs in binaries.
  - No need to use modules or set LD_LIBRARY_PATH *at runtime*
  - Things work *the way you built them*

CONCRETIZATION FILLS IN MISSING PARTS OF REQUESTED SPECS.

mpileaks ^callpath@1.0+debug ^libelf@0.8.11

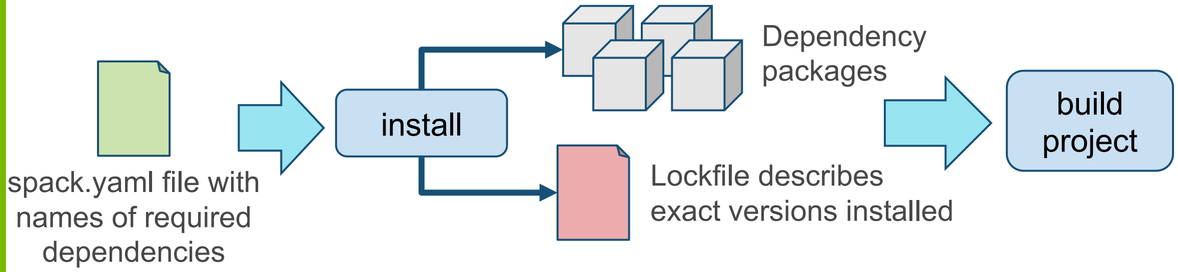


Concrete spec is fully constrained
and can be passed to install.

- Workflow:
  1. Users input only an *abstract* spec with some constraints
  2. Spack makes choices according to policies (site/user/etc.)
  3. Spack installs *concrete* configurations of package + dependencies

- Dependency resolution is an NP-complete problem!
  - Different versions/configurations of packages require different versions/configurations of dependencies
  - Concretizer searches for a configuration that satisfies all the requirements
  - This is basically a SAT/SMT solve

SPACK ENVIRONMENTS MAKE IT EASY TO MANAGE SETS OF DEPENDENCIES FOR MANY USE CASES

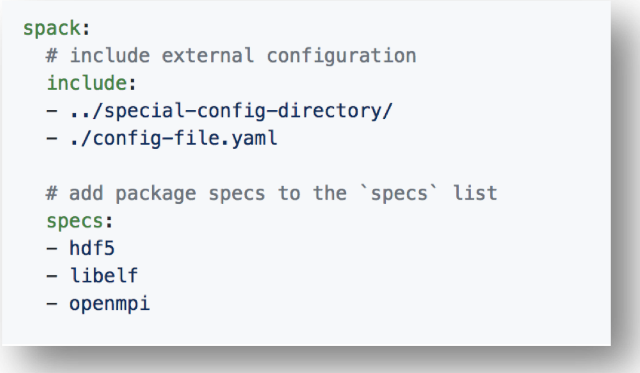

spack.yaml file with names of required dependencies
install
Dependency packages
Lockfile describes exact versions installed
build project

spack.yaml is full of abstract requirements
- Concretize specs in spack.yaml to get spack.lock

Environments are just sets of regular specs, for:
- Creating a work environment for users
- Managing dependencies for developers

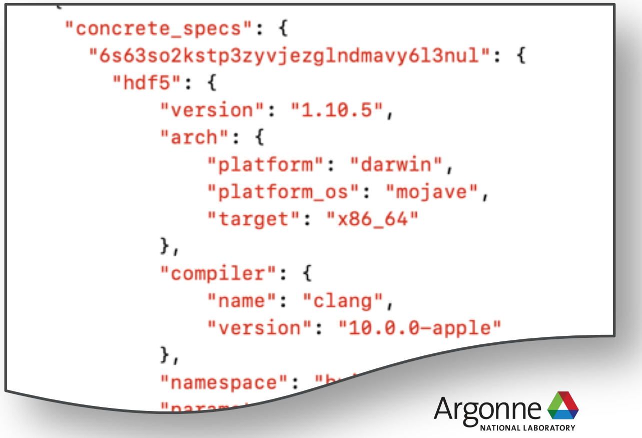
Similar to other dependency management models
- Bundler, pipenv, cargo, npm, etc.

Simple spack.yaml file



```yaml
spack:
  # include external configuration
  include:
  - ../special-config-directory/
  - ./config-file.yaml

  # add package specs to the `specs` list
  specs:
  - hdf5
  - libelf
  - openmpi
```

Concrete spack.lock file (generated)



```json
  "concrete_specs": {
    "6s63so2kstp3zyvjezglndmavy6l3nul": {
      "hdf5": {
          "version": "1.10.5",
          "arch": {
              "platform": "darwin",
              "platform_os": "mojave",
              "target": "x86_64"
          },
          "compiler": {
              "name": "clang",
              "version": "10.0.0-apple"
          },
          "namespace": "b
```

NEW FEATURES IN 0.14.X

SPACK 0.14.0 WAS RELEASED FEBRUARY 23, 2020

- **Lots of new features!**
  - Completely reworked **GitLab pipeline generation**
    - spack ci command
  - **Generate container recipes** from environments
    - spack containerize command
  - **Distributed/parallel builds**
    - srun –N 8 spack install
    - Spack instances coordinate effectively via locks
- **Closely follows Spack 0.13 features (released around the time of SC19)**
  - **Spack stacks:** combinatorial environments for facility deployment
  - Spack detects and builds for **specific microarchitectures**
  - **Chaining**: use dependencies from external "upstream" Spack instances

EVER TRIED TO FIGURE OUT WHAT YOUR PROCESSOR IS?

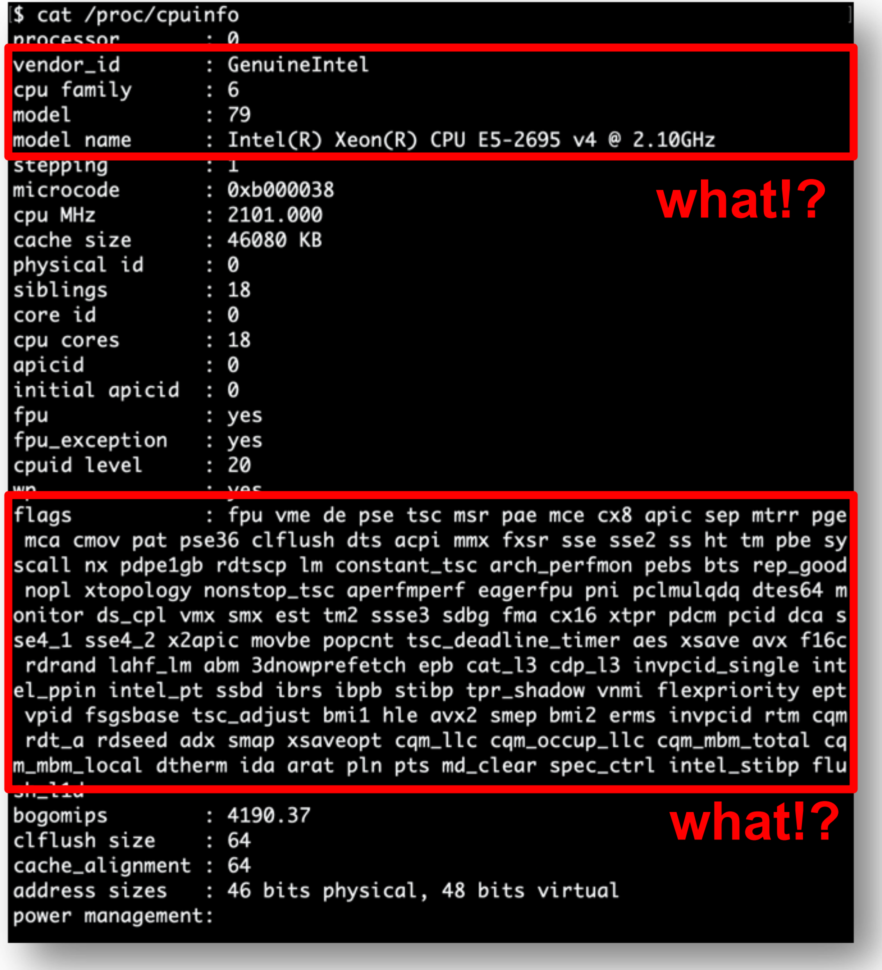
You can get a lot of information from:

- **/proc/cpuinfo** on linux
- sysctl tool on macs

But it’s not exactly intuitive

Humans call this architecture “broadwell”

oh.



```
$ cat /proc/cpuinfo
processor       : 0
vendor_id       : GenuineIntel
cpu family      : 6
model           : 79
model name      : Intel(R) Xeon(R) CPU E5-2695 v4 @ 2.10GHz
stepping        : 1
microcode       : 0xb000038
cpu MHz         : 2101.000
cache size      : 46080 KB
physical id     : 0
siblings        : 18
core id         : 0
cpu cores       : 18
apicid          : 0
initial apicid  : 0
fpu             : yes
fpu_exception   : yes
cpuid level     : 20
wp              : yes
flags           : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge
 mca cmov pat pse36 clflush dts acpi mmx fxsr sse sse2 ss ht tm pbe sy
scall nx pdpe1gb rdtscp lm constant_tsc arch_perfmon pebs bts rep_good
 nopl xtopology nonstop_tsc aperfmperf eagerfpu pni pclmulqdq dtes64 m
onitor ds_cpl vmx smx est tm2 ssse3 sdbg fma cx16 xtpr pdcm pcid dca s
se4_1 sse4_2 x2apic movbe popcnt tsc_deadline_timer aes xsave avx f16c
 rdrand lahf_lm abm 3dnowprefetch epb cat_l3 cdp_l3 invpcid_single int
el_ppin intel_pt ssbd ibrs ibpb stibp tpr_shadow vnmi flexpriority ept
 vpid fsgsbase tsc_adjust bmi1 hle avx2 smep bmi2 erms invpcid rtm cqm
 rdt_a rdseed adx smap xsaveopt cqm_llc cqm_occup_llc cqm_mbm_total cq
m_mbm_local dtherm ida arat pln pts md_clear spec_ctrl intel_stibp flu
sh_l1d
bogomips        : 4190.37
clflush size    : 64
cache_alignment : 64
address sizes   : 46 bits physical, 48 bits virtual
power management:
```

what!?

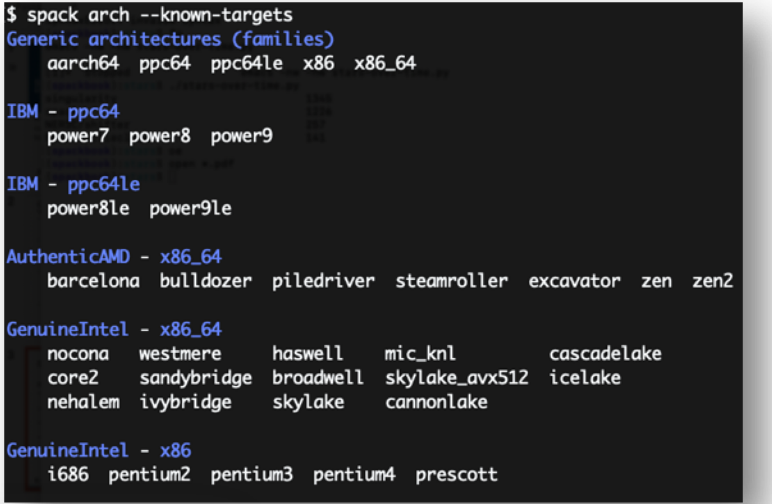
what!?

SPACK NOW UNDERSTANDS SPECIFIC TARGET MICROARCHITECTURES

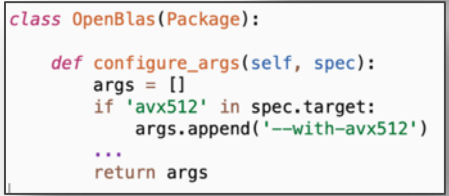
- Spack knows what type of machine you're on
  - Detects based on /proc/cpuinfo (Linux), sysctl (Mac)
  - Allows comparisons for compatibility, e.g.:

```
skylake > broadwell
zen2 > x86_64
```

- Key features:
  - Know which compilers support which chips with which flags
  - Determine compatibility
  - Enable creation and reuse of optimized binary packages
  - Easily query available architecture features for portable build recipes

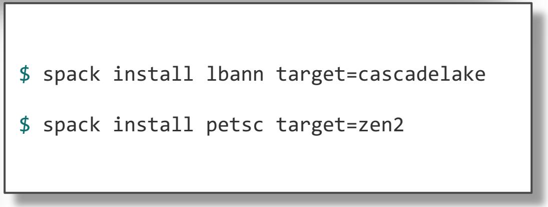


```
$ spack arch --known-targets
Generic architectures (families)
    aarch64  ppc64  ppc64le  x86  x86_64

IBM - ppc64
    power7  power8  power9

IBM - ppc64le
    power8le  power9le

AuthenticAMD - x86_64
    barcelona  bulldozer  piledriver  steamroller  excavator  zen  zen2

GenuineIntel - x86_64
    nocona   westmere     haswell    mic_knl          cascadelake
    core2    sandybridge  broadwell  skylake_avx512   icelake
    nehalem  ivybridge    skylake    cannonlake

GenuineIntel - x86
    i686  pentium2  pentium3  pentium4  prescott
```



```
class OpenBlas(Package):

    def configure_args(self, spec):
        args = []
        if 'avx512' in spec.target:
            args.append('--with-avx512')
        ...
        return args
```

Simple feature query



```
$ spack install lbann target=cascadelake

$ spack install petsc target=zen2
```

Specialized installations

ARCHSPEC: A LIBRARY FOR REASONING ABOUT MICROARCHITECTURES

- Standalone library, extracted from Spack
- Use fine-grained, human-readable labels, e.g.:
  - broadwell, haswell, skylake
  - instead of x86_64, aarch64, ppc64 etc.
- Query capabilities
  - “Does haswell support AVX-512?” “no.”
- Query compiler flags
  - “How do I compile for broadwell with icc?”
- Python package for now, but we want more bindings!
  - Actual data is in a common JSON file w/schema

arch
spec

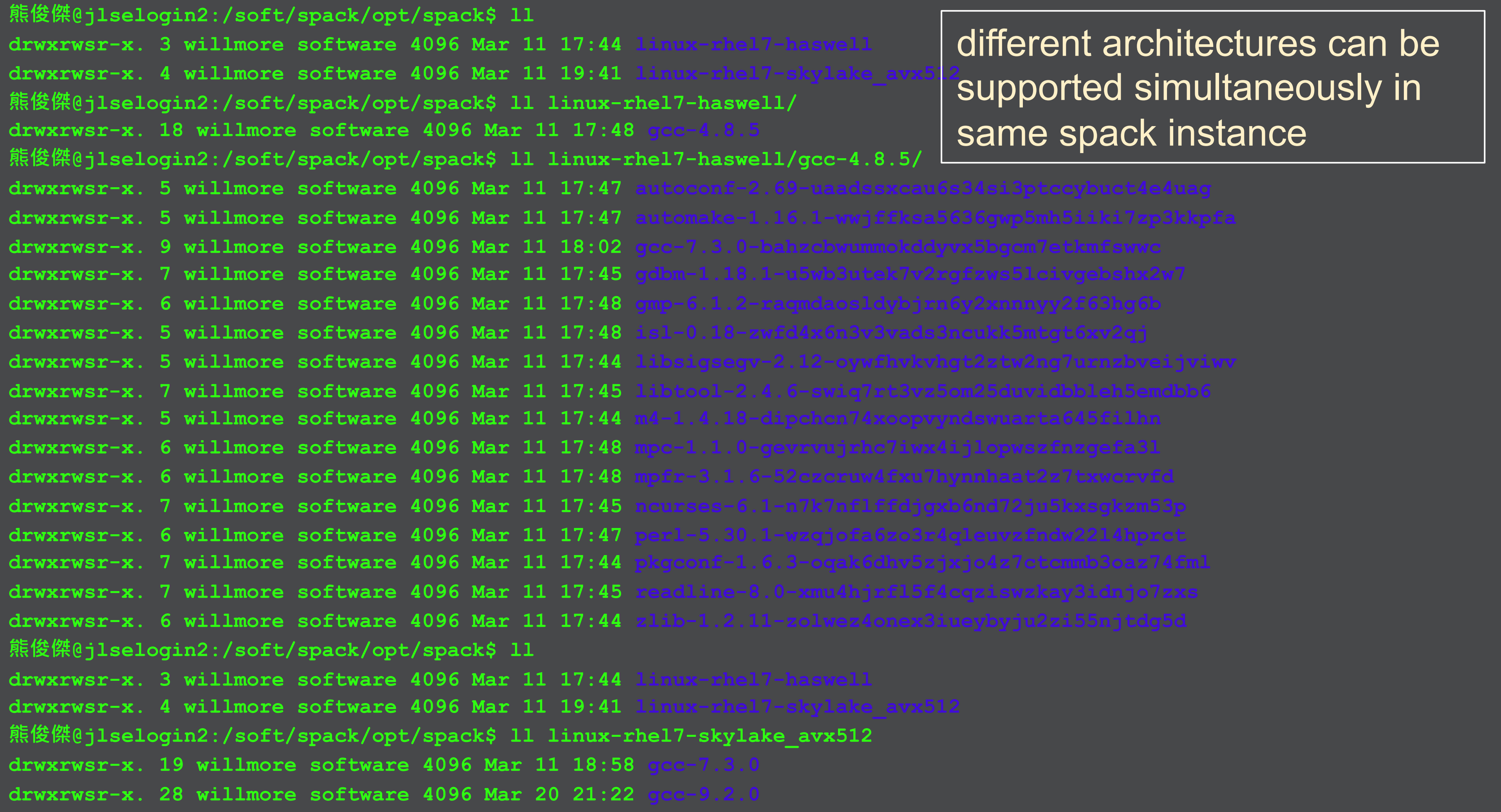
github.com/archspec

ReadTheDocs: archspec.rtfd.io

License: Apache 2.0 OR MIT

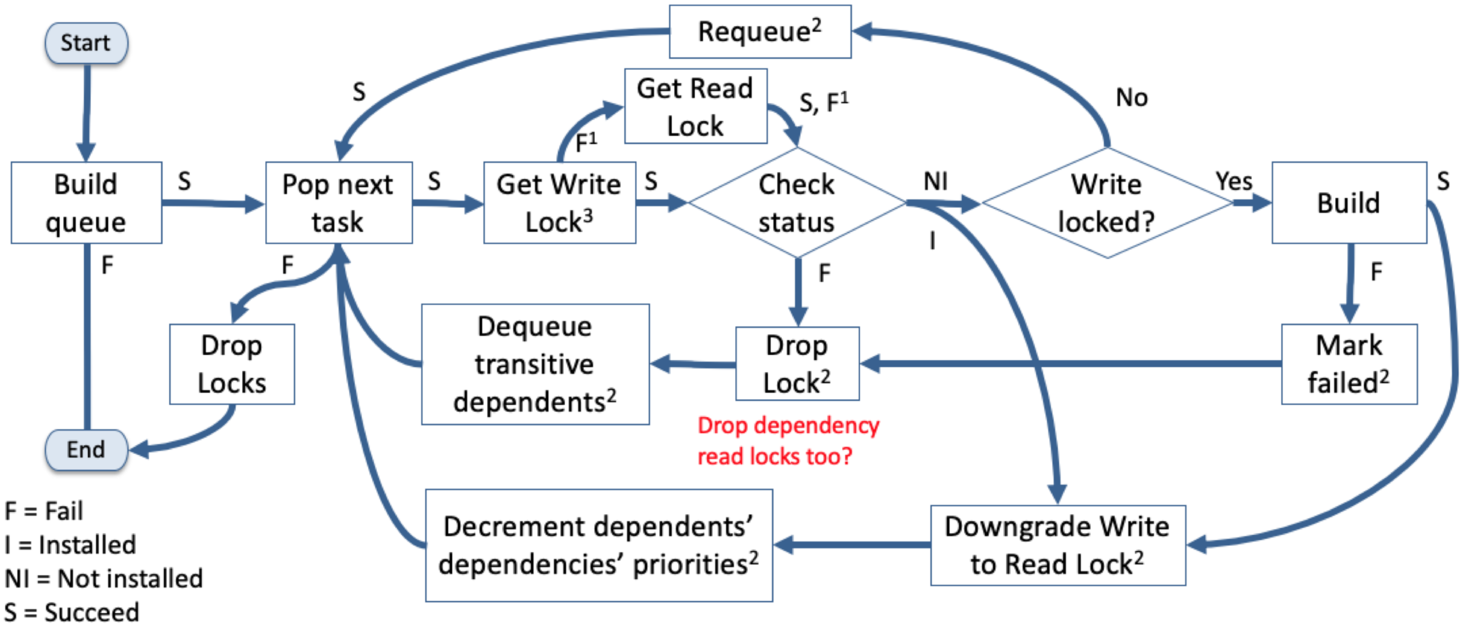
pip3 install archspec

different architectures can be supported simultaneously in same spack instance



```
熊俊傑@jlselogin2:/soft/spack/opt/spack$ ll
drwxrwsr-x. 3 willmore software 4096 Mar 11 17:44 linux-rhel7-haswell
drwxrwsr-x. 4 willmore software 4096 Mar 11 19:41 linux-rhel7-skylake_avx512
熊俊傑@jlselogin2:/soft/spack/opt/spack$ ll linux-rhel7-haswell/
drwxrwsr-x. 18 willmore software 4096 Mar 11 17:48 gcc-4.8.5
熊俊傑@jlselogin2:/soft/spack/opt/spack$ ll linux-rhel7-haswell/gcc-4.8.5/
drwxrwsr-x. 5 willmore software 4096 Mar 11 17:47 autoconf-2.69-uaadssxcau6s34si3ptccybuct4e4uag
drwxrwsr-x. 5 willmore software 4096 Mar 11 17:47 automake-1.16.1-wwjffksa5636gwp5mh5iiki7zp3kkpfa
drwxrwsr-x. 9 willmore software 4096 Mar 11 18:02 gcc-7.3.0-bahzcbwummokddyvx5bgcm7etkmfswwc
drwxrwsr-x. 7 willmore software 4096 Mar 11 17:45 gdbm-1.18.1-u5wb3utek7v2rgfzws5lcivgebshx2w7
drwxrwsr-x. 6 willmore software 4096 Mar 11 17:48 gmp-6.1.2-raqmdaosldybjrn6y2xnnnyy2f63hg6b
drwxrwsr-x. 5 willmore software 4096 Mar 11 17:48 isl-0.18-zwfd4x6n3v3vads3ncukk5mtgt6xv2qj
drwxrwsr-x. 5 willmore software 4096 Mar 11 17:44 libsigsegv-2.12-oywfhvkvhgt2ztw2ng7urnzbveijviwv
drwxrwsr-x. 7 willmore software 4096 Mar 11 17:45 libtool-2.4.6-swiq7rt3vz5om25duvidbbleh5emdbb6
drwxrwsr-x. 5 willmore software 4096 Mar 11 17:44 m4-1.4.18-dipchcn74xoopvyndswuarta645filhn
drwxrwsr-x. 6 willmore software 4096 Mar 11 17:48 mpc-1.1.0-gevrvujrhc7iwx4ijlopwszfnzgefa3l
drwxrwsr-x. 6 willmore software 4096 Mar 11 17:48 mpfr-3.1.6-52czcruw4fxu7hynnhaat2z7txwcrvfd
drwxrwsr-x. 7 willmore software 4096 Mar 11 17:45 ncurses-6.1-n7k7nflffdjgxb6nd72ju5kxsgkzm53p
drwxrwsr-x. 6 willmore software 4096 Mar 11 17:47 perl-5.30.1-wzqjofa6zo3r4qleuvzfndw22l4hprct
drwxrwsr-x. 7 willmore software 4096 Mar 11 17:44 pkgconf-1.6.3-oqak6dhv5zjxjo4z7ctcmmb3oaz74fml
drwxrwsr-x. 7 willmore software 4096 Mar 11 17:45 readline-8.0-xmu4hjrfl5f4cqziswzkay3idnjo7zxs
drwxrwsr-x. 6 willmore software 4096 Mar 11 17:44 zlib-1.2.11-zolwez4onex3iueybyju2zi55njtdg5d
熊俊傑@jlselogin2:/soft/spack/opt/spack$ ll
drwxrwsr-x. 3 willmore software 4096 Mar 11 17:44 linux-rhel7-haswell
drwxrwsr-x. 4 willmore software 4096 Mar 11 19:41 linux-rhel7-skylake_avx512
熊俊傑@jlselogin2:/soft/spack/opt/spack$ ll linux-rhel7-skylake_avx512
drwxrwsr-x. 19 willmore software 4096 Mar 11 18:58 gcc-7.3.0
drwxrwsr-x. 28 willmore software 4096 Mar 20 21:22 gcc-9.2.0
```

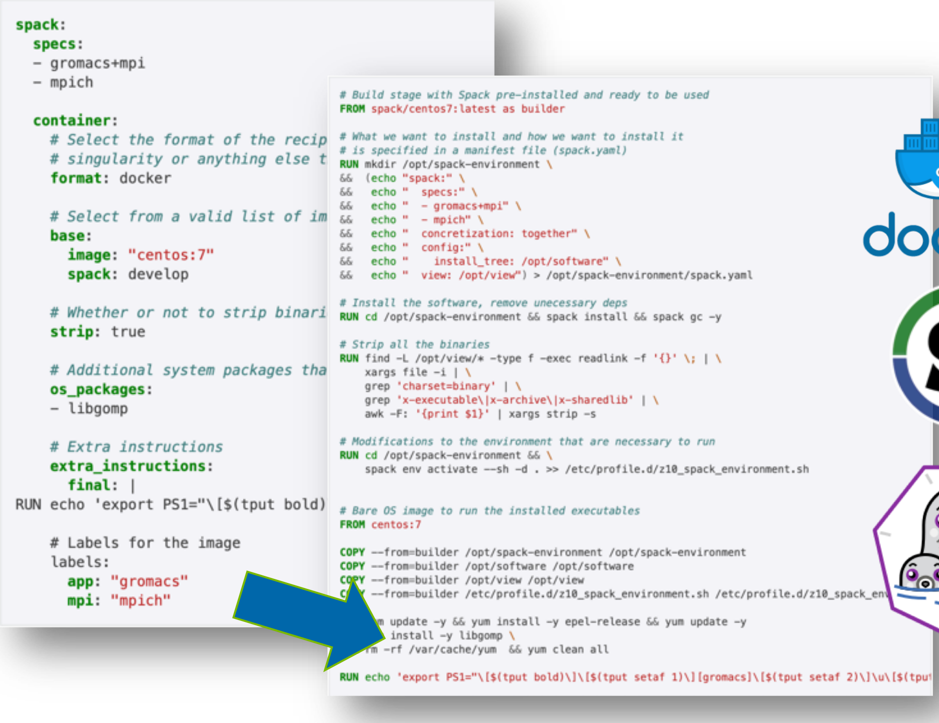
NEW DISTRIBUTED LOCKING ALGORITHM ENABLES (MIMD) PARALLEL BUILDS (0.14)



Drop dependency read locks too?

- **Spack instances can coordinate with each other using only filesystem locks (no MPI required)**
  - Independently run instances on login nodes, or
  - srun –N 8 –n 32 spack install -j 16 <package>

GENERATE CONTAINER IMAGES FROM ENVIRONMENTS (0.14)



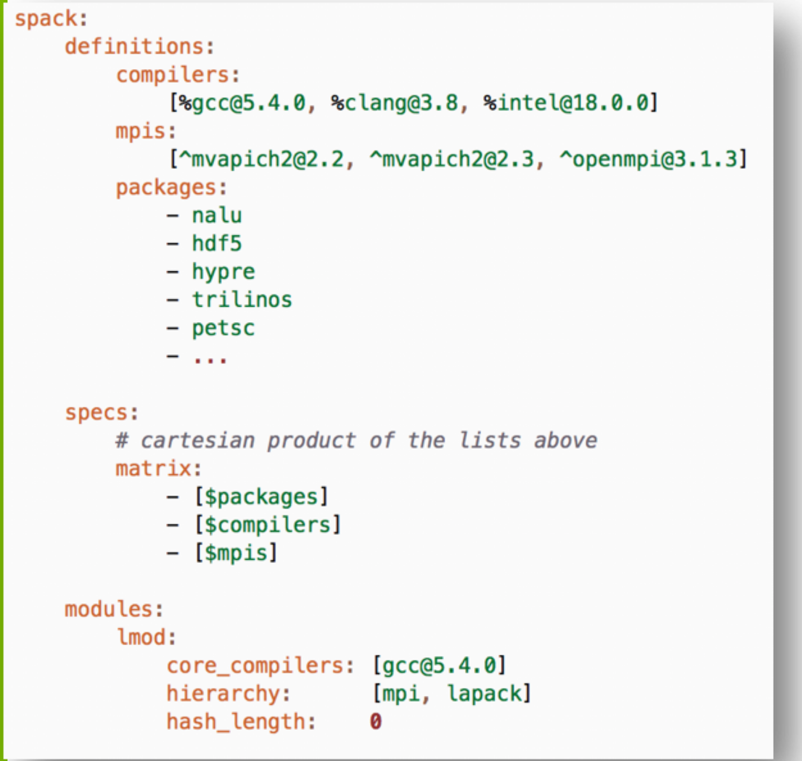
```
spack:
  specs:
  - gromacs+mpi
  - mpich

  container:
    # Select the format of the recip
    # singularity or anything else t
    format: docker

    # Select from a valid list of im
    base:
      image: "centos:7"
      spack: develop

    # Whether or not to strip binari
    strip: true

    # Additional system packages tha
    os_packages:
    - libgomp

    # Extra instructions
    extra_instructions:
      final: |
RUN echo 'export PS1="\[$(tput bold)

    # Labels for the image
    labels:
      app: "gromacs"
      mpi: "mpich"
```

```
# Build stage with Spack pre-installed and ready to be used
FROM spack/centos7:latest as builder

# What we want to install and how we want to install it
# is specified in a manifest file (spack.yaml)
RUN mkdir /opt/spack-environment \
&&  (echo "spack:" \
&&   echo "  specs:" \
&&   echo "  - gromacs+mpi" \
&&   echo "  - mpich" \
&&   echo "  concretization: together" \
&&   echo "  config:" \
&&   echo "    install_tree: /opt/software" \
&&   echo "  view: /opt/view") > /opt/spack-environment/spack.yaml

# Install the software, remove unecessary deps
RUN cd /opt/spack-environment && spack install && spack gc -y

# Strip all the binaries
RUN find -L /opt/view/* -type f -exec readlink -f '{}' \; | \
    xargs file -i | \
    grep 'charset=binary' | \
    grep 'x-executable\|x-archive\|x-sharedlib' | \
    awk -F: '{print $1}' | xargs strip -s

# Modifications to the environment that are necessary to run
RUN cd /opt/spack-environment && \
    spack env activate --sh -d . >> /etc/profile.d/z10_spack_environment.sh


# Bare OS image to run the installed executables
FROM centos:7

COPY --from=builder /opt/spack-environment /opt/spack-environment
COPY --from=builder /opt/software /opt/software
COPY --from=builder /opt/view /opt/view
C    --from=builder /etc/profile.d/z10_spack_environment.sh /etc/profile.d/z10_spack_env

    m update -y && yum install -y epel-release && yum update -y
      install -y libgomp \
    rm -rf /var/cache/yum  && yum clean all

RUN echo 'export PS1="\[$(tput bold)\]\[$(tput setaf 1)\][gromacs]\[$(tput setaf 2)\]\u\[$(tpu
```

spack containerize

- Any Spack environment can be bundled into a container image
  - Optional container section allows finer-grained customization
- Generated Dockerfile uses multi-stage builds to minimize size of final image
  - Strips binaries
  - Removes unneeded build deps with spack gc
- Can also generate Singularity recipes

SPACK STACKS: COMBINATORIAL ENVIRONMENTS FOR ENTIRE FACILITY DEPLOYMENTS



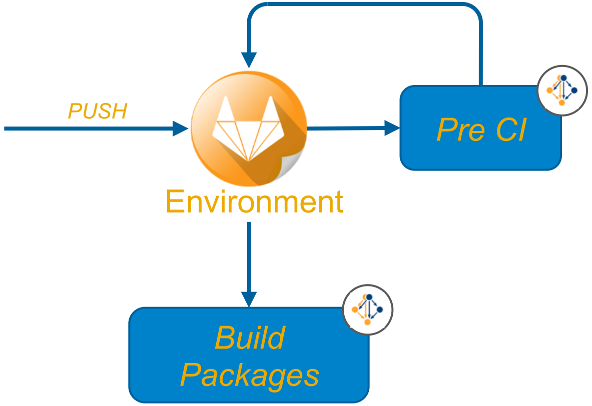
```
spack:
    definitions:
        compilers:
            [%gcc@5.4.0, %clang@3.8, %intel@18.0.0]
        mpis:
            [^mvapich2@2.2, ^mvapich2@2.3, ^openmpi@3.1.3]
        packages:
            - nalu
            - hdf5
            - hypre
            - trilinos
            - petsc
            - ...

    specs:
        # cartesian product of the lists above
        matrix:
            - [$packages]
            - [$compilers]
            - [$mpis]

    modules:
        lmod:
            core_compilers: [gcc@5.4.0]
            hierarchy:      [mpi, lapack]
            hash_length:    0
```

- Allow users to easily express a huge cartesian product of specs
  - All the packages needed for a facility
  - Generate modules tailored to the site
  - Generate a directory layout to browse the packages
- Build on the environments workflow
  - Manifest + lockfile
  - Lockfile enables reproducibility
- Relocatable binaries allow the same binary to be used in a stack, regular install, or container build.
  - Difference is how the user interacts with the stack
  - Single-PATH stack vs. modules.

WHAT'S NEW WITH SPACK PIPELINES

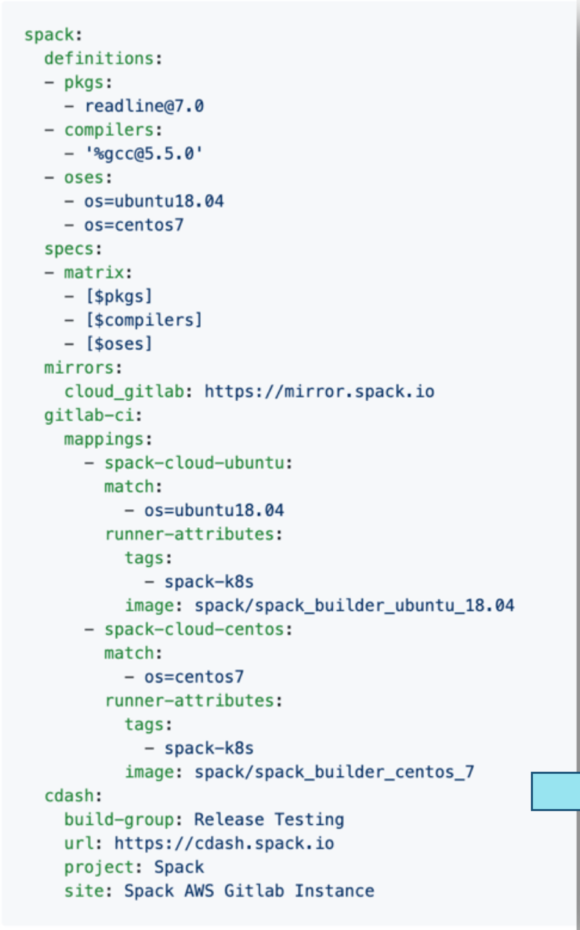
New workflow is centered around users' environment repository.

- Now:
  - Easy to control how Spack is cloned.
    - Clone a fork instead of Github
    - Clone a particular branch or ref
    - Or don't clone Spack at all. Can also just use a Spack that is preinstalled in your runners' environments.



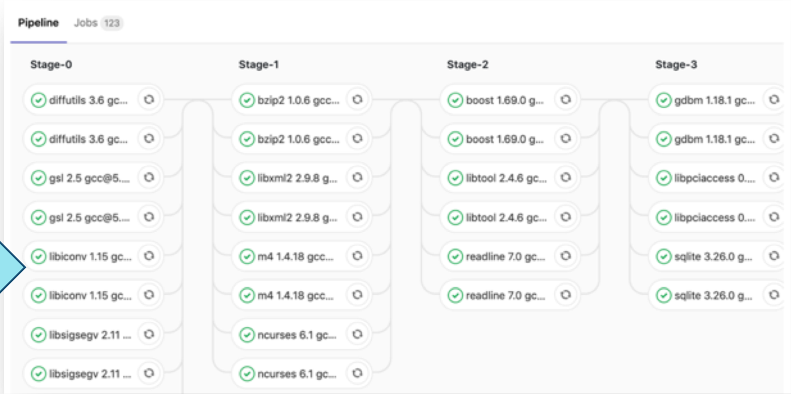
SPACK CAN NOW GENERATE CI PIPELINES FROM ENVIRONMENTS (ENHANCED IN 0.14)

- User adds a gitlab-ci section to environment
  - Spack maps builds to GitLab runners
  - Generate gitlab-ci.yml with spack ci command
- Can run in a container cluster or bare metal at an HPC site
  - Sends progress to CDash

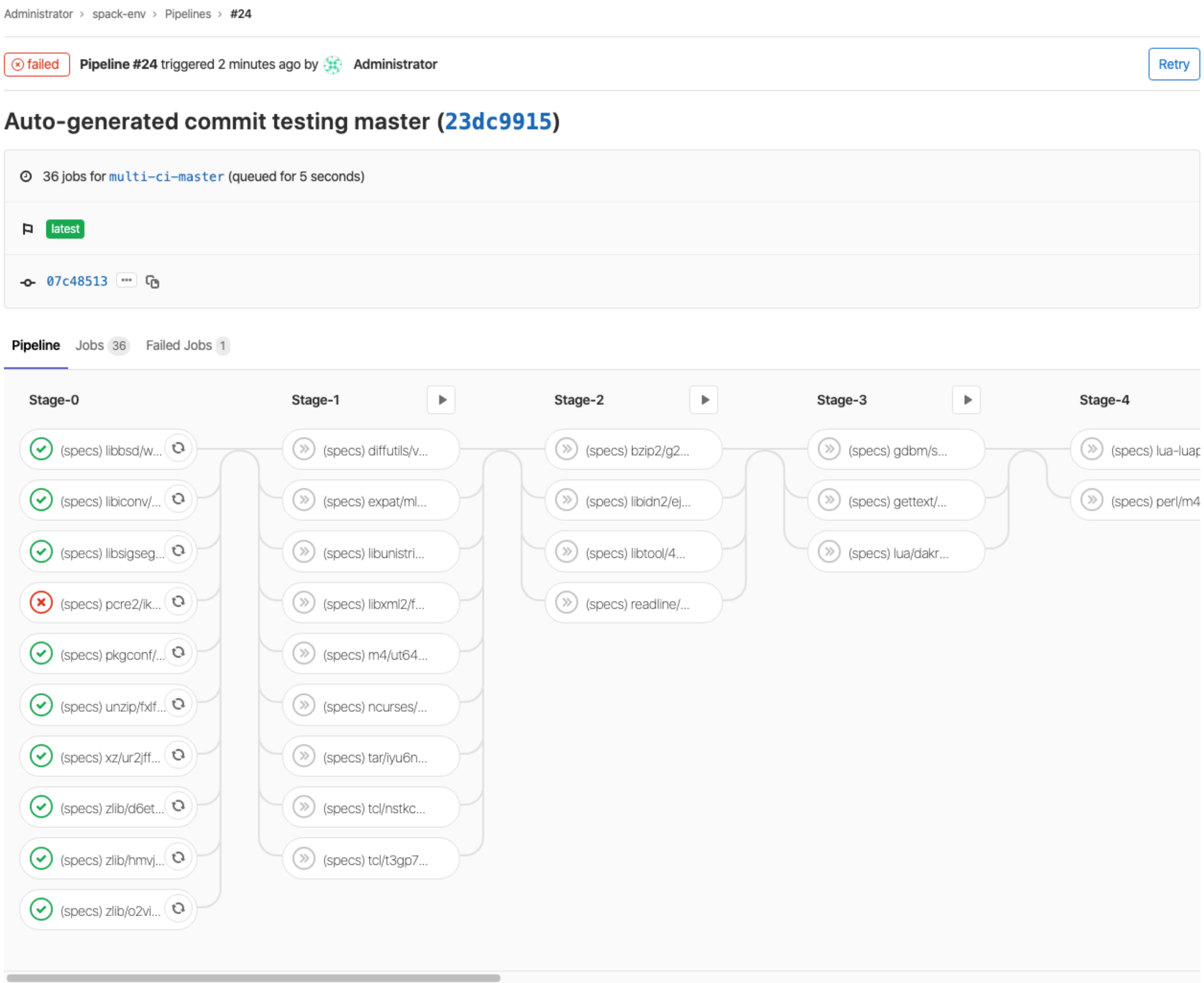


```
spack:
  definitions:
  - pkgs:
    - readline@7.0
  - compilers:
    - '%gcc@5.5.0'
  - oses:
    - os=ubuntu18.04
    - os=centos7
  specs:
  - matrix:
    - [$pkgs]
    - [$compilers]
    - [$oses]
  mirrors:
    cloud_gitlab: https://mirror.spack.io
  gitlab-ci:
    mappings:
      - spack-cloud-ubuntu:
        match:
          - os=ubuntu18.04
        runner-attributes:
          tags:
            - spack-k8s
          image: spack/spack_builder_ubuntu_18.04
      - spack-cloud-centos:
        match:
          - os=centos7
        runner-attributes:
          tags:
            - spack-k8s
          image: spack/spack_builder_centos_7
  cdash:
    build-group: Release Testing
    url: https://cdash.spack.io
    project: Spack
    site: Spack AWS Gitlab Instance
```

spack ci



Stage-0	Stage-1	Stage-2	Stage-3
✓ diffutils 3.6 gc...	✓ bzip2 1.0.6 gcc...	✓ boost 1.69.0 g...	✓ gdbm 1.18.1 gc...
✓ diffutils 3.6 gc...	✓ bzip2 1.0.6 gcc...	✓ boost 1.69.0 g...	✓ gdbm 1.18.1 gc...
✓ gsl 2.5 gcc@5....	✓ libxml2 2.9.8 g...	✓ libtool 2.4.6 gc...	✓ libpciaccess 0....
✓ gsl 2.5 gcc@5....	✓ libxml2 2.9.8 g...	✓ libtool 2.4.6 gc...	✓ libpciaccess 0....
✓ libiconv 1.15 gc...	✓ m4 1.4.18 gcc...	✓ readline 7.0 gc...	✓ sqlite 3.26.0 g...
✓ libiconv 1.15 gc...	✓ m4 1.4.18 gcc...	✓ readline 7.0 gc...	✓ sqlite 3.26.0 g...
✓ libsigsegv 2.11 ...	✓ ncurses 6.1 gc...		
✓ libsigsegv 2.11 ...	✓ ncurses 6.1 gc...		



Administrator › spack-env › Pipelines › #24

ⓧ failed **Pipeline #24** triggered 2 minutes ago by **Administrator**

Retry

Auto-generated commit testing master (23dc9915)

36 jobs for `multi-ci-master` (queued for 5 seconds)

latest

`07c48513`

Pipeline Jobs 36 Failed Jobs 1

Stage-0
- ✓ (specs) libbsd/w...
- ✓ (specs) libiconv/...
- ✓ (specs) libsigseg...
- ✗ (specs) pcre2/ik...
- ✓ (specs) pkgconf/...
- ✓ (specs) unzip/fxlf...
- ✓ (specs) xz/ur2jff...
- ✓ (specs) zlib/d6et...
- ✓ (specs) zlib/hmvj...
- ✓ (specs) zlib/o2vi...

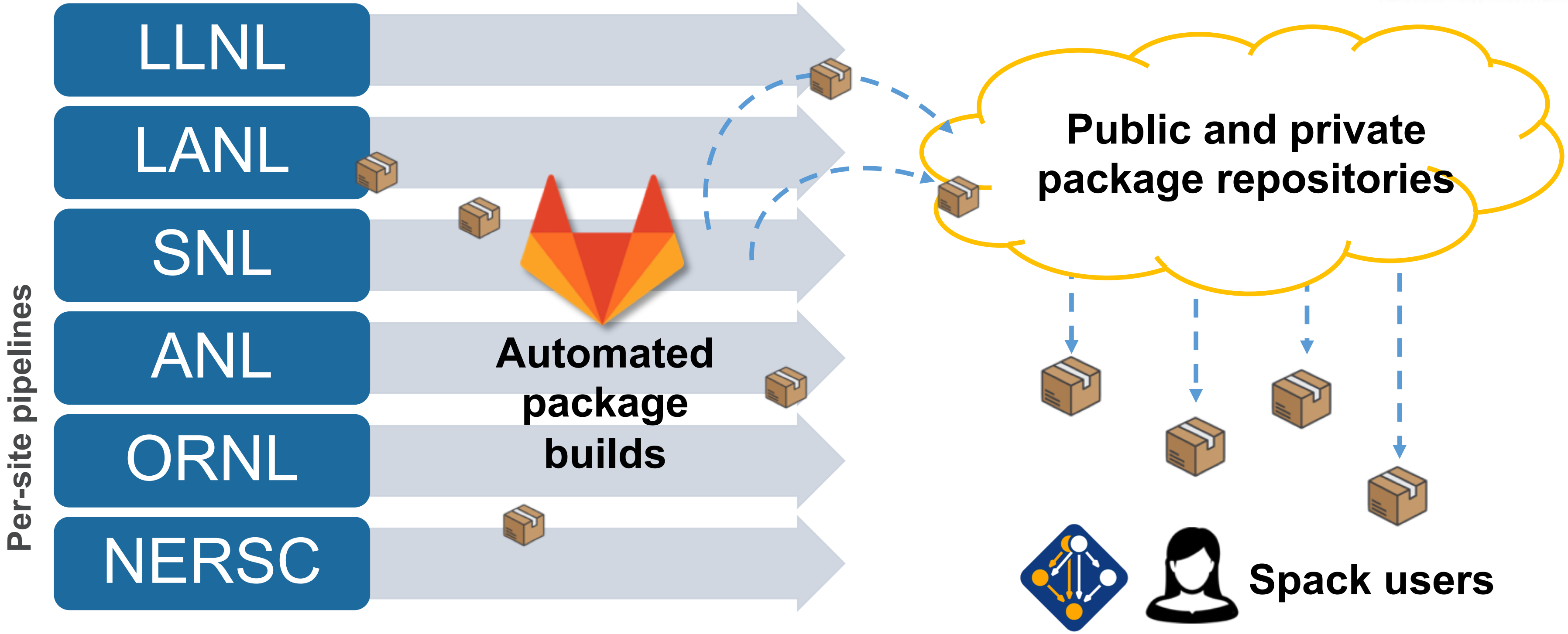
Stage-1 ▶
- (specs) diffutils/v...
- (specs) expat/ml...
- (specs) libunistri...
- (specs) libxml2/f...
- (specs) m4/ut64...
- (specs) ncurses/...
- (specs) tar/iyu6n...
- (specs) tcl/nstkc...
- (specs) tcl/t3gp7...

Stage-2 ▶
- (specs) bzip2/g2...
- (specs) libidn2/ej...
- (specs) libtool/4...
- (specs) readline/...

Stage-3 ▶
- (specs) gdbm/s...
- (specs) gettext/...
- (specs) lua/dakr...

Stage-4
- (specs) lua-luap
- (specs) perl/m4

AUTOMATED BUILDS USING GITLAB CI WILL ENABLE A ROBUST, WIDELY AVAILABLE HPC SOFTWARE ECOSYSTEM.



With pipeline efforts at E6 labs, users will no longer need to *build* their own software for high performance.

THE SPACK COMMUNITY

- **Spack simplifies HPC software for:**
  - Users
  - Developers
  - Cluster installations
  - The largest HPC facilities
- **Spack is central to ECP's software strategy**
  - Enable software reuse for developers and users
  - Allow the facilities to consume the entire ECP stack
- **Spack has a thriving open source community**
  - Contributing your package allows others to easily use and build on them
  - *Very* active Slack channel and GitHub site provide help
  - Spack community is very open to new features and ideas!

Visit:
spack.io

github.com/spack/spack

@spackpm

SPACK AT ALCF

CURRENT ALCF THETA SOFTWARE CONFIGURATION

- The ‘image’ - read-only filesystem on every compute node
- ‘/soft’ – LUSTRE mount containing may builds
- ‘/theta-archive’ - GPFS containing deprecated versions
- Module system – User interface to manipulate software environment
- Python – software ecosystem managed via conda or similar
- Spack – (LUSTRE) increasingly relevant deployment tool

CURRENT ECP SPACK EFFORTS AT ALCF (Q3/4 FY2020)

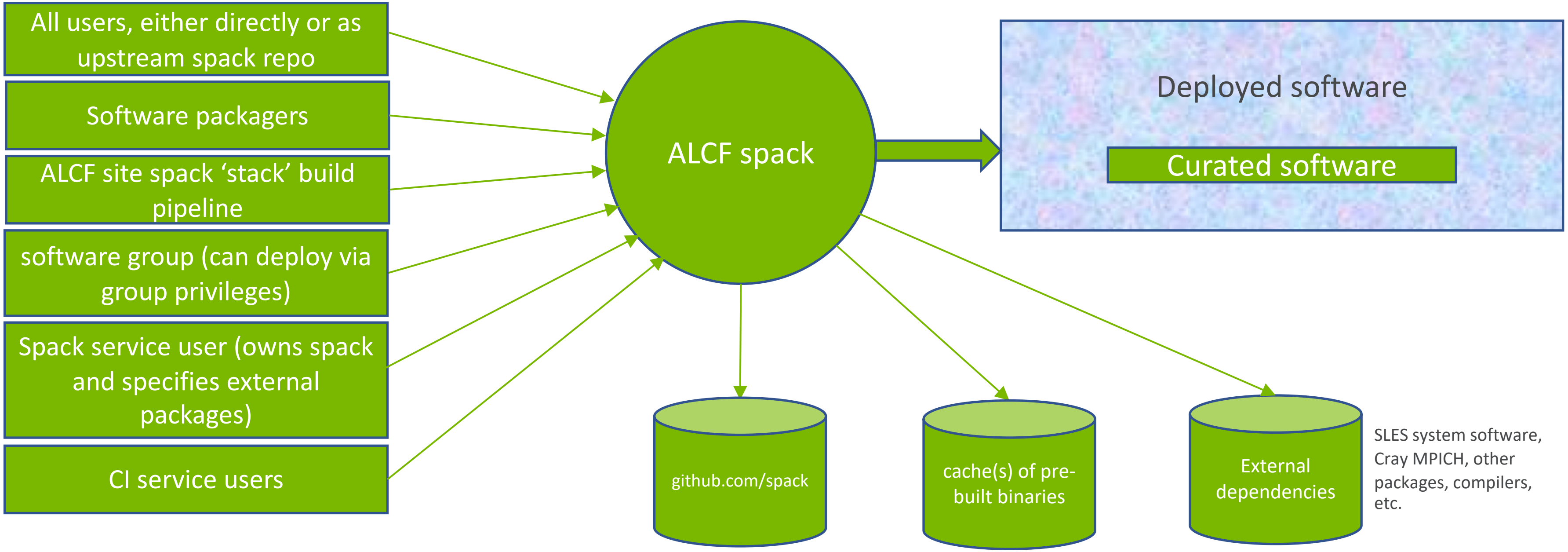
- Spack pipelines
- Vendor Stack Integration
- Onboarding of CI
- Container Integration
- Ongoing deployment efforts

ALCF MULTI-USER SPACK VISION:

- Built on top of Linux user/group/owner permission model
- Owned by ‘spack’ service user
- Allows for incorporation of optional upstream spack installations and repositories of built packages
- Curation managed *via* symbolic link according to local policy
- Curated versions available *via* environment modules
- tests

Those using spack to build:

Those using spack-built software:



All users, either directly or as upstream spack repo

Software packagers

ALCF site spack ‘stack’ build pipeline

software group (can deploy via group privileges)

Spack service user (owns spack and specifies external packages)

CI service users

ALCF spack

Deployed software

Curated software

github.com/spack

cache(s) of pre-built binaries

External dependencies

SLES system software, Cray MPICH, other packages, compilers, etc.

Local spack instance is updated from official versioned releases of spack

Could be federated or local

MULTI-USER SPACK: CHALLENGES

- Spack learning curve and inherent complexity
- Incompatible builds (not every combination works and package.py may not be aware of bad combination)
- Packages whose build systems do not support cross-compilation
- Missing and/or out-of-date package.py files (spack's chicken and egg problem)
- Questionable choices by concretizer based on what's already built/available
- Some builds require disruptive updates to external/system packages (e.g. openssl)
  - These are often low-level dependencies which affect the entire DAG
- Conversely, vendor updates require re-working the settings for external dependencies
  - Philosophical differences between vendors, sysadmins, packagers, other stakeholders on how software should be deployed
  - Updates to vendor-provided packages may break compatibility of a previously working build
- Conflicting/confusing settings (six levels of configuration scope plus stacks, environments, etc…)
- Dependency on an unreleased software version (xSDK depends on non-existent adol-c@2.64)
  - <soapbox> version number is a control variable </soapbox>
- Bugs which manifest in multi-user version, e.g. overwriting internal settings with different ownership and without group write permissions.
- Other Minor issues
  - Non-ECP sponsored dependencies
  - Non-semantic or unusual versioning
- **Users/packagers ended up cloning their own version of spack anyway…**

CONCLUSIONS