Spack

A Package Manager for Exascale

ECP All Hands Meeting
Knoxville, TN
February 1, 2017

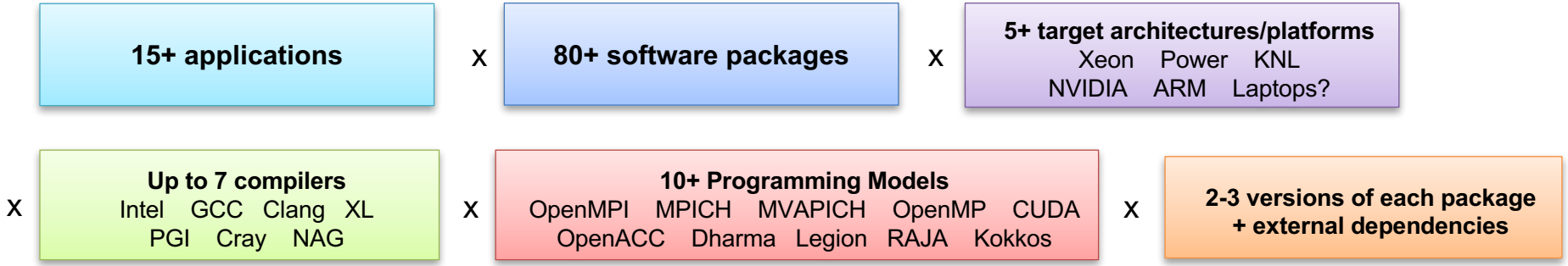
Todd Gamblin
Center for Applied Scientific Computing
Lawrence Livermore National Laboratory

LLNL-PRES-806064

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract DE-AC52-07NA27344.
Lawrence Livermore National Security, LLC

https://spack.io

The complexity of the exascale ecosystem threatens productivity.



= up to **1,260,000** combinations!

- Developers, users, and facilities dedicate (many) FTEs to building & porting, but…
- Programming models and libraries are critical for performance portability.

We must make it easier to rely on others' software!

The exascale software stack is our deliverable, but how will we deliver it?

- Stack needs to run on:
  - Testbeds
  - Commodity clusters
  - Laptops
  - Exascale facility platforms

- Each new environment requires an FTE, or a significant fraction of someone's time.

- We are tasked with building a software stack that will have an impact on industry.
  - Needs to be robust, tested, and reliable
  - Needs to be easy to get up and running

To meet these goals, we need more automation.

Ok, so how can we automate things?

Spack is a package manager for exascale

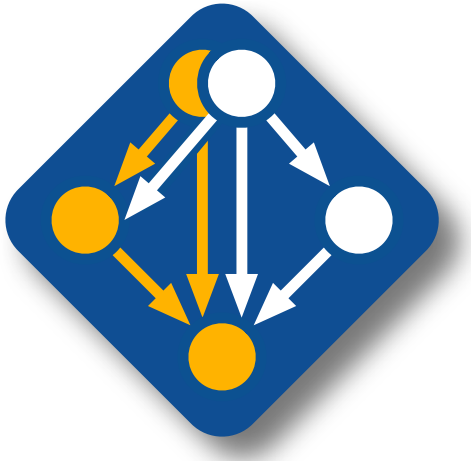
1. Install Spack:

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

2. Install a package:

```
spack install hdf5
```

3. HDF5 and all of its dependencies are:
   - Downloaded
   - Built
   - Installed within the spack directory.

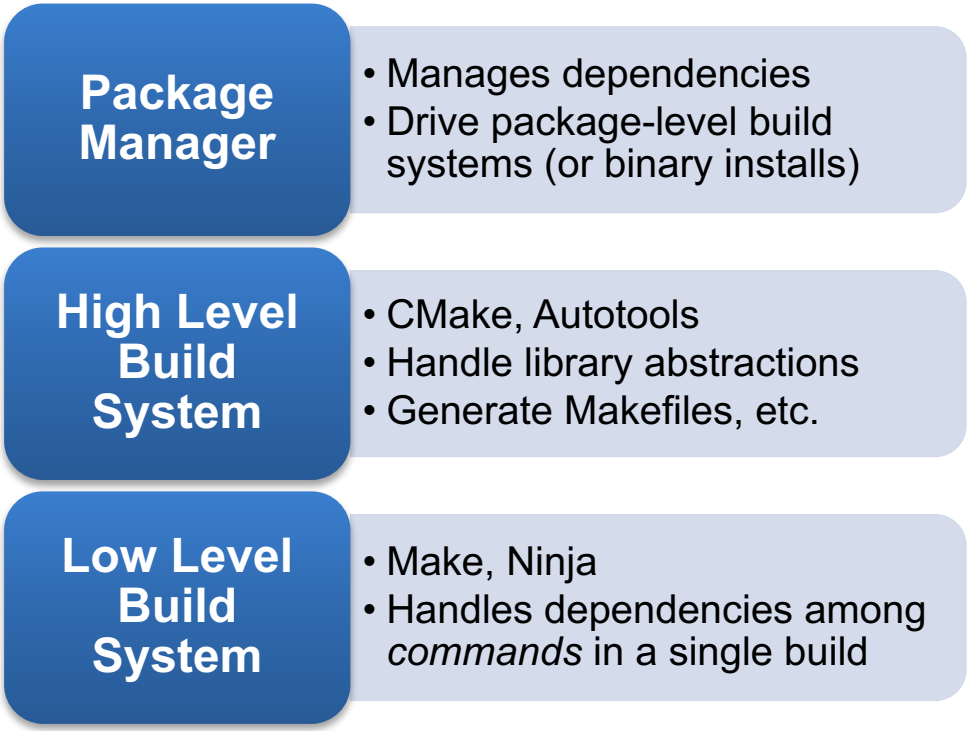


github.com/LLNL/spack

Get Spack!

What's a package manager?

- Spack is a ***package manager***
  - **Does not replace** CMake/Autotools/Makefile
  - Packages built by Spack can have any build system they want.
- Spack automates builds and ***manages dependencies***
  - Drives package-level build systems
  - Ensures consistent builds
- Figuring out magic configure line takes time
  - Spack is a cache of recipes
  - Allows effort to be leveraged across teams



Who can use Spack?

1. **End Users of HPC Software**
   - Install and run HPC applications and tools in home directory
   - No privileged access required

2. **HPC Application Teams**
   - Manage third-party dependency libraries
   - Package applications for others to use

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

Why use Spack?

1. Automate the process of building complex software packages
2. Automate testing with different compilers and options
3. Leverage the work of others with shared build recipes
4. Distribute your software to a growing audience of HPC users

Ok, so how do I use this thing?

`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 1,100 packages now.

`spack find` shows what is installed

```
$ spack find
==> 103 installed packages.
-- linux-redhat6-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-redhat6-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-redhat6-x86_64 / intel@14.0.2 ---------------------------
hwloc@1.9  mpich@3.0.4  starpu@1.1.4

-- linux-redhat6-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-redhat6-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
```

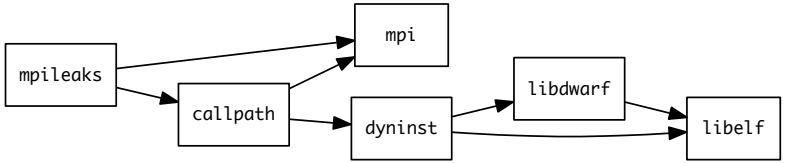
- All the versions coexist!
  - Multiple versions of same package are ok.
- Packages are installed to automatically find correct dependencies.
- Binaries work *regardless of user's environment*.
- Spack also generates module files.
  - Use to deploy at facilities.

Most existing tools do not support combinatorial versioning

- Traditional binary package managers
  - RPM, yum, APT, yast, etc.
  - Designed to manage a single stack.
  - Install *one* version of each package in a single prefix (`/usr`).
  - Seamless upgrades to a *stable, well tested* stack
- Port systems
  - BSD Ports, portage, Macports, Homebrew, Gentoo, etc.
  - Minimal support for builds parameterized by compilers, dependency versions.
- Virtual Machines and Linux Containers (Docker)
  - Containers allow users to build environments for different applications.
  - Someone has to build the image
  - Package managers accelerate the process of creating images

Spack handles combinatorial software complexity.

Dependency DAG



Installation Layout

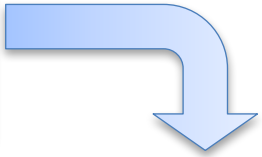
Hash

```
spack/opt/
    linux-x86_64/
        gcc-4.7.2/
            mpileaks-1.1-0f54bf34cadk/
        intel-14.1/
            hdf5-1.8.15-lkf14aq3nqiz/
    bgq/
        xl-12.1/d
            hdf5-1-8.16-fqb3a15abrwx/
    ...
```

- Each unique dependency graph is a unique ***configuration***.
- Each configuration installed in a unique directory.
  - 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
  - Things work *the way you built them*

Users can query the full dependency configuration of installed packages.

```
$ spack find callpath
==> 2 installed packages.
-- linux-x86_64 / clang@3.4 --------     -- linux-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-x86_64 / clang@3.4 -----------     -- linux-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.

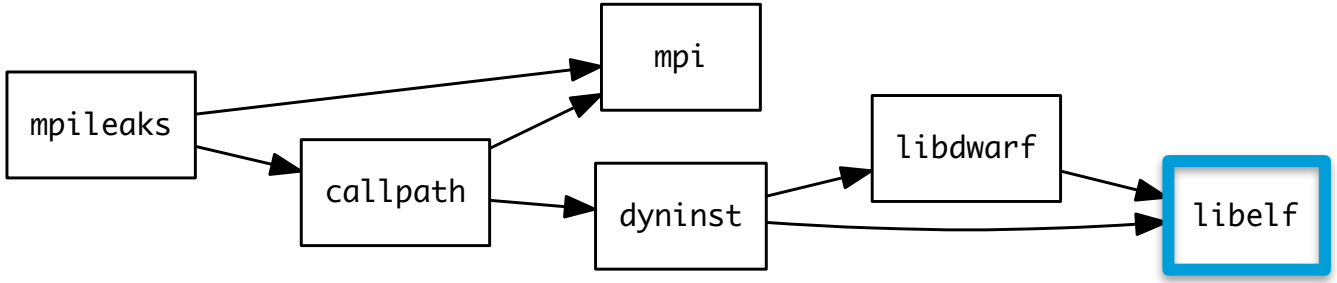
Spack provides a *spec* syntax to describe customized DAG configurations

```
$ 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'       setting compiler flags
$ spack install mpileaks@3.3 os=CNL10 target=haswell  setting target for cross-compile
$ spack install mpileaks@3.3 ^mpich@3.2 %gcc@4.9.3    ^ constraints on dependencies
```

- 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

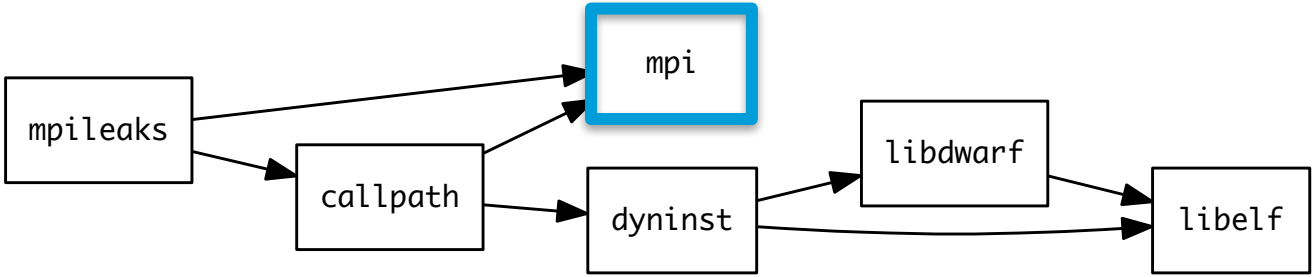
Specs can constrain versions of dependencies




```
$ spack install mpileaks %intel@12.1 ^libelf@0.8.12
```

- Spack ensures *one* configuration of each library per DAG
  - Ensures ABI consistency.
  - User does not need to know DAG structure; only the dependency *names.*
- Spack can ensure that builds use the same compiler, or you can mix
  - Working on ensuring ABI compatibility when compilers are mixed.

Spack handles ABI-incompatible, versioned interfaces like MPI



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

```
$ spack install mpileaks ^mvapich@1.9
```

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

- Let Spack choose MPI implementation, as long as it provides MPI 2 interface:

```
$ spack install mpileaks ^mpi@2
```

Spack packages are simple Python scripts

```
from spack import *

class Dyninst(Package):
    """API for dynamic binary instrumentation."""

    homepage = "https://paradyn.org"

    version('8.2.1', 'abf60b7faabe7a2e', url="http://www.paradyn.org/release8.2/DyninstAPI-8.2.1.tgz")
    version('8.1.2', 'bf03b33375afa66f', url="http://www.paradyn.org/release8.1.2/DyninstAPI-8.1.2.tgz")
    version('8.1.1', 'd1a04e995b7aa709', url="http://www.paradyn.org/release8.1/DyninstAPI-8.1.1.tgz")

    depends_on("libelf")
    depends_on("libdwarf")
    depends_on("boost@1.42:")

    def install(self, spec, prefix):
        libelf = spec['libelf'].prefix
        libdwarf = spec['libdwarf'].prefix

        with working_dir('spack-build', create=True):
            cmake('..',
                  '-DBoost_INCLUDE_DIR=%s' % spec['boost'].prefix.include,
                  '-DBoost_LIBRARY_DIR=%s' % spec['boost'].prefix.lib,
                  '-DBoost_NO_SYSTEM_PATHS=TRUE'
                  *std_cmake_args)
            make()
            make("install")

    @when('@:8.1')
    def install(self, spec, prefix):
        configure("--prefix=" + prefix)
        make()
        make("install")
```

Metadata

Versions and URLs

Dependencies

Patches, variants (not shown)

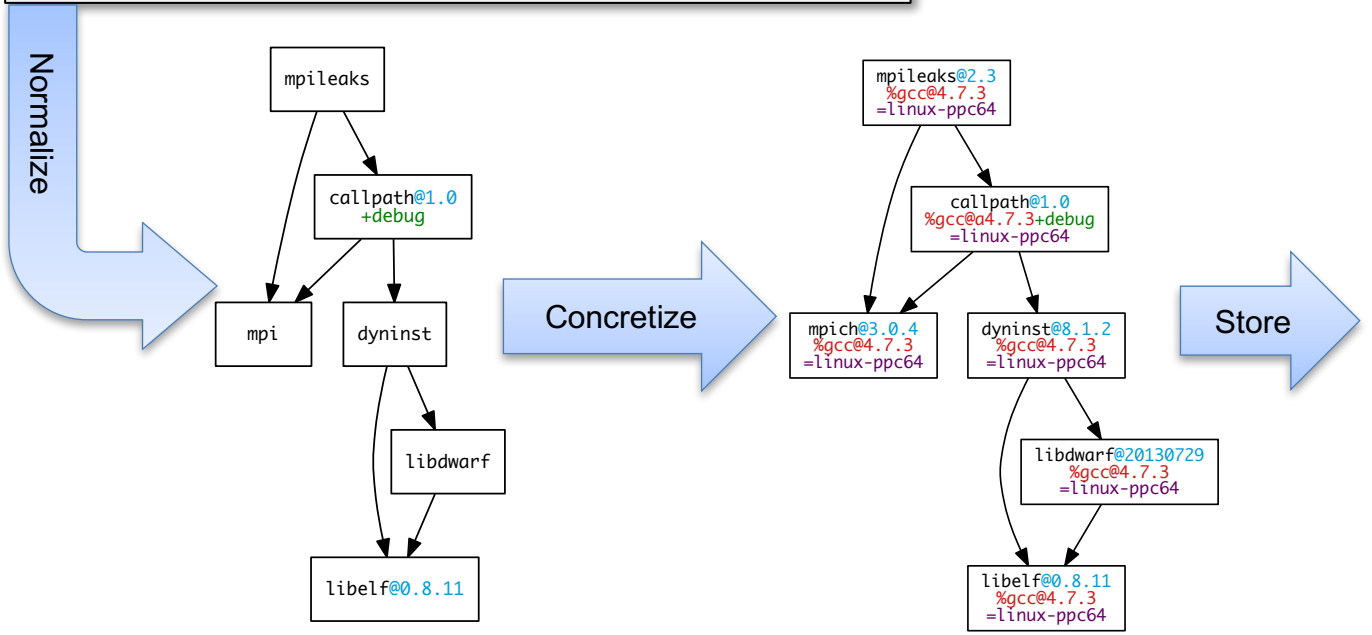
Commands for installation

Access build config through the *spec* parameter.

Concretization fills in missing configuration details when the user is not explicit.

```
mpileaks ^callpath@1.0+debug ^libelf@0.8.11
```

User input: *abstract* spec with some constraints



Abstract, normalized spec with some dependencies.

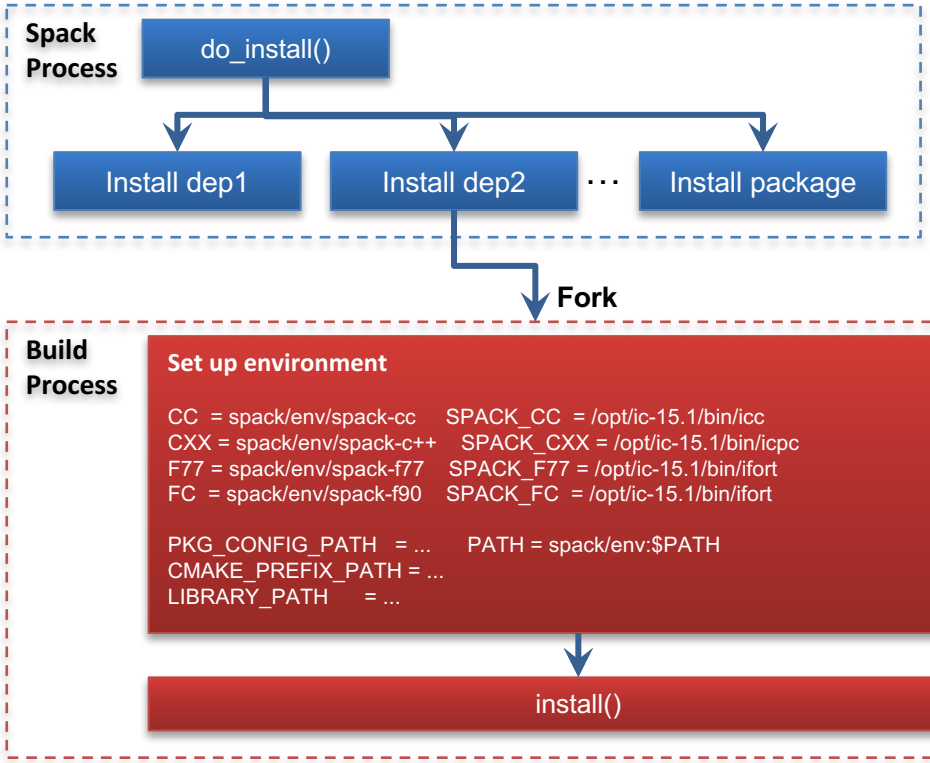
Concrete spec is fully constrained and can be passed to install.

spec.yaml

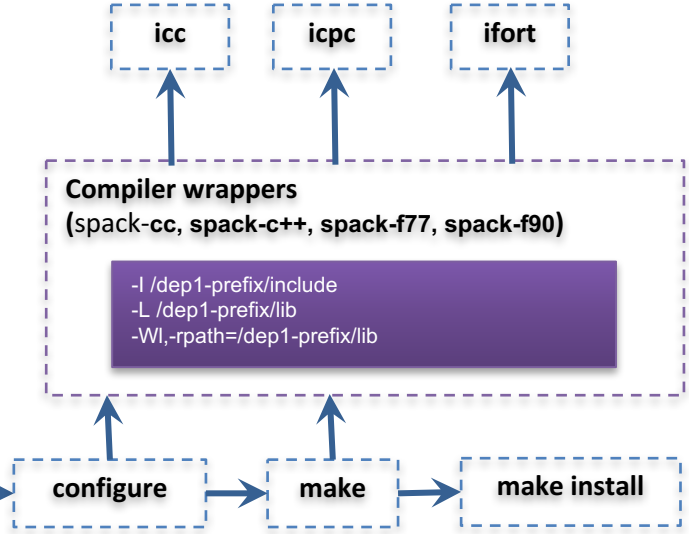
```
spec:
- mpileaks:
    arch: linux-x86_64
    compiler:
      name: gcc
      version: 4.9.2
    dependencies:
      adept-utils: kszrtkpbzac3ss2ixcjkcorlaybnptp4
      callpath: bah5f4h4d2n47mgycej2mtrnrivvxy77
      mpich: aa4ar6ifj23yijqmdabeakpejcli72t3
    hash: 33hjjhxi7p6gyzn5ptgyes7sghyprujh
    variants: {}
    version: '1.0'
- adept-utils:
    arch: linux-x86_64
    compiler:
      name: gcc
      version: 4.9.2
    dependencies:
      boost: teesjv7ehpe5ksspjim5dk43a7qnowlq
      mpich: aa4ar6ifj23yijqmdabeakpejcli72t3
    hash: kszrtkpbzac3ss2ixcjkcorlaybnptp4
    variants: {}
    version: 1.0.1
- boost:
    arch: linux-x86_64
    compiler:
      name: gcc
      version: 4.9.2
    dependencies: {}
    hash: teesjv7ehpe5ksspjim5dk43a7qnowlq
    variants: {}
    version: 1.59.0
...
```

Detailed provenance is stored with the installed package

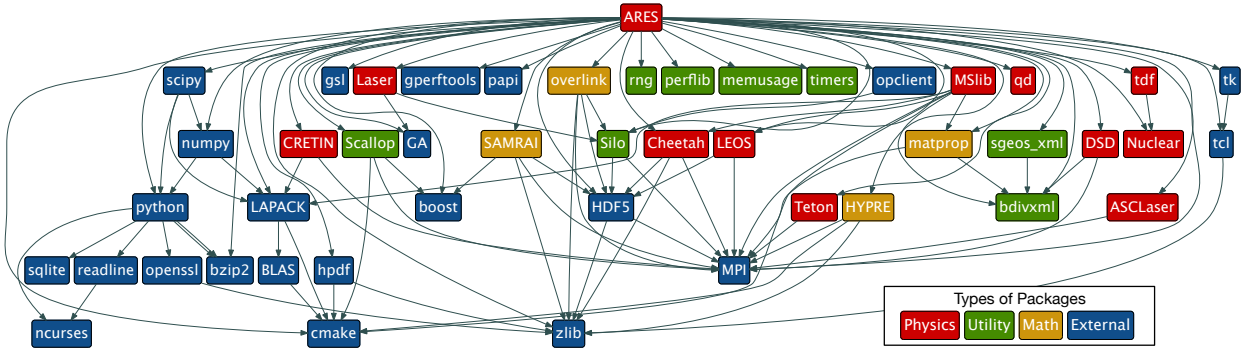
Spack builds each package in its own compilation environment



- **Forked build process isolates environment for each build.**
  Uses compiler wrappers to:
  - Add include, lib, and RPATH flags
  - Ensure that dependencies are found automatically
  - Load Cray modules (use right compiler/system deps)



Spack automates the build of LLNL multi-physics codes



	Linux			*BG/Q*	*Cray XE6*
	MVAPICH	*MVAPICH2*	*OpenMPI*	*BG/Q MPI*	*Cray MPI*
GCC	C P L D			C P L D	
Intel 14	C P L D				
Intel 15	C P L D	D			
PGI		D	C P L D		C L D
Clang	C P L D			C L D	
XL				C P L D	

- ARES is a 1, 2, and 3-D radiation hydrodynamics code
- The ARES configuration shown here has 47 dependencies
- ARES team runs nightly builds for 36 different configurations
  4 code versions:
  - **(C)**urrent Production
  - **(P)**revious Production
  - **(L)**ite
  - **(D)**evelopment

Spack enabled testing on 3 platforms, 6 compilers, and 3 MPI implementations.

Projects like Spack are not sustainable without a community

- HPC centers and developers are working on the same software
- Spack leverages network effects
- We have been deliberately open in order to build a community to sustain Spack.

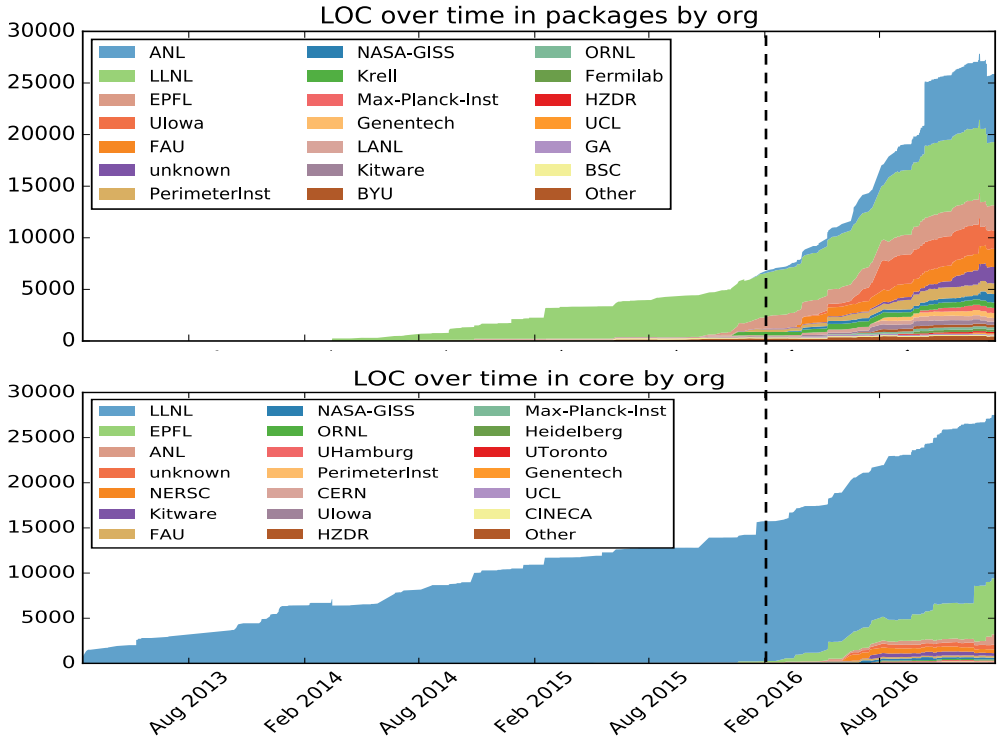
Individual

Team

Community

COMPLEXITY

Contributions to Spack have grown rapidly over the past year



- Just over 1 year ago, LLNL provided most of the contributions to Spack
- Since last year, we've gone from 300 to over 1,100 packages
- Over 75% of lines of code in packages are now from external contributors.
  - Contributors add to the core as well.
- We are committed to sustaining Spack's open source ecosystem!

The Spack community now spans DOE and beyond

- **30+ organizations**
  110+ contributors
  Sharing **1,145 packages** and growing

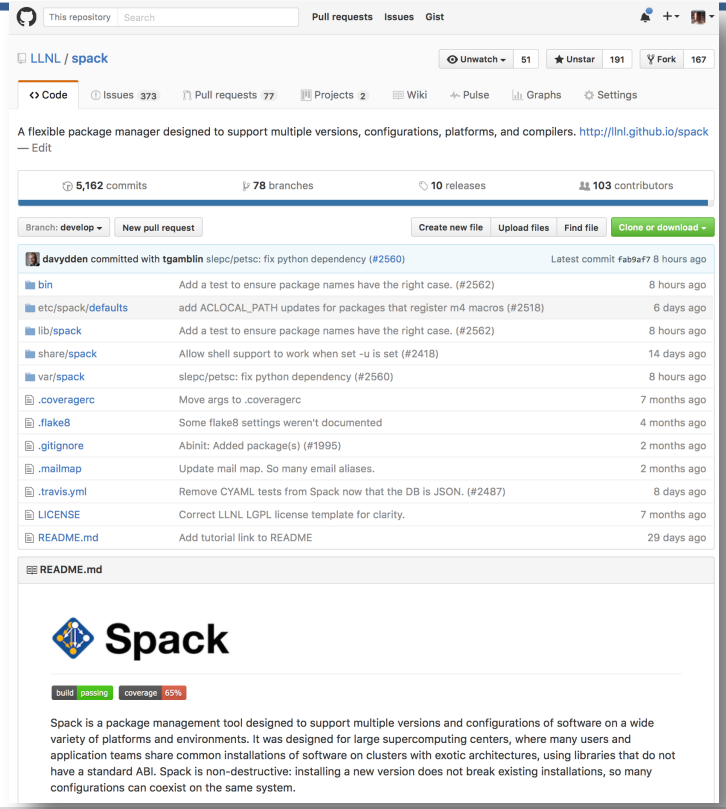
- **Spack can be a central repository for tools**
  - Make it easy for others to use them!
  - Build code the same way across HPC centers.

- **Spack is used in production at LLNL**
  - Livermore Computing
  - LLNL production multi-physics codes

- **Other organizations using Spack:**
  - **NERSC** using Spack on Cori: Cray support.
  - **EPFL** (Switzerland) contributing core features.
  - **Fermi, CERN, BNL**: high energy physics ecosystem.
  - **ANL** using Spack on production Linux clusters.
  - **NASA** packaging an Ice model code.
  - **ORNL** working with us on Spack for CORAL.
  - **Kitware**: core features, ParaView, UV-CDAT support

We use GitHub to manage large numbers of contributors

- Without git & github, we couldn't manage large numbers of contributions
- Git handles concurrent development easily with forks and branches
- Many people are familiar with how to contribute to github projects
  - Pull requests are well understood & accessible

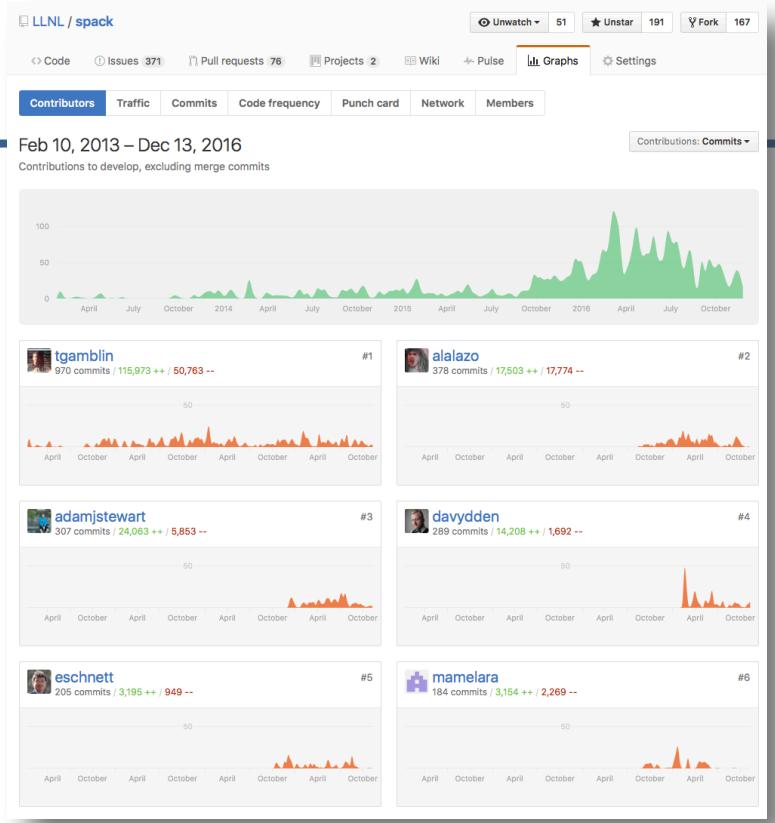


github.com/LLNL/spack

Community Support

- Spack has organically developed a core community of people who:
  - Review pull requests
  - Participate in discussions
  - Suggest new features
  - Implement new features

- In addition to all that, they also argue with each other
  - Managing open source projects is about transparency and managing expectations
  - Make sure the goals of your project are clear and people know how to contribute

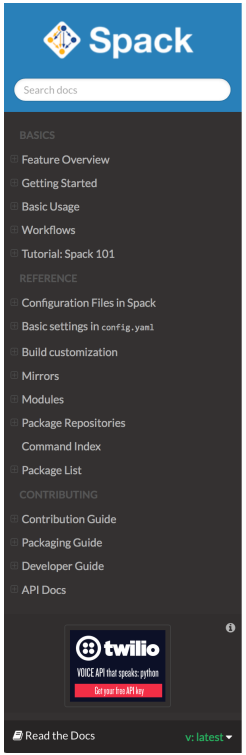


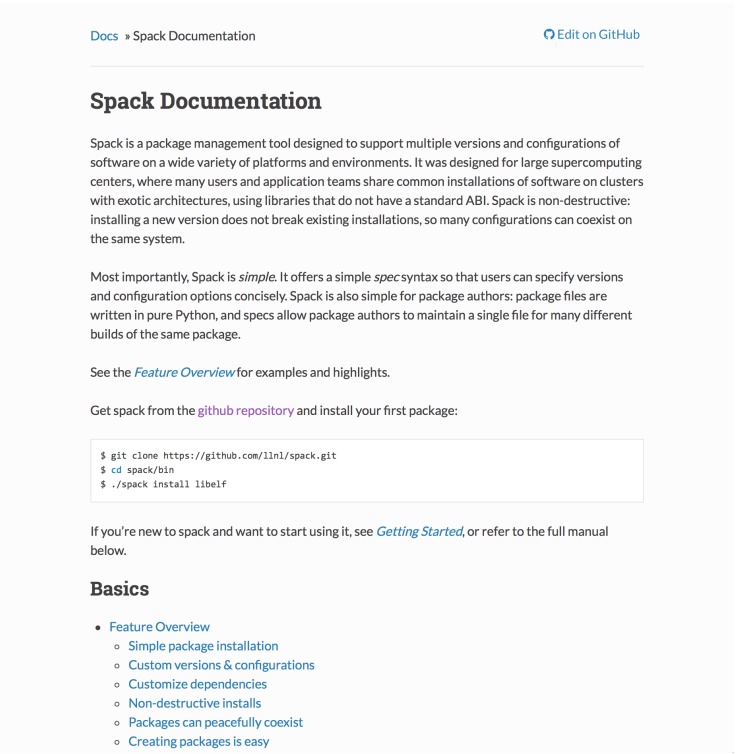
github.com/LLNL/spack/graphs/contributors

Readthedocs has been very helpful for recruiting users

- Users are more inclined to use a well documented project
- Easy to generate custom docs using Sphinx
- Readthedocs keeps docs in sync with github repository
  - Also provides older doc versions

https://spack.readthedocs.io

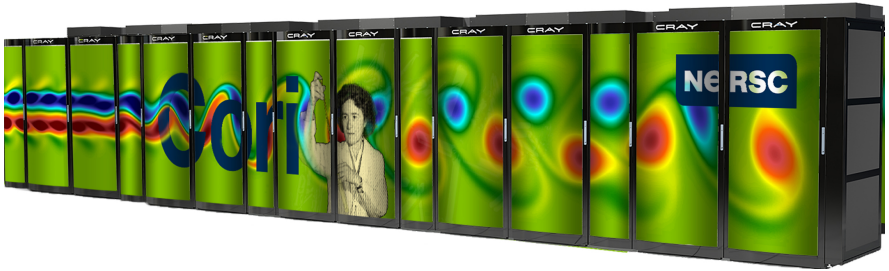




How can Spack integrate with ECP?

We are actively working with facilities to use Spack for site-wide software deployment

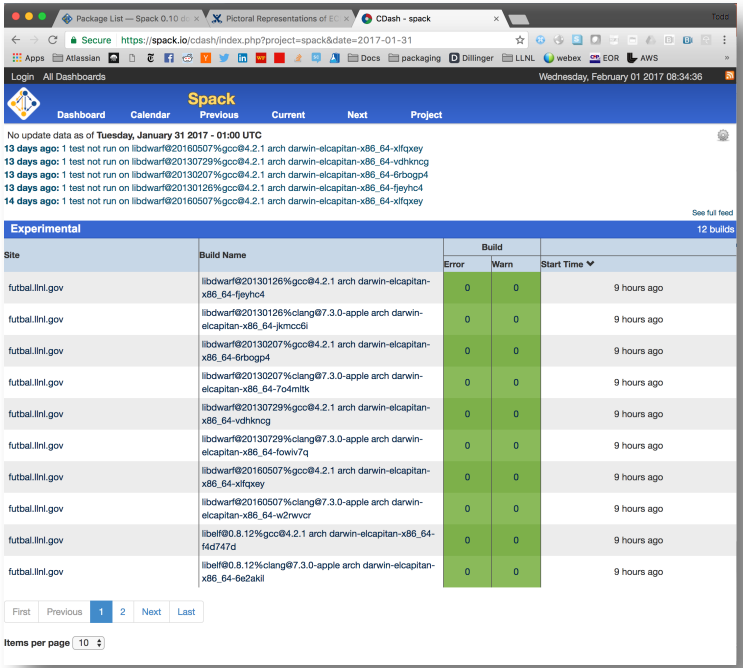
- NERSC, OLCF, and ANL LCRC have contributed extensively to Spack
  - Cray support & packaging
  - ALCF is investigating Spack
- Spack can automatically generate module files for installed packages
- We are looking into Spack as a mechanism for facility users to make their software available publicly.
- At LLNL we have developed support for auto-generating RPMs with Spack.



We can use Spack to build an ECP software distribution

- We have begun to build a public CDash server that shows Spack build status
- We would like to coordinate with LCFs to run Spack tests and report results publicly
  - Users could see in real-time where ECP builds are working, where they are failing
- We are working towards native, optimized binary packages in Spack
  - CERN contributed a first cut of this feature.

https://spack.io/cdash



All of these features require support for continuous integration from facilities.

Spack can be used with containers to rapidly create reproducible environments

- Containers like Docker, Singularity, Shifter, etc. provide a convenient way to snapshot an entire application environment
- Spack can be used to build applications outside or inside a container
- We would like to provide integration with Singularity containers with MPI
- This could serve as a deployment model for ECP.

We are actively seeking collaborations with ECP teams

- **To build the ECP stack, we need help from ECP teams.**
  - Already working with CEED, xSDK, LLNL teams.
- **We are interested in shared milestones to package ECP software suites with Spack.**
  - We will help teams to build packages and to contribute them to the public Spack project.
- **Join us at our breakout this afternoon**
  1:30– 4:00pm, Room 301D
  - xSDK will discuss their software stack first
  - Spack session to follow
  - We are particularly interested in hearing about teams' packaging/deployment needs.
- **Or email tgamblin@llnl.gov, or visit our poster**

github.com/LLNL/spack

@spackpm

Get Spack!


Lawrence Livermore
National Laboratory