

Spack Supercomputer PACKage Manager

Daniel Topa

CCS-2: Computational Physics and Methods

dantopa@lanl.gov
03-200-264
6-0817

2018-01-17

Overview

- 1. What is Spack?**
- 2. What does Spack do?**
- 3. How does Spack work?**
- 4. Why Use Spack?**

Challenge: Install Spack, GSL

- 1. Clone Spack**
- 2. Install GNU Scientific Library (GSL)**

Challenge: Install Spack, GSL

- 1. Clone Spack**
- 2. cd to new directory**
- 3. Set environment variables**
- 4. Install GNU Scientific Library (GSL)**

Challenge: Install Spack, GSL

1. \$ git clone https://github.com/spack/spack mySpack
2. \$ cd mySpack
3. \$. share/spack/setup-env.sh
4. \$ Spack install gsl

Results: Clone Spack

```
$ git clone https://github.com/spack/spack mySpack
Cloning into 'mySpack'...
remote: Counting objects: 98699, done.
remote: Compressing objects: 100% (14/14), done.
remote: Total 98699 (delta 9), reused 1 (delta 1), pack-reused
98684
Receiving objects: 100% (98699/98699), 34.05 MiB | 15.80 MiB/s,
done.
```

Results: Install GSL - Summary

```
$ spack install gsl
==> Installing gsl
==> . . .
==> Successfully installed gsl
    Fetch: 3.59s. Build: 2m 29.58s. Total: 2m 33.17s.
[+] /Users/l127914/mySpack/opt/spack/darwin-sierra-x86_64/
clang-9.0.0-apple/ gsl-2.4-2sems7ybkixmyqmxu6hp4z4nlhpgqkrl
```



Results: Install GSL

```
$ spack install gsl
==> Installing gsl
==> Fetching http://mirror.switch.ch/ftp/mirror/gnu/gsl/...
#####
# 100.0%
==> Staging archive: /Users/l127914/mySpack/var/spack/stage/...
==> Created stage in /Users/l127914/mySpack/var/spack/stage/...
==> No patches needed for gsl
==> Building gsl [AutotoolsPackage]
==> Executing phase: 'autoreconf'
==> Executing phase: 'configure'
==> Executing phase: 'build'
==> Executing phase: 'install'
==> Successfully installed gsl
Fetch: 3.59s. Build: 2m 29.58s. Total: 2m 33.17s.
[+] /Users/l127914/mySpack/opt/spack/darwin-sierra-x86_64/
clang-9.0.0-apple/ gsl-2.4-2sems7ybkixmyqmxu6hp4z4nlhpgqkrl
```

Results: Install GSL- Verify

```
$ spack find  
==> 1 installed package  
-- darwin-sierra-x86_64 / clang@9.0.0-apple -----  
gsl@2.4
```

Hash Encodes Version, Compiler, Architecture



CCS Net: spack find -ldf hypre

```
-- linux-rhel7-x86_64 / gcc@6.4.0 ----

ppqpzkf    hypre@2.13.0%gcc
qmqnjy6      ^openblas@0.2.20%gcc
nw4jyek      ^openmpi@1.10.5%gcc
nul7v5e       ^hwloc@1.11.8%gcc
o53uqv7       ^libpciaccess@0.13.5%gcc
pnpcuxa       ^libxml2@2.9.4%gcc
pfw5gjb        ^xz@5.2.3%gcc
w43e56t        ^zlib@1.2.11%gcc

3ff6771    hypre@2.13.0%gcc
qmqnjy6      ^openblas@0.2.20%gcc
lwe5lft       ^openmpi@2.1.2%gcc
nul7v5e       ^hwloc@1.11.8%gcc
o53uqv7       ^libpciaccess@0.13.5%gcc
pnpcuxa       ^libxml2@2.9.4%gcc
pfw5gjb        ^xz@5.2.3%gcc
w43e56t        ^zlib@1.2.11%gcc
```

Hash Encodes Full Provenance



About Spack

- ▶ Supercomputer **PACKage Manager**
- ▶ Open source
- ▶ Started by Todd Gamblin (LLNL)
- ▶ Best hope for HPC software control
- ▶ Version 0.11

Spack Introduction - SC15

The Spack Package Manager: Bringing Order to HPC Software Chaos

Todd Gamblin
tgamblin@llnl.gov

Matthew LeGendre
legendre1@llnl.gov

Michael R. Collette
mcollette@llnl.gov

Gregory L. Lee
lee218@llnl.gov

Adam Moody
moody20@llnl.gov

Bronis R. de Supinski
bronis@llnl.gov

Scott Futral
futral@llnl.gov

Lawrence Livermore National Laboratory

ABSTRACT

Large HPC centers spend considerable time supporting software for thousands of users, but the complexity of HPC software is quickly outpacing the capabilities of existing software management tools. Scientific applications require specific versions of compilers, MPI, and other dependency libraries, so using a single, standard software stack is infeasible. However, managing many configurations is difficult because the configuration space is combinatorial in size.

We introduce Spack, a tool used at Lawrence Livermore National Laboratory to manage this complexity. Spack provides a novel, recursive specification syntax to invoke parametric builds of packages and dependencies. It allows any number of builds to coexist on the same system, and it ensures that installed packages can find their dependencies, *regardless of the environment*. We show through real-world use cases that Spack supports diverse and demanding applications, bringing order to HPC software chaos.

Worse, the space of required builds grows combinatorially with each new configuration parameter. As a result, LLNL staff spend countless hours dealing with build and deployment issues.

Existing package management tools automate parts of the build process [2, 10, 11, 12, 23, 24, 38, 39, 41]. For the most part, they focus on keeping a single, stable set of packages up to date, and they do not handle installation of multiple versions or configurations. Those that *do* handle multiple configurations typically require that package files be created for each combination of options [10, 11, 12, 23], leading to a profusion of files and maintenance issues. Some allow limited forms of composition [11, 12, 23], but their dependency management is overly rigid, and they burden users with combinatorial naming and versioning problems.

This paper describes our experiences with the *Spack* package manager, which we have developed at LLNL to manage increasing software complexity. It makes the following contributions:

Spack: A Path to Exascale



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-606064

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract DE-AC52-07NA27345. Lawrence Livermore National Security, LLC.

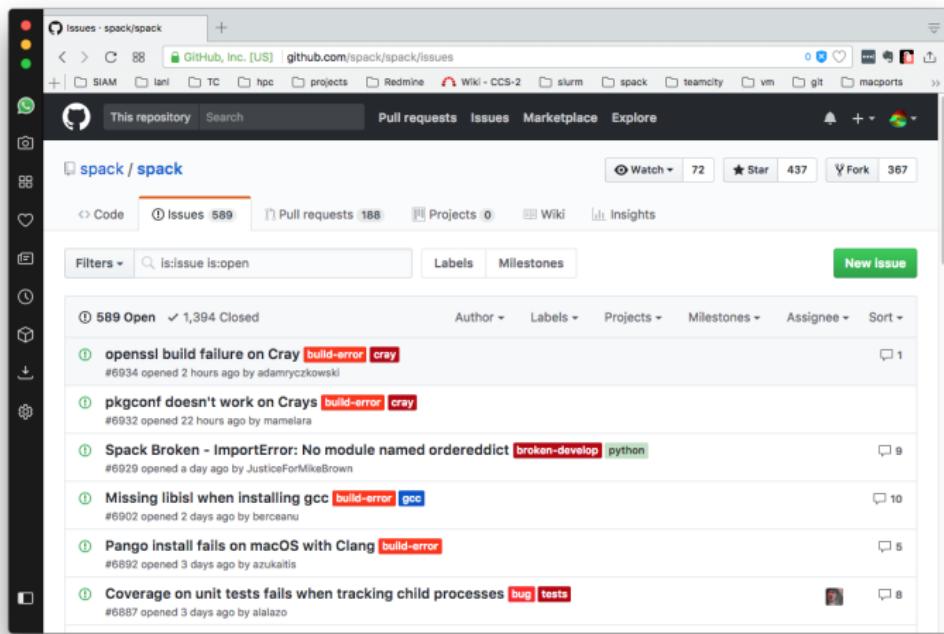
<https://spack.io>

 Lawrence Livermore
National Laboratory

Vibrant Development Community

- ▶ GitHub Issue Tracking
- ▶ Google Discussion Group
- ▶ Weekly Teleconference
- ▶ New User Guide
- ▶ Tutorials
- ▶ [LANL: user_contrib](#)

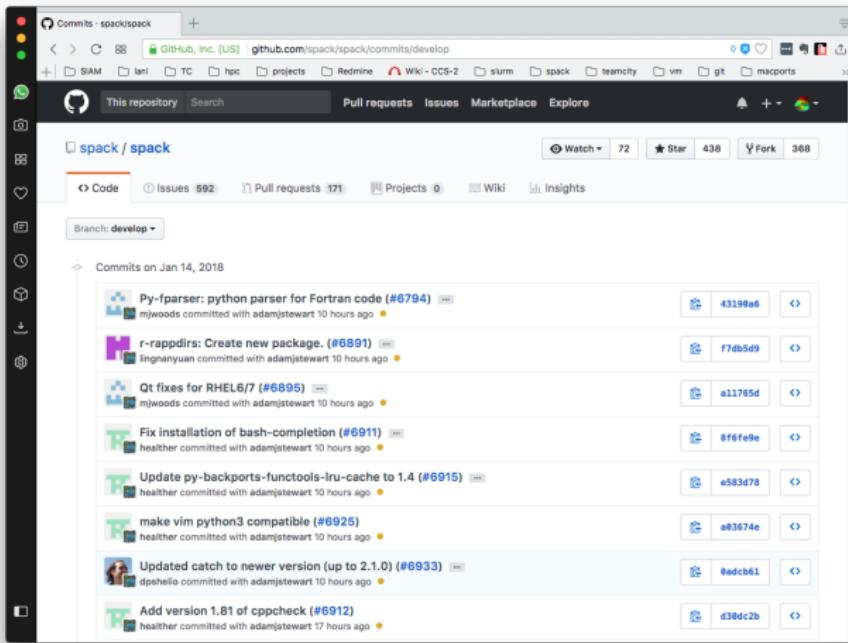
Github Issue Tracking



The screenshot shows the GitHub Issues page for the `spack/spack` repository. The page displays 589 open issues. The issues listed are:

- openssl build failure on Cray [build-error](#) [cray](#) #6934 opened 2 hours ago by adamryczkowski
- pkgconf doesn't work on Crays [build-error](#) [cray](#) #6932 opened 22 hours ago by mameilara
- Spack Broken - ImportError: No module named ordereddict [broken-develop](#) [python](#) #6929 opened a day ago by JusticeForMikeBrown
- Missing liblsl when installing gcc [build-error](#) [gcc](#) #6902 opened 2 days ago by berceanu
- Pango install fails on macOS with Clang [build-error](#) #6892 opened 3 days ago by azukaitis
- Coverage on unit tests fails when tracking child processes [bug](#) [tests](#) #6887 opened 3 days ago by alalazo

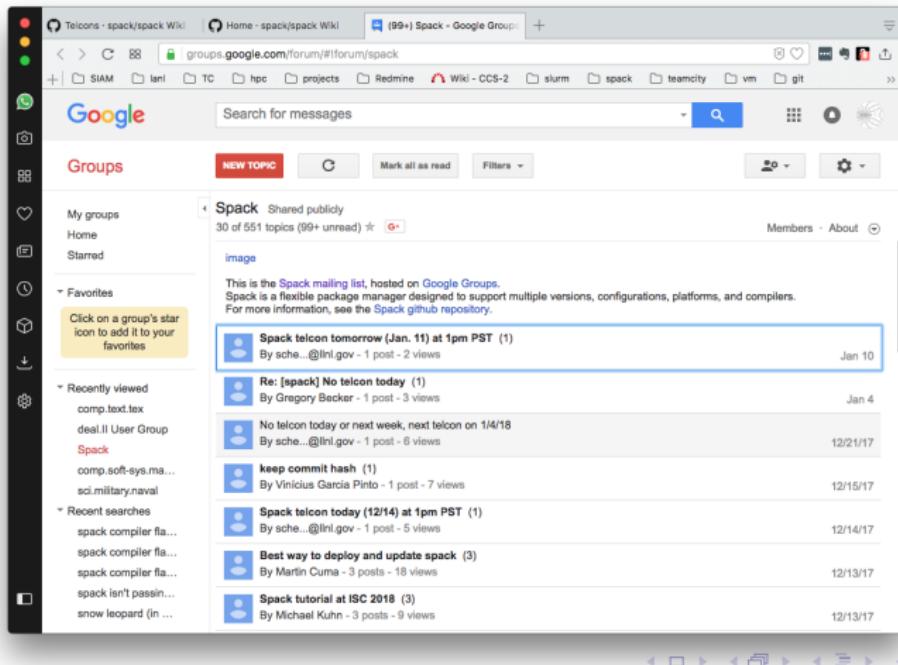
Dynamic Development Team



The screenshot shows a GitHub repository page for 'spack / spack'. The 'Code' tab is selected, and the 'Branch: develop' dropdown is open. Below it, a list of recent commits on Jan 14, 2018, is displayed:

- Py-fparser: python parser for Fortran code (#6794) - mwoods committed with adamjstewart 10 hours ago
- r-rappdirs: Create new package. (#6801) - lingyuanyan committed with adamjstewart 10 hours ago
- Qt fixes for RHEL6/7 (#6895) - mwoods committed with adamjstewart 10 hours ago
- Fix installation of bash-completion (#6911) - healthier committed with adamjstewart 10 hours ago
- Update py-backports-functools-lru-cache to 1.4 (#6915) - healthier committed with adamjstewart 10 hours ago
- make vim python3 compatible (#6925) - healthier committed with adamjstewart 10 hours ago
- Updated catch to newer version (up to 2.1.0) (#6933) - dphelio committed with adamjstewart 10 hours ago
- Add version 1.81 of cppcheck (#6912) - healthier committed with adamjstewart 17 hours ago

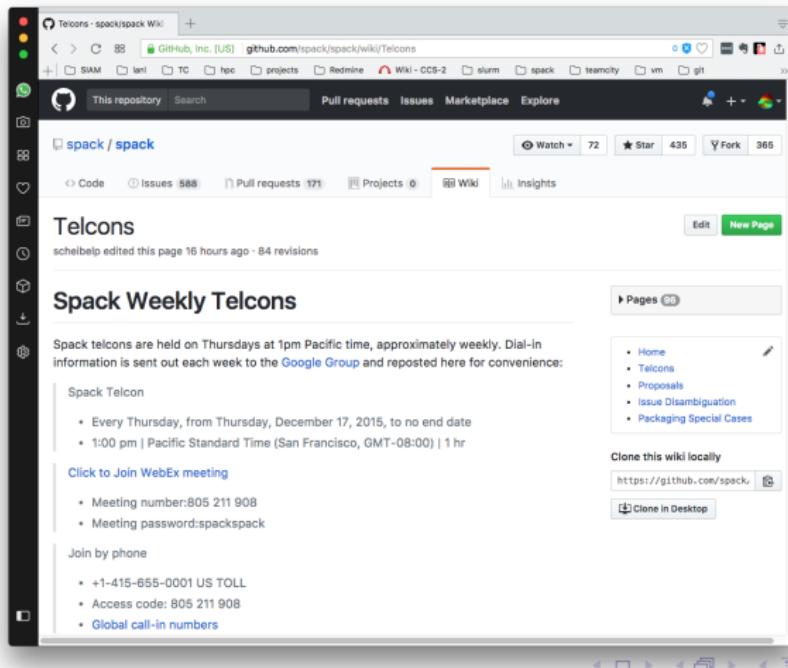
Spack Google Group



The screenshot shows a web browser window displaying the Spack Google Group. The URL in the address bar is groups.google.com/forum/#forum/spack. The page has a sidebar on the left with various navigation links like 'Groups', 'My groups', 'Home', 'Starred', 'Favorites', 'Recently viewed', and 'Recent searches'. The main content area shows a list of messages under the 'Spack' group. One message is highlighted with a blue border: 'Spack telcon tomorrow (Jan. 11) at 1pm PST (1)' by sch...@llnl.gov. Other messages listed include 'Re: [spack] No telcon today (1)', 'No telcon today or next week, next telcon on 1/4/18', 'keep commit hash (1)', 'Spack telcon today (12/14) at 1pm PST (1)', 'Best way to deploy and update spack (3)', and 'Spack tutorial at ISC 2018 (3)'. The bottom of the browser window shows standard navigation controls.

Message	Author	Date
Spack telcon tomorrow (Jan. 11) at 1pm PST (1)	sch...@llnl.gov	Jan 10
Re: [spack] No telcon today (1)	Gregory Becker	Jan 4
No telcon today or next week, next telcon on 1/4/18	sch...@llnl.gov	12/21/17
keep commit hash (1)	Vinicius Garcia Pinto	12/15/17
Spack telcon today (12/14) at 1pm PST (1)	sch...@llnl.gov	12/14/17
Best way to deploy and update spack (3)	Martin Cuma	12/13/17
Spack tutorial at ISC 2018 (3)	Michael Kuhn	12/13/17

Weekly Teleconference



The screenshot shows a web browser displaying a GitHub wiki page for the Spack repository. The page is titled "Telcons" and contains information about weekly teleconferences. It includes details on meeting times, joining via WebEx, and phone numbers. A sidebar on the right provides links to other pages like Home, Telcons, Proposals, and Issue Disambiguation.

Telcons
scheibelp edited this page 16 hours ago · 84 revisions

Spack Weekly Telcons

Spack telcons are held on Thursdays at 1pm Pacific time, approximately weekly. Dial-in information is sent out each week to the [Google Group](#) and reposted here for convenience:

Spack Telcon

- Every Thursday, from Thursday, December 17, 2015, to no end date
- 1:00 pm | Pacific Standard Time (San Francisco, GMT-08:00) | 1 hr

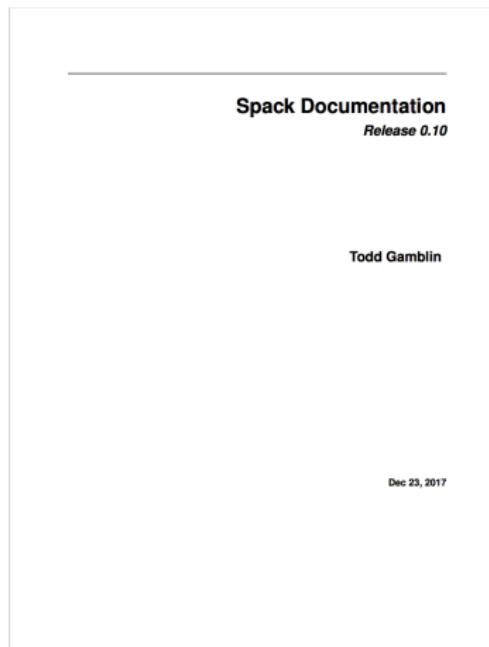
Click to Join WebEx meeting

- Meeting number: 805 211 908
- Meeting password: spackspack

Join by phone

- +1-415-655-0001 US TOLL
- Access code: 805 211 908
- Global call-in numbers

Spack User Guide: 600+ pp



Spack Documentation
Release 0.10

Todd Gamblin

Dec 23, 2017

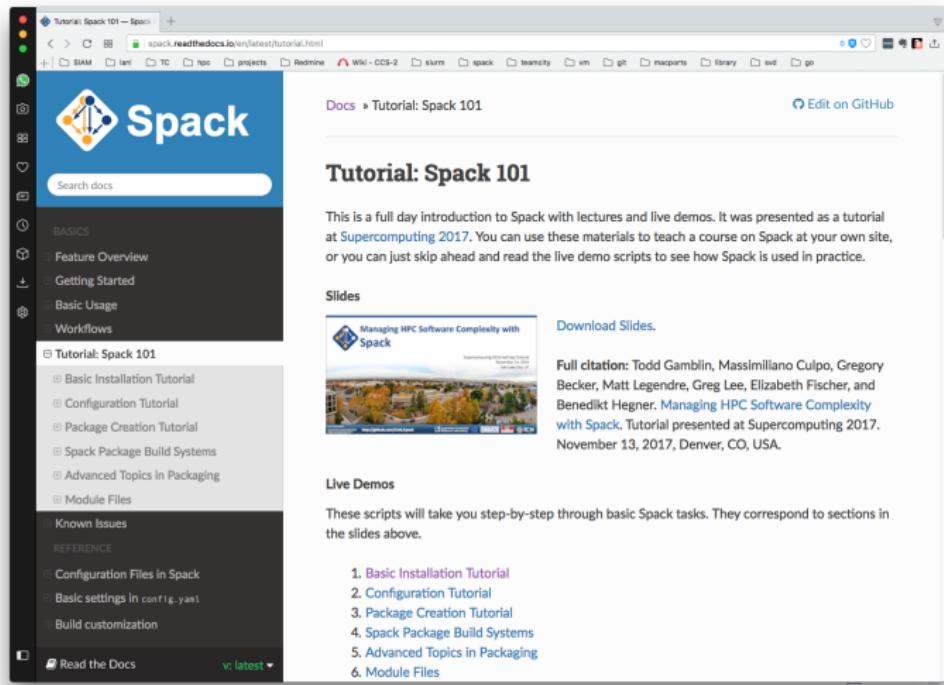
Spack User Guide: Part I - Users

<p>Basics</p> <hr/> <p>1 Feature Overview</p> <ul style="list-style-type: none"> 1.1 Simple package installation 3 1.2 Custom versions & configurations 3 1.3 Customer dependencies 4 1.4 Non-destructive installs 4 1.5 Packages are easily creatable 4 1.6 Creating packages is easy 4 <p>2 Getting Started</p> <ul style="list-style-type: none"> 2.1 Prerequisites 7 2.2 Installation 7 2.3 Configuration 9 2.4 Vendor-Specific Compiler Configuration 13 2.5 System Packages 16 2.6 Using External Packages 18 2.7 GPG Signing 20 2.8 Spack on Cray 21 <p>3 Basic Usage</p> <ul style="list-style-type: none"> 3.1 Listing available packages 38 3.2 Installing packages 39 3.3 Seeing installed packages 42 3.4 Spec & dependencies 44 3.5 Virtual dependencies 46 3.6 Building packages 50 3.7 Filesystem requirements 53 3.8 Getting Help 57 <p>4 Workflows</p> <ul style="list-style-type: none"> 4.1 Definitions 59 4.2 Building Packages 60 4.3 Running Scripts From Packages 62 4.4 Developing Software With Spack 68 4.5 Using Spack in Other Distros 73 4.6 Using Spack in Cross Docker Images 74 4.7 Upstream Bug Fixes 76 	<p>5 Tutorial: Spack 101 79</p> <ul style="list-style-type: none"> 5.1 Basic Installation Tutorial 80 5.2 Configuration Tutorial 121 5.3 Package Creation Tutorial 137 5.4 Packaging Programs In Systems 148 5.5 Advanced Topics In Packaging 182 5.6 Module Files 192 <p>6 Known Issues 223</p> <ul style="list-style-type: none"> 6.1 Default variants are not taken into account during concretization 223 6.2 Spack fails to detect dependencies 223 6.3 spack setup doesn't work 224 <p>7 Configuration Files in Spack 228</p> <ul style="list-style-type: none"> 7.1 YAML Format 228 7.2 Configuration Scopes 236 7.3 Configuration Sources 238 7.4 Spec precedence 239 7.5 Config file variables 227 <p>8 Basic settings in config.yaml 231</p> <ul style="list-style-type: none"> 8.1 install_tree 231 8.2 build_stage_directory 232 8.3 module_root 233 8.4 build_stage 233 8.5 source_cache 234 8.6 spack 234 8.7 verify_gpg 234 8.8 checksums 234 8.9 build_jobs 234 8.10 build_jobs 235 <p>9 Build customization 327</p> <ul style="list-style-type: none"> 9.1 External Packages 327 9.2 Concretization Preferences 328 <p>10 Mirrors 341</p> <ul style="list-style-type: none"> 10.1 spack mirror 341 10.2 spack mirror add 342 10.3 spack mirror addS 343 10.4 spack mirror list 344 10.5 spack mirror remove 344 10.6 Mirror preferences 344 10.7 Local Default Cache 344 <p>11 Modules 345</p> <ul style="list-style-type: none"> 11.1 Using module files via Spack 345 11.2 Module file concretization 349 11.3 Generating Module Files 353 <p>12 Packing Requirements 359</p> <ul style="list-style-type: none"> 12.1 repack 360 12.2 Namespaces 360 12.3 Overriding built-in packages 361 12.4 spack repzo 363
---	--

Spack User Guide: Part II - Developers

12.5 Ray namespaces and Python	265
13 Build cache	267
13.1 Creating build cache files	267
13.2 Finding or installing build cache files	267
13.3 Calculation	268
13.4 spack buildcache	268
14 Command Index	271
15 Contribution Guide	273
15.1 Continuous Integration	273
15.2 Git Workflows	273
16 Packaging Guide	281
16.1 Creating & editing packages	281
16.2 Naming & directory structure	284
16.3 Trusted Downloads	286
16.4 License Headers	287
16.5 Finding new versions	290
16.6 Fetching from code repositories	292
16.7 Generating external dependencies	292
16.8 Licensed software	294
16.9 Paths	296
16.10 Using Bzip2	296
16.11 Parallel builds	299
16.12 Dependencies	300
16.13 Conflicts	304
16.14 External dependencies	304
16.15 Virtual dependencies	307
16.16 Abstract & concrete specs	308
16.17 Spec constraints	310
16.18 Implementing the installation procedure	311
16.19 The build environment	313
16.20 Spack’s build system	314
16.21 Compiler wrappers	321
16.22 MPI support in Spack	322
16.23 Shared libraries	323
16.24 File manipulation functions	325
16.25 Style guidelines for packages	327
16.26 Using Spack’s conventions	329
16.27 Graphing dependencies	330
16.28 Interactive shell support	334
17 Developer Guide	339
17.1 Overview	339
17.2 Directory Structure	339
17.3 Code Standard	341
17.4 Spec objects	342
17.5 Package objects	342
17.6 Ray objects	343
17.7 Writing commands	343
17.8 Unit tests	343
17.9 Unit testing	343
17.10 Developer commands	343
18 spack package	347
18.1 Subpackages	347
18.2 Submodules	444
18.3 spack module	444
18.4 spack.environment module	445
18.5 spack.build_environment module	448
18.6 spack.compiler module	451
18.7 spack.compiler_exec module	452
18.8 spack.config module	453
18.9 spack.database module	455
18.10 spack.environment module	457
18.12 spack.directives module	458
18.13 spack.directory_layout module	460
18.14 spack.environment module	460
18.15 spack.error module	467
18.16 spack.fetch_strategy module	467
18.17 spack.fetcher module	472
18.18 spack.fetcher_view module	473
18.19 spack.pkg module	475
18.20 spack.repo module	476
18.21 spack.repo module	477
18.22 spack.maintainable module	478
18.23 spack.package module	479
18.24 spack.package_fetch module	480
18.25 spack.package_list module	491
18.26 spack.package module	491
18.27 spack.provider module	492
18.28 spack.provider_index module	493
18.29 spack.reference module	494
18.30 spack.repository module	495
18.31 spack.resource module	499
18.32 spack.spec module	499
18.33 spack.spec_index module	500
18.34 spack.store module	510
18.35 spack.tegic module	511
18.36 spack.version module	511
18.37 spack.variant module	515
18.38 spack.version module	519
18.39 Module contents	522
19 Ray package	585
19.1 Subpackages	585
19.2 Module contents	571
20 Indices and tables	573
Python Module Index	575

Spack Tutorial



The screenshot shows a web browser window displaying the 'Tutorial: Spack 101' page from the Spack documentation site. The browser's address bar shows the URL: spack.readthedocs.io/en/latest/tutorial.html. The page has a dark blue header with the Spack logo and a search bar. A sidebar on the left contains links for 'BASICS', 'Slides', 'Live Demos', and 'REFERENCE'. The main content area features a slide titled 'Managing HPC Software Complexity with Spack' by Todd Gamblin, Massimiliano Culpo, Gregory Becker, Matt Legendre, Greg Lee, Elizabeth Fischer, and Benedikt Hegner. It also lists 'Basic Installation Tutorial', 'Configuration Tutorial', 'Package Creation Tutorial', 'Spack Package Build Systems', 'Advanced Topics in Packaging', and 'Module Files' as live demos.

Spack Tutorial SC17



Managing HPC Software Complexity with Spack

Supercomputing 2017 Full-day Tutorial
November, 2017
Denver, Colorado



LLNL-PRES-000054

github.com/spack/spack

 Lawrence Livermore
National Laboratory

 SCITAS

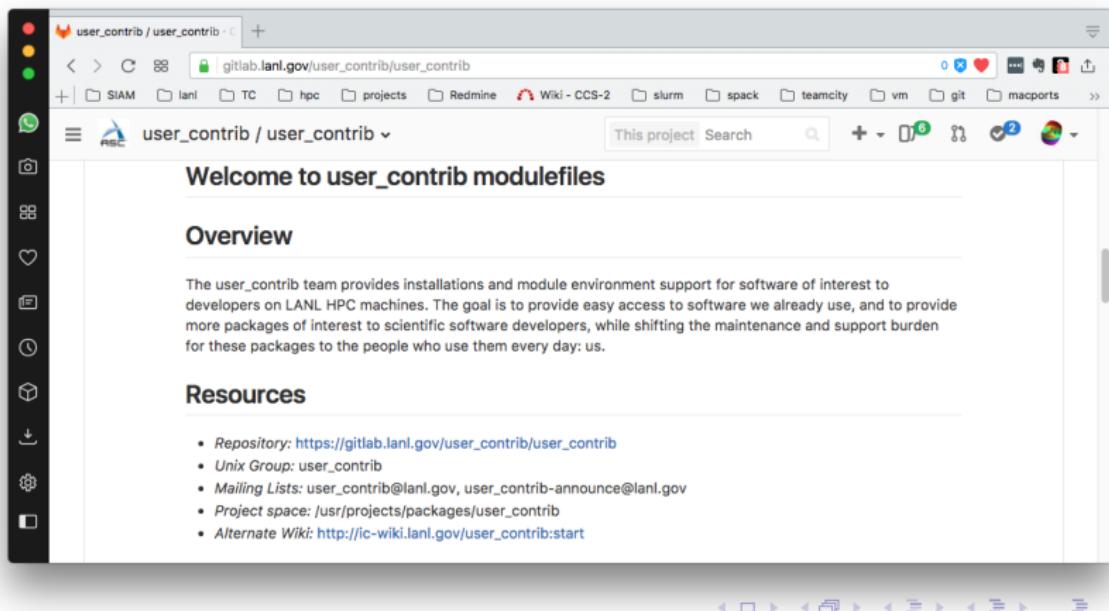


 NERSC

 Argonne
NATIONAL LABORATORY

 SC17
Supercomputing Conference

LANL: user_contrib



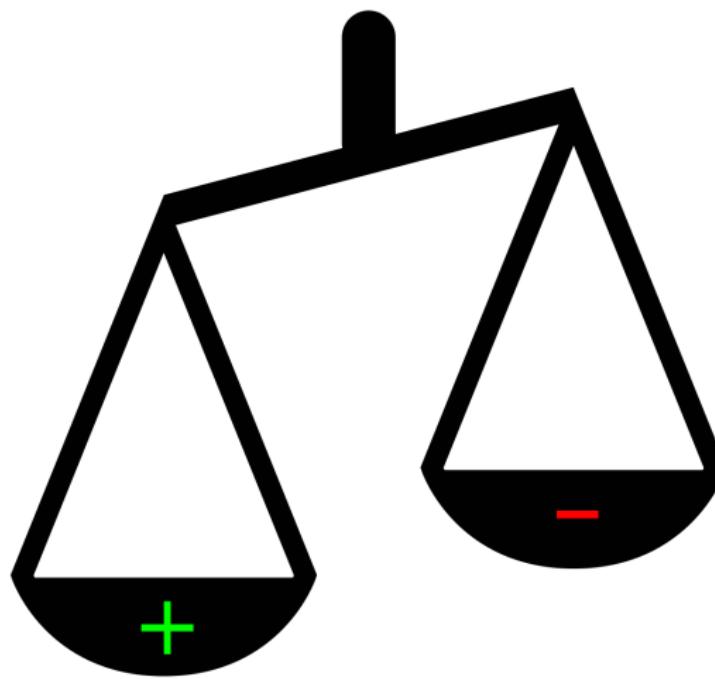
The screenshot shows a web browser window with the URL gitlab.lanl.gov/user_contrib/user_contrib. The page title is "Welcome to user_contrib modulefiles". The content includes an "Overview" section stating: "The user_contrib team provides installations and module environment support for software of interest to developers on LANL HPC machines. The goal is to provide easy access to software we already use, and to provide more packages of interest to scientific software developers, while shifting the maintenance and support burden for these packages to the people who use them every day: us." Below this is a "Resources" section with the following bullet points:

- **Repository:** https://gitlab.lanl.gov/user_contrib/user_contrib
- **Unix Group:** user_contrib
- **Mailing Lists:** user_contrib@lanl.gov, user_contrib-announce@lanl.gov
- **Project space:** /usr/projects/packages/user_contrib
- **Alternate Wiki:** http://ic-wiki.lanl.gov/user_contrib:start

Caveat Emptor

- ▶ Pre-beta
- ▶ # issues > # developers
- ▶ Punished for our sins
- ▶ Success demands discipline

Benefits >> Inconveniences



A Simple Script to Build a Software Stack

1. Select **4** Compilers
2. Build **2** Versions of Python
3. Install, Activate Several Python Packages
4. Build Trilinos against **3** Versions of OpenMPI

Simplicity: Script I

```
#!/bin/bash

ver_mpi="1.10.5 2.1.2 3.0.0"; ver_gcc="5.3.0 6.4.0 7.2.0"

for v in ${ver_gcc}; do
    spack install gcc @ ${v}
done
```

Simplicity: Script II

```
#!/bin/bash

ver_mpi="1.10.5 2.1.2 3.0.0"; ver_gcc="5.3.0 6.4.0 7.2.0"

for v in ${ver_gcc}; do
    spack install gcc @ ${v}
done

for v in ${ver_gcc}; do
    spack install python @ 2.7.14 % gcc @ ${v}
    spack install python             % gcc @ ${v}
    for m in ${ver_mpi}; do
        spack install trilinos % gcc @ ${v} ^openmpi @ ${m}
    done
done
```

Trilinos: 4 Compilers, 3 OpenMPIs, 2 Pythons

```
-- linux-rhel7-x86_64 / gcc@4.8.5 -----
autoconf@2.69      libbsd@0.8.6        netlib-scalapack@2.0.2    py-pyparsing@2.2.0
automake@1.15.1    libedit@3.1-20170329  openblas@0.2.20       py-pytz@2017.2
binutils@2.29.1    ibiconv@1.15       openmpi@1.10.5       py-requests@2.14.2
bison@3.0.4         libjpeg-turbo@1.5.0   openmpi@2.1.2       py-setuptools@35.0.2
boost@1.65.1        libpciaccess@0.13.5  openmpi@3.0.0       py-setuptools@35.0.2
bzip2@1.0.6         libpng@1.6.29     openssl@1.0.2n     py-six@1.10.0
cmake@3.10.1        libpthread-stubs@0.4  parmetis@4.0.3       py-six@1.10.0
curl@7.56.0         libsigsegv@0.11    parmetis@4.0.3       py-sympy@1.0
expat@2.2.2         libtool@2.4.6     parmetis@4.0.3       py-sympy@1.0
flex@2.6.4          libx11@1.6.5     pcre@8.41           python@2.7.14
freetype@2.7.1      libxau@1.0.8     pcre@8.41           python@3.6.2
gcc@5.3.0           libxcb@1.12      perl@5.24.1         qhull@2015.2
gcc@6.4.0           libxdmcp@1.1.2    pkgconf@1.4.0       readline@7.0
gcc@7.2.0           libxml2@2.9.4    py-appdirs@1.4.3    sqlite@3.21.0
gdbm@1.14.1         llvm@5.0.0      py-appdirs@1.4.3    suite-sparse@5.1.0
gettext@0.19.8.1    m4@1.4.18     py-configparser@3.5.0 superlu-dist@5.2.2
git@2.15.1          matio@1.5.9     py-cycler@0.10.0    superlu-dist@5.2.2
glm@0.9.7.1         matio@1.5.9     py-cycler@0.10.0    superlu-dist@5.2.2
gmp@6.1.2           matio@1.5.9     py-dateutil@2.5.2   swig@3.0.12
gsl@2.4              metis@5.1.0     py-enum34@1.1.6     tar@1.29
hdf5@1.10.1         mpc@1.0.3      py-gnuplot@1.8      tcl@8.6.6
hdf5@1.10.1         mpfr@3.1.5     py-lit@0.5.0        tk@8.6.6
hdf5@1.10.1         mumps@5.1.1    py-lit@0.5.0        trilinos@12.12.1
help2man@1.47.4     mumps@5.1.1    py-mpmath@0.19       trilinos@12.12.1
hwloc@1.11.8        mumps@5.1.1    py-mpmath@0.19       trilinos@12.12.1
```

Trilinos: 4 Compilers, 3 OpenMPI, 2 Python

hypre@2.13.0	nasm@2.11.06	py-numpy@1.13.3	util-macros@1.19.1
hypre@2.13.0	ncurses@6.0	py-numpy@1.13.3	xcb-proto@1.12
hypre@2.13.0	netcdf@4.4.1.1	py-packaging@16.8	xextproto@7.3.0
inputproto@2.3.2	netcdf@4.4.1.1	py-packaging@16.8	xproto@7.0.31
isl@0.14	netcdf@4.4.1.1	py-pbr@1.10.0	xtrans@1.3.5
isl@0.18	netlib-scalapack@2.0.2	py-pillow@3.2.0	xz@5.2.3
kbproto@1.0.7	netlib-scalapack@2.0.2	py-pyparsing@2.2.0	zlib@1.2.11

- linux-rhel7-x86_64 / gcc@5.3.0 -----

autoconf@2.69	libpciaccess@0.13.5	openmpi@3.0.0	py-setuptools@35.0.2
automake@1.15.1	libpng@1.6.29	openssl@1.0.2n	py-setuptools@35.0.2
binutils@2.29.1	libpthread-stubs@0.4	parmetis@4.0.3	py-six@1.10.0
bison@3.0.4	libssegv@2.11	parmetis@4.0.3	py-six@1.10.0
boost@1.65.1	libtool@2.4.6	parmetis@4.0.3	py-sympy@1.0
bzip2@1.0.6	libx11@1.6.5	pcre@8.41	py-sympy@1.0
cmake@3.10.1	libxau@1.0.8	perl@5.24.1	python@2.7.14
curl@7.56.0	libxcb@1.12	pkgconf@1.4.0	python@3.6.2
expat@2.2.2	libxdmcp@1.1.2	py-appdirs@1.4.3	qhull@2015.2
flex@2.6.4	libxml2@2.9.4	py-appdirs@1.4.3	readline@7.0
freetype@2.7.1	m4@1.4.18	py-configparser@3.5.0	sqlite@3.21.0
gdbm@1.14.1	matio@1.5.9	py-cycler@0.10.0	suite-sparse@5.1.0
gettext@0.19.8.1	matio@1.5.9	py-dateutil@2.5.2	superlu-dist@5.2.2
git@2.15.1	matio@1.5.9	py-enum34@1.1.6	superlu-dist@5.2.2
glm@0.9.7.1	metis@5.1.0	py-gnuplot@1.8	superlu-dist@5.2.2

Trilinos: 4 Compilers, 3 OpenMPI, 2 Python

```

gsl@2.4      mumps@5.1.1    py-lit@0.5.0      tar@1.29
hdf5@1.10.1  mumps@5.1.1    py-lit@0.5.0      tcl@8.6.6
hdf5@1.10.1  mumps@5.1.1    py-mpmath@0.19    tk@8.6.6

- linux-rhel7-x86_64 / gcc@6.4.0 ----

autoconf@2.69    libpciaccess@0.13.5    openmpi@3.0.0      py-setuptools@35.0.2
automake@1.15.1  libpng@1.6.29       openssl@1.0.2n    py-setuptools@35.0.2
binutils@2.29.1  libpthread-stubs@0.4   parmetis@4.0.3     py-six@1.10.0
bison@3.0.4      libsigsegv@0.11       parmetis@4.0.3     py-sympy@1.0
boost@1.65.1     libtool@2.4.6        pcre@8.41        py-sympy@1.0
bzip2@1.0.6      libx11@1.6.5       perl@5.24.1      python@2.7.14
cmake@3.10.1     libxau@1.0.8       pkgconf@1.4.0     python@3.6.2
curl@7.56.0      libxcb@1.12        py-appdirs@1.4.3   qhull@2015.2
expat@2.2.2      libxdmcp@1.1.2     py-appdirs@1.4.3   readline@7.0
flex@2.6.4       libxml2@2.9.4      py-configparser@3.5.0  sqlite@3.21.0
freetype@2.7.1   m4@1.4.18        py-cycler@0.10.0   suite-sparse@5.1.0
gdbm@1.14.1      matio@1.5.9       py-dateutil@2.5.2  superlu-dist@5.2.2
gettext@0.19.8.1  matio@1.5.9       py-enum34@1.1.6   superlu-dist@5.2.2
git@2.15.1       matio@1.5.9       py-gnuplot@1.8    superlu-dist@5.2.2
glm@0.9.7.1      metis@5.1.0      py-lit@0.5.0      tar@1.29
gsl@2.4          mumps@5.1.1      py-lit@0.5.0      tcl@8.6.6
hdf5@1.10.1     mumps@5.1.1      py-mpmath@0.19    tk@8.6.6
hdf5@1.10.1     mumps@5.1.1

```

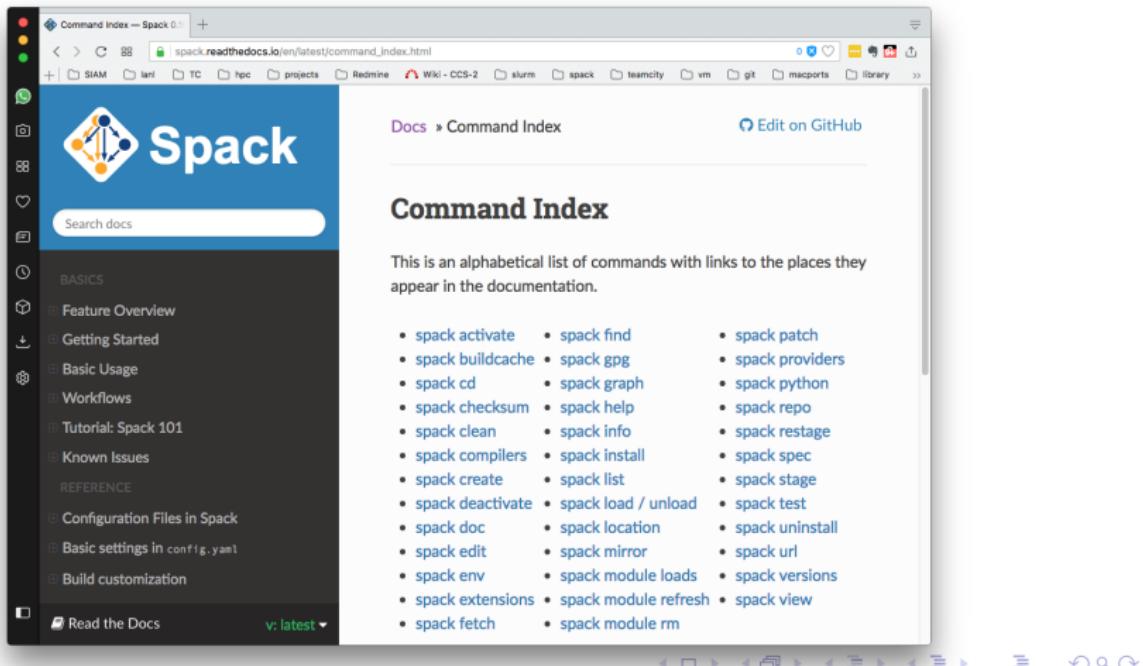
Spack Programming Elements

1. Command Set
2. Sigils
3. Configuration Files (*.yaml)
4. Either commands – Or yaml files

More Details Here:

- ▶ Configuration Files in Spack
- ▶ Build Customization

Spack Command Set



The screenshot shows a web browser displaying the "Command Index" page for Spack 0.5. The page has a blue header with the Spack logo and a search bar. The left sidebar contains navigation links for "BASICS", "Feature Overview", "Getting Started", "Basic Usage", "Workflows", "Tutorial: Spack 101", and "Known Issues". Below these are "REFERENCE" links for "Configuration Files in Spack", "Basic settings in config.yaml", and "Build customization". At the bottom of the sidebar are "Read the Docs" and "v: latest" buttons. The main content area is titled "Command Index" and includes a sub-header "Edit on GitHub". It describes the page as an alphabetical list of commands with links to their documentation. A large list of Spack commands is provided in three columns:

- spack activate
- spack find
- spack patch
- spack buildcache
- spack gpg
- spack providers
- spack cd
- spack graph
- spack python
- spack checksum
- spack help
- spack repo
- spack clean
- spack info
- spack restage
- spack compilers
- spack install
- spack spec
- spack create
- spack list
- spack stage
- spack deactivate
- spack load / unload
- spack test
- spack doc
- spack location
- spack uninstall
- spack edit
- spack mirror
- spack url
- spack env
- spack module loads
- spack versions
- spack extensions
- spack module refresh
- spack view
- spack fetch
- spack module rm

Spack Command Set

(de)activate	arch	buildcache	cd	checksum
clean	compiler	compilers	config	edit
env	extensions	fetch	find	gpg
graph	help	info	(un)install	list
(un)load	location	mirror	module	patch
providers	python	repo	spec	(re)stage
test	url	versions	view	

Spack Workhorse Commands

(de)activate	arch	buildcache	cd	checksum
clean	compiler	compilers	config	edit
env	extensions	fetch	find	gpg
graph	help	info	(un)install	list
(un)load	location	mirror	module	patch
providers	python	repo	spec	(re)stage
test	url	versions	view	

Info Command Examples: providers (CCS)

```
$ spack providers mpi
intel-mpi    mpich    mpich@3:    mvapich2@1.9    openmpi
openmpi@1.7.5:    spectrum-mpi    intel-parallel-studio+mpi
mpich@1:    mvapich2    mvapich2@2.0:    openmpi@1.6.5
openmpi@2.0.0:
```



```
$ spack providers lapack
atlas    intel-mkl    intel-parallel-studio+mkl
netlib-lapack    openblas    veclibfort
```

Info Command Examples: compilers (Power8+)

```
$ spack compilers
==> Available compilers

-- clang centos7-x86_64 -----
clang@5.0.1 clang@5.0.0 clang@4.0.1 clang@3.9.1 clang@3.8.1 clang@3.7.1 clang@3.5.2

-- gcc centos7-x86_64 -----
gcc@7.2.0 gcc@7.1.0 gcc@6.3.0 gcc@6.2.0 gcc@6.1.0 gcc@5.4.0 gcc@5.3.0 gcc@5.2.0 gcc@5.1.0
gcc@4.9.3 gcc@4.9.2 gcc@4.9.1 gcc@4.9.0 gcc@4.8.5 gcc@4.6.4

-- gcc rhel7-ppc64le -----
gcc@7.2.0 gcc@7.1.0 gcc@6.3.0 gcc@5.4.0 gcc@4.8.5

-- intel centos7-x86_64 -----
intel@18.0.1 intel@17.0.0 intel@16.0.1 intel@15.0.3 intel@15.0.0 intel@13.1.2 intel@13.1.0
intel@17.0.1 intel@16.0.3 intel@16.0.0 intel@15.0.1 intel@14.0.2 intel@13.1.1 intel@13.0.1

-- pgi centos7-x86_64 -----
pgi@17.4 pgi@16.10 pgi@16.5 pgi@16.1 pgi@15.10 pgi@15.7

-- pgi rhel7-ppc64le -----
pgi@17.5 pgi@16.10
```

Info Command Examples: versions (CCS)

```
$ spack versions hypre
==> Safe versions (already checksummed):
develop 2.13.0 2.12.1 2.11.2 2.11.1 2.10.1 2.10.0b xsdk-0.2.0
==> Remote versions (not yet checksummed):
2.13.0-28-g42e267b 2.12.0-2-ga34db78 2.12.
```

Info Command Examples: arch (CCS)

```
dantopa@cn2000:spack.power8.2018-01-13 $ spack arch  
linux-rhel7-ppc64le
```

```
dantopa@tt-fey1:spack.tt.2018-01-04 $ spack arch  
cray-CNL-haswell
```

```
dantopa@ccscs2.lanl.gov:spack.ccs.2018-01-114 $ spack arch  
linux-rhel7-x86_64
```

```
1127914@pn1249300.lanl.gov $ spack arch  
darwin-sierra-x86_64
```

Spack Sigils

sigil | 'sijəl |

noun

an inscribed or painted symbol considered to have magical power.

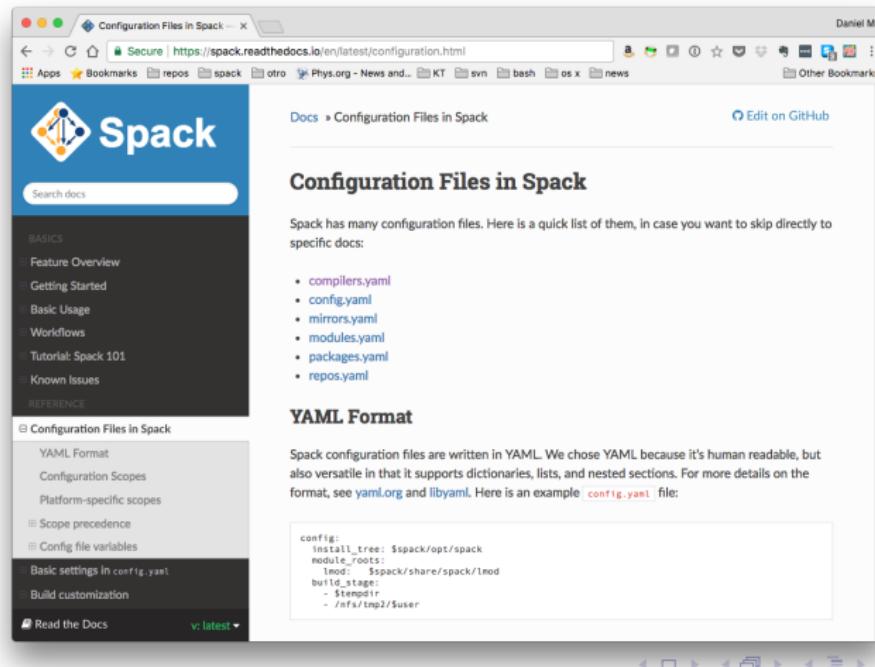
Spack Sigils

sigil	denotes
@	package version
%	compiler version
^	dependencies
+	Boolean TRUE
-, ~	Boolean FALSE
/	hash

Sigil Examples

```
spack install hypre
spack install hypre @ 2.12.1
spack install hypre % gcc @ 4.8.5
spack install hypre +mpi ^ mpich
```

You Have To Read The Manual



The screenshot shows a web browser window displaying the Spack Configuration Files documentation. The URL is <https://spack.readthedocs.io/en/latest/configuration.html>. The page title is "Configuration Files in Spack". The left sidebar has sections for "BASICS" (Feature Overview, Getting Started, Basic Usage, Workflows, Tutorial: Spack 101, Known Issues) and "REFERENCE" (Configuration Files in Spack, YAML Format, Configuration Scopes, Platform-specific scopes, Scope precedence, Config file variables). The main content area starts with a list of configuration files: compilers.yaml, config.yaml, mirrors.yaml, modules.yaml, packages.yaml, and repos.yaml. Below this is a section titled "YAML Format" which explains that configuration files are written in YAML and provides a code example:

```
config:
    install_tree: $spack/opt/spack
    module_roots:
        - /tmp
    build_stage:
        build_stagedir:
            - /nfs/tmp2/$user
```

Spack Configuration Files

- 1. config.yaml**
- 2. compilers.yaml**
- 3. mirrors.yaml**
- 4. modules.yaml**
- 5. packages.yaml**
- 6. repos.yaml**

CCS: modules.yaml

```
modules:
  enable:
    - tcl
    - dotkit
  tcl:
    hash_length:  0
    naming_scheme:
      '$PACKAGE}/${VERSION}-${COMPILERNAME}-${COMPILERVER}'
      all:
        suffixes:
          ^openmpi@1.10.5:  'openmpi@1.10.5'
          ^openmpi@2.1.0:  'openmpi@2.1.0'
          ^openmpi@2.2.0:  'openmpi@2.2.0'
          ^openmpi@3.0.0:  'openmpi@3.0.0'
          ^python@2.7.14:  'python@2.7.14'
          ^python@3.6.2:  'python@3.6.2'
```

CCS: packages.yaml

```
packages:  
  all:  
    compiler: [gcc, intel, pgi, clang, xl, nag]  
    providers:  
      blas: [openblas]  
      lapack: [openblas]  
      mkl: [intel-mkl]  
      mpi: [openmpi, mpich]  
      scalapack: [netlib-scalapack]
```

CCS: compilers.yaml I

```
compilers:
- compiler:
  environment:  {}
  extra_rpaths: []
  flags:  {}
  operating_system: rhel7
  paths:
    cc: /usr/bin/gcc
    cxx: /usr/bin/g++
    f77: /usr/bin/gfortran
    fc: /usr/bin/gfortran
  spec: gcc4.8.5
  target: x86_64
```

CCS: compilers.yaml II

```
compilers:
- compiler:
  environment: {}
  extra_rpaths: []
  flags: {}
  operating_system: rhel7
  paths:
    cc:
      /scratch/dantopa/spack.ccs.developmental/spack.ccs.2018-01-11/opt/spack/linux-rhel7-x86_64/gcc-4.8.5/gcc-4.8.5
        cxx:
          /scratch/dantopa/spack.ccs.developmental/spack.ccs.2018-01-11/opt/spack/linux-rhel7-x86_64/gcc-4.8.5/gcc-4.8.5
            f77:
              /scratch/dantopa/spack.ccs.developmental/spack.ccs.2018-01-11/opt/spack/linux-rhel7-x86_64/gcc-4.8.5/gcc-4.8.5
                fc:
                  /scratch/dantopa/spack.ccs.developmental/spack.ccs.2018-01-11/opt/spack/linux-rhel7-x86_64/gcc-4.8.5/gcc-4.8.5
                    spec: gcc7.2.0
                    target: x86_64
```

spack config get ...

spack config get {
 compilers
 config
 mirrors
 modules
 packages
 repos

Command Sequence

1. spack install hypre
2. spack install hypre % gcc @4.8.5
3. spack install hypre % gcc @4.8.5 +mpi ~mpich
4. spack install hypre % gcc @4.8.5 +mpi ~openmpi@2.1.2
5. spack find -ldf hypre
6. spack spec /jk47cxx

Load Compilers from MacPorts I

```
$ port select -list gcc
Available versions for gcc:
  mp-gcc47
  mp-gcc48
  mp-gcc5
  mp-gcc6
  mp-gcc7 (active)
  mp-gcc8
  none
$ sudo port select -set gcc mp-gcc8
Selecting 'mp-gcc8' for 'gcc' succeeded.  'mp-gcc8' is now
active.
```

Load Compilers from MacPorts II

```
$ spack compiler find
==> Added 1 new compiler to
/Users/l127914/.spack/darwin/compilers.yaml
    gcc@7.2.0
==> Compilers are defined in the following files:
    /Users/l127914/.spack/darwin/compilers.yaml
```

Task: Load Compilers from Modules on Darwin

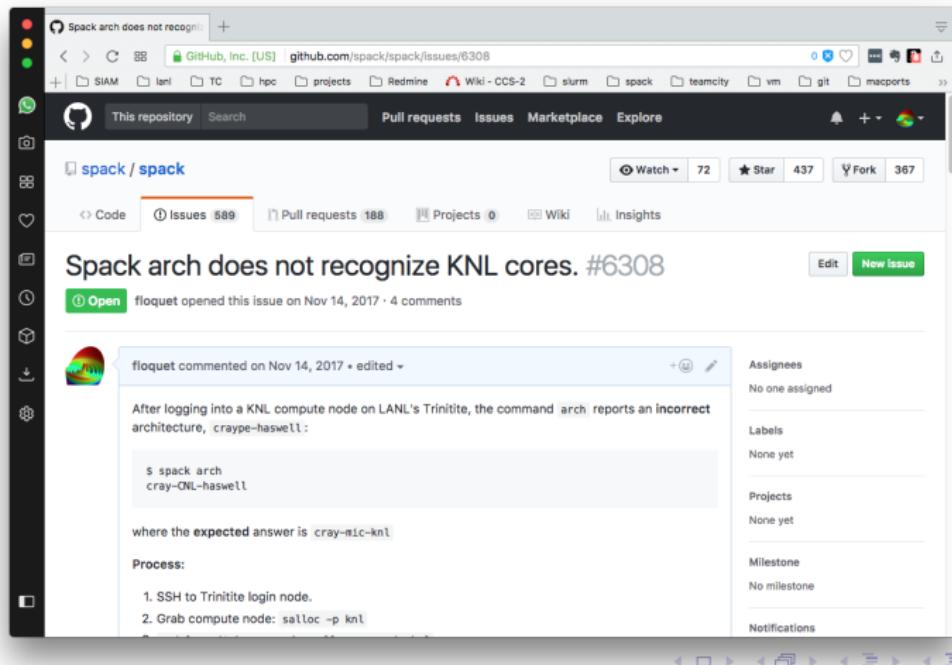
Table: module avail excerpt

clang/3.5.2	gcc/4.6.4	intel/13.0.1
clang/3.7.1	gcc/4.9.0	intel/13.1.0
clang/3.8.1	gcc/4.9.1	intel/13.1.1
clang/3.9.1	gcc/4.9.2	intel/13.1.2
clang/4.0.1	gcc/4.9.3(default)	intel/14.0.2
clang/5.0.1(default)	gcc/5.1.0	intel/15.0.0
pgi/15.10	gcc/5.2.0	intel/15.0.1
pgi/15.7	gcc/5.3.0	intel/15.0.3(default)
pgi/16.1	gcc/6.1.0	intel/16.0.0
pgi/16.10(default)	gcc/6.2.0	intel/16.0.1
pgi/16.5	gcc/6.3.0	intel/16.0.3
pgi/17.4	gcc/7.1.0	intel/17.0.0
	gcc/7.2.0	intel/17.0.1
		intel/18.0.1

Load Compilers from Modules

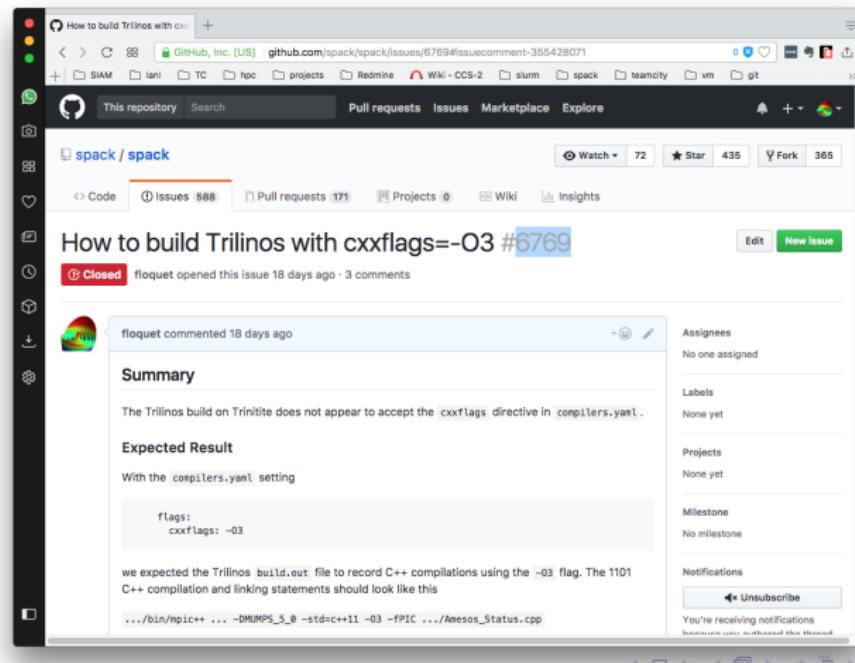
```
$ module load <vendor/version>
$ spack compiler find
$ module purge
```

Spack doesn't recognize KNL (6308)



The screenshot shows a GitHub issue page for the Spack repository. The title of the issue is "Spack arch does not recognize KNL cores. #6308". The issue was opened by floquet on Nov 14, 2017, with 4 comments. The user floquet commented on Nov 14, 2017, stating: "After logging into a KNL compute node on LANL's Trinitite, the command `arch` reports an incorrect architecture, `cray-pe-haswell`". Below this comment, there is a code block showing the command `$ spack arch cray-CNL-haswell`. The user also notes that the expected answer is `cray-mlc-knl`. The issue has 589 issues, 188 pull requests, 0 projects, and 437 stars. The right sidebar shows assignees (None assigned), labels (None yet), projects (None yet), and milestones (No milestone). Notifications are also listed.

CMake won't accept compiler flags (6769)



The screenshot shows a GitHub issue page for the Spack repository. The issue is titled "How to build Trilinos with cxxflags=-O3 #6769". It is marked as "Closed" by floquet, who opened it 18 days ago and has 3 comments. The summary notes that the Trilinos build on Trinitite does not appear to accept the `cxxflags` directive in `compilers.yaml`. The expected result is that with the `compilers.yaml` setting:

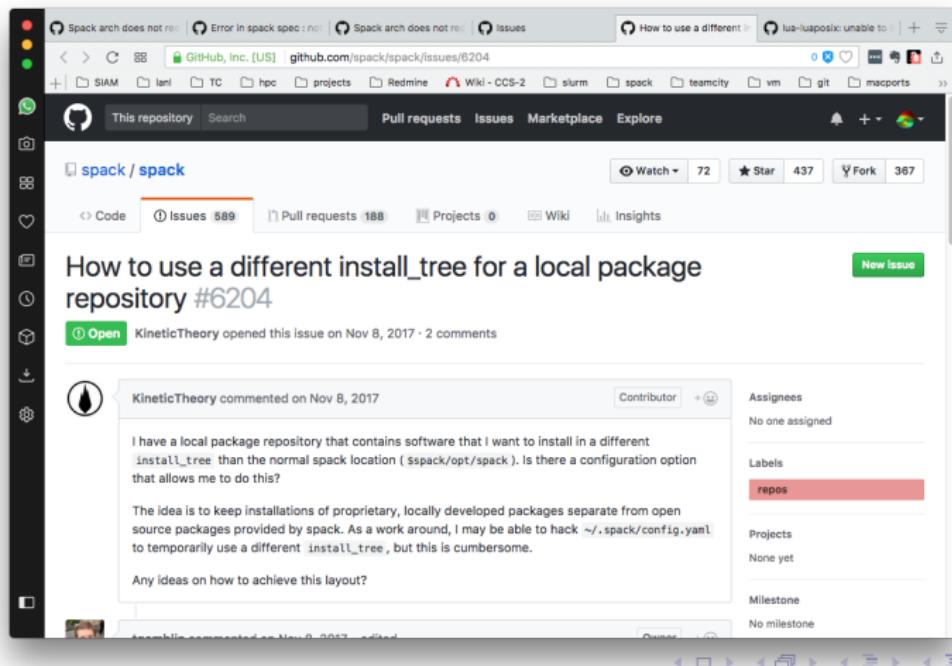
```
flags:
cxxflags: -O3
```

the build.out file would record C++ compilations using the `-O3` flag. The user notes that C++ compilation and linking statements should look like this:

```
.../bin/mpic++ ... -DMUMPS_5_0 -std=c++11 -O3 -fPIC .../Amesos_Status.cpp
```

On the right side of the issue page, there are sections for Assignees (No one assigned), Labels (None yet), Projects (None yet), Milestone (No milestone), and Notifications (You're receiving notifications). There is also a "Unsubscribe" button.

How to use a different install_tree (6204)



The screenshot shows a GitHub issue page for a Spack repository. The title of the issue is "How to use a different install_tree for a local package repository #6204". The issue was opened by KineticTheory on Nov 8, 2017, with 2 comments. A comment from KineticTheory on Nov 8, 2017, discusses the desire to keep locally developed packages separate from open source packages by using a different install tree. Another comment from the same user suggests using a configuration file to achieve this. The issue has been assigned to the 'repos' label. The GitHub interface includes a sidebar with various icons and a header with navigation links like 'Issues' and 'Marketplace'.

How to use a different install_tree for a local package repository #6204

Open KineticTheory opened this issue on Nov 8, 2017 · 2 comments

KineticTheory commented on Nov 8, 2017

I have a local package repository that contains software that I want to install in a different `install_tree` than the normal spack location (`$spack/opt/spack`). Is there a configuration option that allows me to do this?

The idea is to keep installations of proprietary, locally developed packages separate from open source packages provided by spack. As a work around, I may be able to hack `~/.spack/config.yaml` to temporarily use a different `install_tree`, but this is cumbersome.

Any ideas on how to achieve this layout?

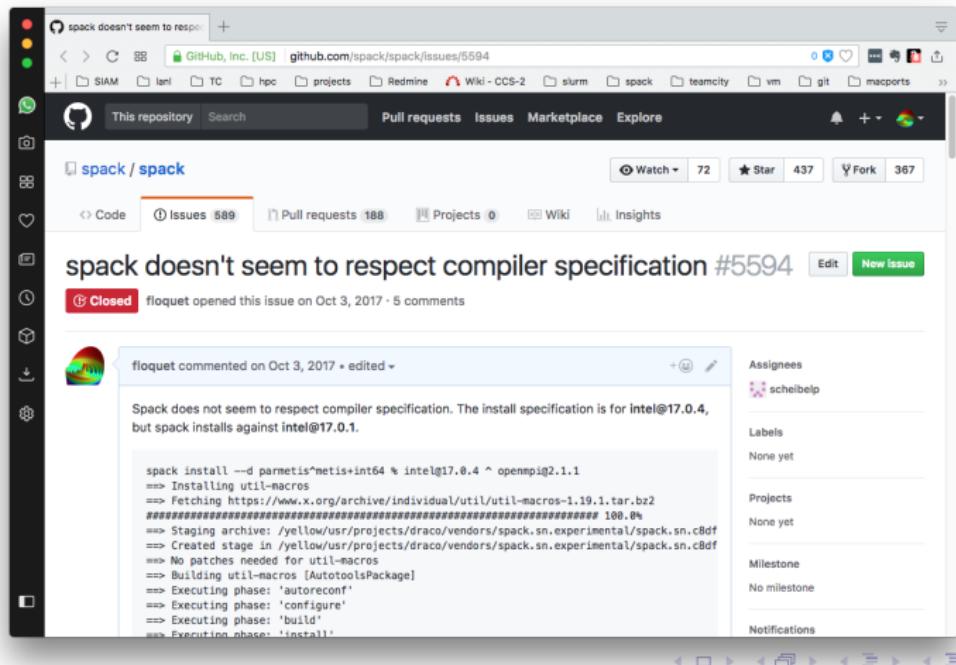
Assignees
No one assigned

Labels
repos

Projects
None yet

Milestone
No milestone

Compiler selection (5594)

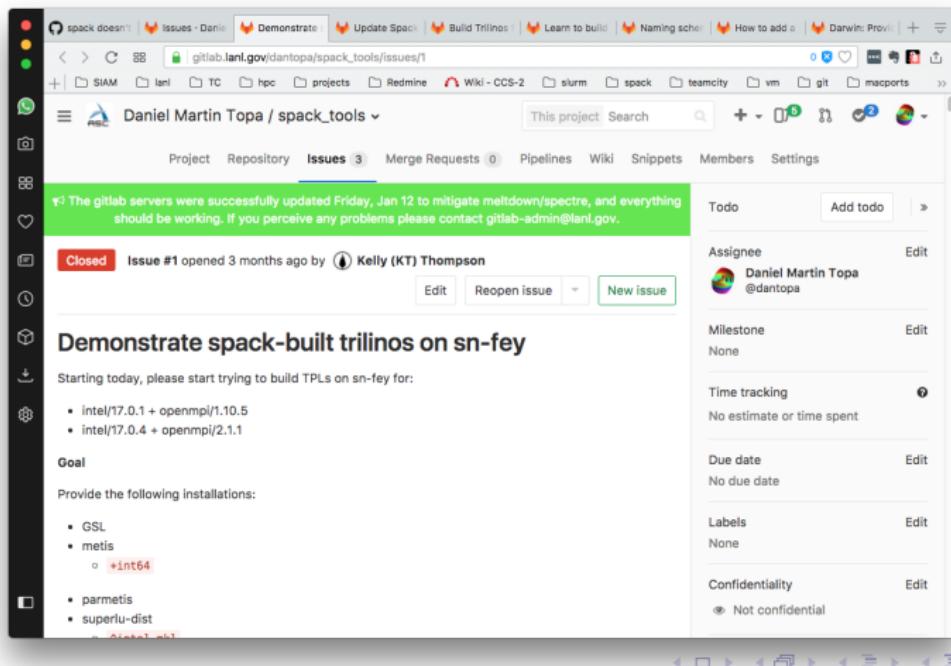


The screenshot shows a GitHub issue page for the Spack repository. The issue is titled "spack doesn't seem to respect compiler specification #5594". It is marked as "Closed" by floquet on Oct 3, 2017, with 5 comments. The issue body contains a screenshot of a terminal showing a spack install command failing to respect a compiler specification. The terminal output is as follows:

```
spack install --add parmetismetis+int64 % intel@17.0.4 ^ openmpi2.1.1
=> Installing util-macros
=> Fetching https://www.x.org/archive/individual/util/util-macros-1.19.1.tar.bz2
=====
=> Staging archive: /yellow/usr/projects/draco/vendors/spack.sn.experimental/spack.sn.c8df
=> Created stage in /yellow/usr/projects/draco/vendors/spack.sn.experimental/spack.sn.c8df
=> No patches needed for util-macros
=> Building util-macros [AutotoolsPackage]
=> Executing phase: 'autoreconf'
=> Executing phase: 'configure'
=> Executing phase: 'build'
=> Executing phase: 'install'
```

The issue has 589 issues, 188 pull requests, 0 projects, and 437 stars. The assignee is schelbelp, and there are no labels or milestones. Notifications are turned off.

Build Trilinos on Snow (1)



The screenshot shows a GitLab issue page for a project named "Daniel Martin Topa / spack_tools". The page has a header with navigation links like "Project", "Repository", "Issues 3", "Merge Requests 0", "Pipelines", "Wiki", "Snippets", "Members", and "Settings". A green status bar at the top indicates: "The gitlab servers were successfully updated Friday, Jan 12 to mitigate meltdown/spectre, and everything should be working. If you perceive any problems please contact gitlab-admin@lanl.gov." Below this, there is a "Closed" issue titled "#1 opened 3 months ago by Kelly (KT) Thompson". The issue details section includes buttons for "Edit", "Reopen issue", and "New issue". The main content area contains a heading "Demonstrate spack-built trilinos on sn-fey" and a list of instructions: "Starting today, please start trying to build TPLs on sn-fey:

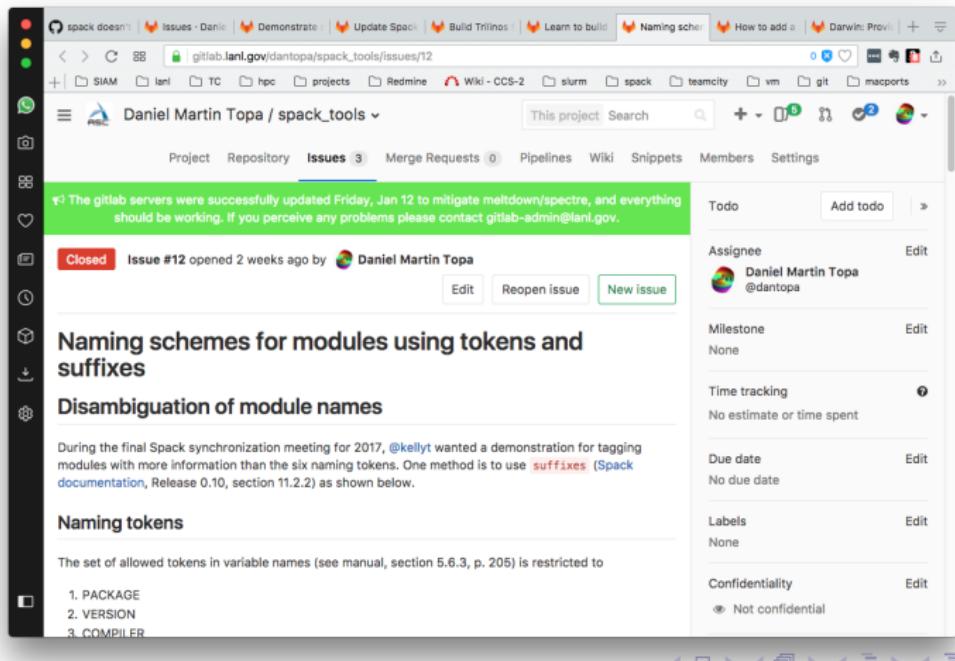
- intel/17.0.1 + openmpi/1.10.5
- intel/17.0.4 + openmpi/2.1.1

". There is also a "Goal" section with a list of required installations: "Provide the following installations:

- GSL
- metis
 - +int64
- parmetis
- superlu-dist
 - +int64

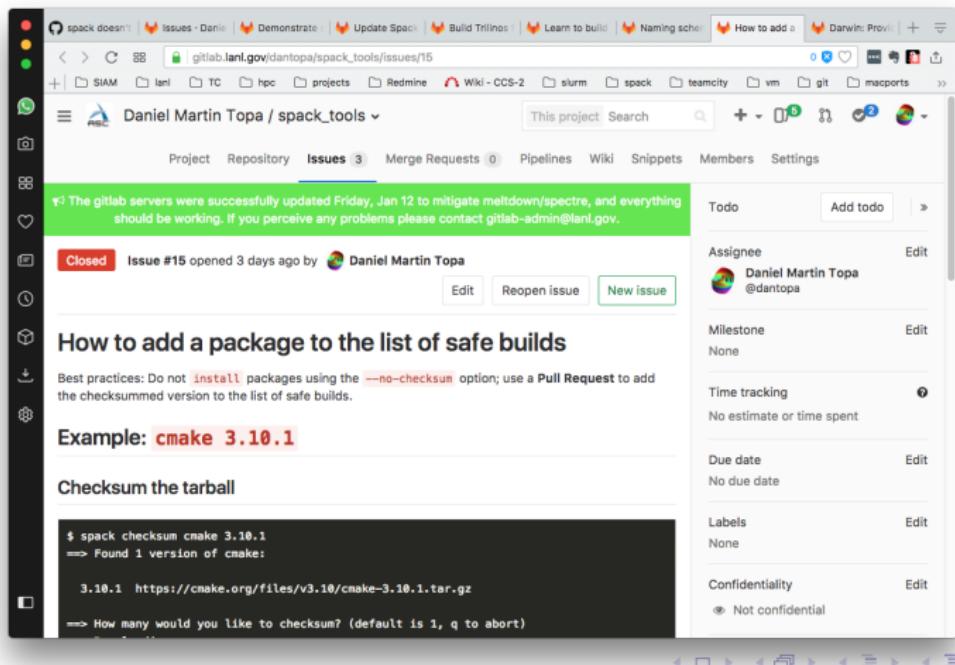
". On the right side of the page, there are edit buttons for "Assignee" (set to Daniel Martin Topa), "Milestone" (None), "Time tracking" (No estimate or time spent), "Due date" (No due date), "Labels" (None), and "Confidentiality" (Not confidential).

Naming Spack Modules (12)



The screenshot shows a GitLab issue page for a project named "Daniel Martin Topa / spack_tools". The page has a header with navigation links like Project, Repository, Issues (3), Merge Requests (0), Pipelines, Wiki, Snippets, Members, and Settings. On the left is a sidebar with various icons. The main content area has a green banner at the top with the text: "The gitlab servers were successfully updated Friday, Jan 12 to mitigate meltdown/spectre, and everything should be working. If you perceive any problems please contact gitlab-admin@lanl.gov." Below the banner is a "Closed" issue #12, opened 2 weeks ago by Daniel Martin Topa. The issue title is "Naming schemes for modules using tokens and suffixes". Underneath the title is a section titled "Disambiguation of module names". A text block explains: "During the final Spack synchronization meeting for 2017, @kellyt wanted a demonstration for tagging modules with more information than the six naming tokens. One method is to use [suffixes](#) (Spack documentation, Release 0.10, section 11.2.2) as shown below." Below this is a section titled "Naming tokens" with a list: "1. PACKAGE", "2. VERSION", and "3. COMPILER". To the right of the issue details is a sidebar with sections for Todo, Assignee (Daniel Martin Topa), Milestone (None), Time tracking (No estimate or time spent), Due date (No due date), Labels (None), and Confidentiality (Not confidential).

Safe Builds (15)



The screenshot shows a GitLab issue page for the project "Daniel Martin Topa / spack_tools". The issue is titled "How to add a package to the list of safe builds". It has 3 issues and 0 merge requests. The issue is closed and was opened 3 days ago by Daniel Martin Topa. The content of the issue includes a brief description of best practices for adding packages to the safe builds list, an example command ("cmake 3.10.1"), and a terminal session showing the execution of the command.

How to add a package to the list of safe builds

Best practices: Do not `install` packages using the `--no-checksum` option; use a **Pull Request** to add the checksummed version to the list of safe builds.

Example: `cmake 3.10.1`

Checksum the tarball

```
$ spack checksum cmake 3.10.1
=> Found 1 version of cmake:
    3.10.1 https://cmake.org/files/v3.10/cmake-3.10.1.tar.gz
=> How many would you like to checksum? (default is 1, q to abort)
```

Spack

Supercomputer PACKage Manager

Daniel Topa

CCS-2: Computational Physics and Methods

dantopa@lanl.gov
03-200-264
6-0817

2018-01-17