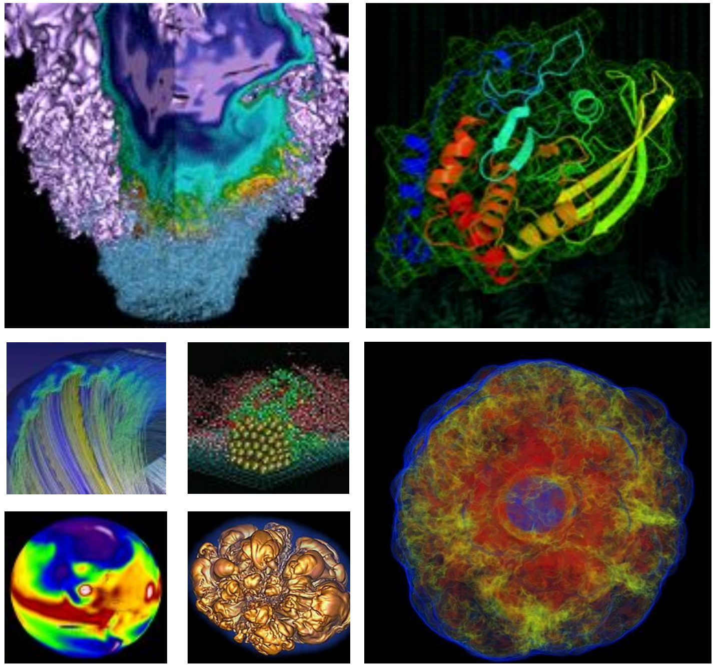
NUG Monthly Meeting



NERSC

18 November, 2021

U.S. DEPARTMENT OF ENERGY | Office of Science

BERKELEY LAB

Today's plan

- Interactive - please participate!
  - Raise hand or just speak up
  - **NERSC User Slack** (link in chat), **#webinars** channel

- Agenda:
  - Win-of-the-month
  - Today-I-learned
  - Announcements/CFPs
  - Topic of the day: **Spack at NERSC** by Steve Leak.
  - Coming meetings: topic suggestions/requests?
  - Last (two!) month's numbers

Win of the month

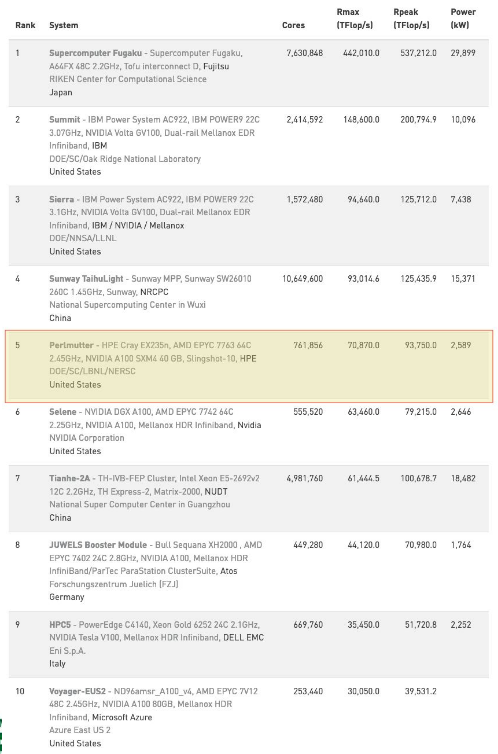
Show off an achievement, or shout out someone else's achievement, e.g.:

- Had a paper accepted
- Solved a bug
- A scientific achievement (maybe candidate for Science highlight, or **High Impact Scientific Achievement award**)
- An **Innovative Use of High Performance Computing** (also a candidate for an award) (https://www.nersc.gov/science/nersc-hpc-achievement-awards/)

Please let us know of award-worthy work from you or your colleagues - tell us what you did, and what was the key insight?

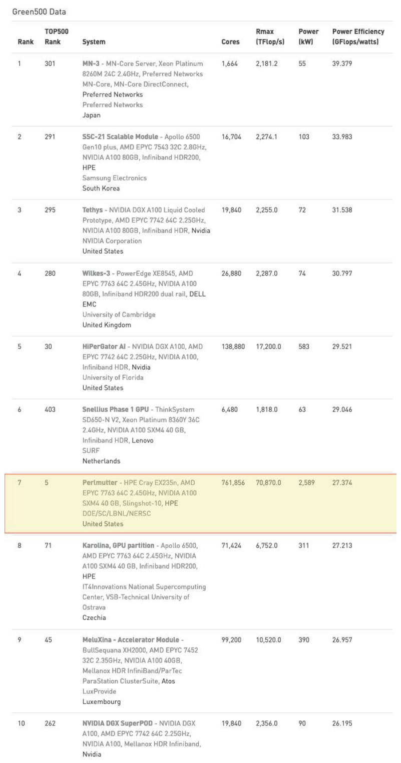
Perlmutter's SC21 Top500 Wins!

#5 overall (HPL) 70.87PF



Rank	System	Cores	Rmax (TFlop/s)	Rpeak (TFlop/s)	Power (kW)
1	**Supercomputer Fugaku** - Supercomputer Fugaku, A64FX 48C 2.2GHz, Tofu interconnect D, Fujitsu RIKEN Center for Computational Science Japan	7,630,848	442,010.0	537,212.0	29,899
2	**Summit** - IBM Power System AC922, IBM POWER9 22C 3.07GHz, NVIDIA Volta GV100, Dual-rail Mellanox EDR Infiniband, IBM DOE/SC/Oak Ridge National Laboratory United States	2,414,592	148,600.0	200,794.9	10,096
3	**Sierra** - IBM Power System AC922, IBM POWER9 22C 3.1GHz, NVIDIA Volta GV100, Dual-rail Mellanox EDR Infiniband, IBM / NVIDIA / Mellanox DOE/NNSA/LLNL United States	1,572,480	94,640.0	125,712.0	7,438
4	**Sunway TaihuLight** - Sunway MPP, Sunway SW26010 260C 1.45GHz, Sunway, NRCPC National Supercomputing Center in Wuxi China	10,649,600	93,014.6	125,435.9	15,371
5	**Perlmutter** - HPE Cray EX235n, AMD EPYC 7763 64C 2.45GHz, NVIDIA A100 SXM4 40 GB, Slingshot-10, HPE DOE/SC/LBNL/NERSC United States	761,856	70,870.0	93,750.0	2,589
6	**Selene** - NVIDIA DGX A100, AMD EPYC 7742 64C 2.25GHz, NVIDIA A100, Mellanox HDR Infiniband, Nvidia NVIDIA Corporation United States	555,520	63,460.0	79,215.0	2,646
7	**Tianhe-2A** - TH-IVB-FEP Cluster, Intel Xeon E5-2692v2 12C 2.2GHz, TH Express-2, Matrix-2000, NUDT National Super Computer Center in Guangzhou China	4,981,760	61,444.5	100,678.7	18,482
8	**JUWELS Booster Module** - Bull Sequana XH2000 , AMD EPYC 7402 24C 2.8GHz, NVIDIA A100, Mellanox HDR InfiniBand/ParTec ParaStation ClusterSuite, Atos Forschungszentrum Juelich (FZJ) Germany	449,280	44,120.0	70,980.0	1,764
9	**HPC5** - PowerEdge C4140, Xeon Gold 6252 24C 2.1GHz, NVIDIA Tesla V100, Mellanox HDR Infiniband, DELL EMC Eni S.p.A. Italy	669,760	35,450.0	51,720.8	2,252
10	**Voyager-EUS2** - ND96amsr_A100_v4, AMD EPYC 7V12 48C 2.45GHz, NVIDIA A100 80GB, Mellanox HDR Infiniband, Microsoft Azure Azure East US 2 United States	253,440	30,050.0	39,531.2	

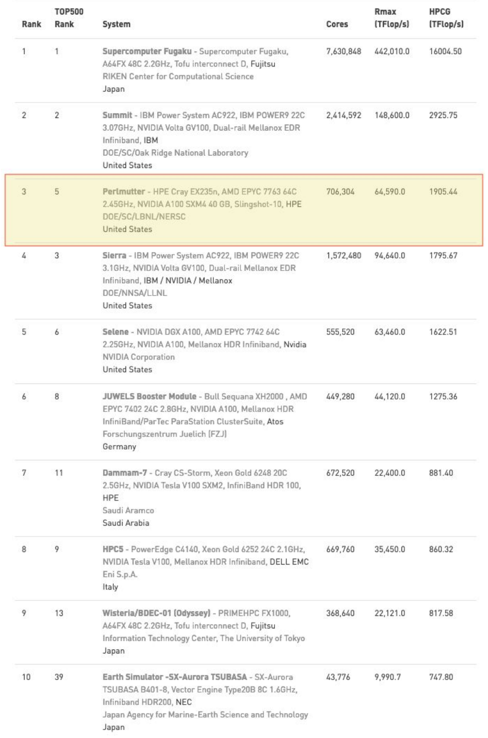
#7 Green 27.37GF/W



Green500 Data

Rank	TOP500 Rank	System	Cores	Rmax (TFlop/s)	Power (kW)	Power Efficiency (GFlops/watts)
1	301	**MN-3** - MN-Core Server, Xeon Platinum 8260M 24C 2.4GHz, Preferred Networks MN-Core, MN-Core DirectConnect, Preferred Networks Preferred Networks Japan	1,664	2,181.2	55	39.379
2	291	**SSC-21 Scalable Module** - Apollo 6500 Gen10 plus, AMD EPYC 7543 32C 2.8GHz, NVIDIA A100 80GB, Infiniband HDR200, HPE Samsung Electronics South Korea	16,704	2,274.1	103	33.983
3	295	**Tethys** - NVIDIA DGX A100 Liquid Cooled Prototype, AMD EPYC 7742 64C 2.25GHz, NVIDIA A100 80GB, Infiniband HDR, Nvidia NVIDIA Corporation United States	19,840	2,255.0	72	31.538
4	280	**Wilkes-3** - PowerEdge XE8545, AMD EPYC 7763 64C 2.45GHz, NVIDIA A100 80GB, Infiniband HDR200 dual rail, DELL EMC University of Cambridge United Kingdom	26,880	2,287.0	74	30.797
5	30	**HiPerGator AI** - NVIDIA DGX A100, AMD EPYC 7742 64C 2.25GHz, NVIDIA A100, Infiniband HDR, Nvidia University of Florida United States	138,880	17,200.0	583	29.521
6	403	**Snellius Phase 1 GPU** - ThinkSystem SD650-N V2, Xeon Platinum 8360Y 36C 2.4GHz, NVIDIA A100 SXM4 40 GB, Infiniband HDR, Lenovo SURF Netherlands	6,480	1,818.0	63	29.046
7	5	**Perlmutter** - HPE Cray EX235n, AMD EPYC 7763 64C 2.45GHz, NVIDIA A100 SXM4 40 GB, Slingshot-10, HPE DOE/SC/LBNL/NERSC United States	761,856	70,870.0	2,589	27.374
8	71	**Karolina, GPU partition** - Apollo 6500, AMD EPYC 7763 64C 2.45GHz, NVIDIA A100 SXM4 40 GB, Infiniband HDR200, HPE IT4Innovations National Supercomputing Center, VSB-Technical University of Ostrava Czechia	71,424	6,752.0	311	27.213
9	45	**MeluXina - Accelerator Module** - BullSequana XH2000, AMD EPYC 7452 32C 2.35GHz, NVIDIA A100 40GB, Mellanox HDR InfiniBand/ParTec ParaStation ClusterSuite, Atos LuxProvide Luxembourg	99,200	10,520.0	390	26.957
10	262	**NVIDIA DGX SuperPOD** - NVIDIA DGX A100, AMD EPYC 7742 64C 2.25GHz, NVIDIA A100, Mellanox HDR Infiniband, Nvidia	19,840	2,356.0	90	26.195

#3 HPCG 1.9PF



Rank	TOP500 Rank	System	Cores	Rmax (TFlop/s)	HPCG (TFlop/s)
1	1	**Supercomputer Fugaku** - Supercomputer Fugaku, A64FX 48C 2.2GHz, Tofu interconnect D, Fujitsu RIKEN Center for Computational Science Japan	7,630,848	442,010.0	16004.50
2	2	**Summit** - IBM Power System AC922, IBM POWER9 22C 3.07GHz, NVIDIA Volta GV100, Dual-rail Mellanox EDR Infiniband, IBM DOE/SC/Oak Ridge National Laboratory United States	2,414,592	148,600.0	2925.75
3	5	**Perlmutter** - HPE Cray EX235n, AMD EPYC 7763 64C 2.45GHz, NVIDIA A100 SXM4 40 GB, Slingshot-10, HPE DOE/SC/LBNL/NERSC United States	706,304	64,590.0	1905.44
4	3	**Sierra** - IBM Power System AC922, IBM POWER9 22C 3.1GHz, NVIDIA Volta GV100, Dual-rail Mellanox EDR Infiniband, IBM / NVIDIA / Mellanox DOE/NNSA/LLNL United States	1,572,480	94,640.0	1795.67
5	6	**Selene** - NVIDIA DGX A100, AMD EPYC 7742 64C 2.25GHz, NVIDIA A100, Mellanox HDR Infiniband, Nvidia NVIDIA Corporation United States	555,520	63,460.0	1622.51
6	8	**JUWELS Booster Module** - Bull Sequana XH2000 , AMD EPYC 7402 24C 2.8GHz, NVIDIA A100, Mellanox HDR InfiniBand/ParTec ParaStation ClusterSuite, Atos Forschungszentrum Juelich (FZJ) Germany	449,280	44,120.0	1275.36
7	11	**Dammam-7** - Cray CS-Storm, Xeon Gold 6248 20C 2.5GHz, NVIDIA Tesla V100 SXM2, InfiniBand HDR 100, HPE Saudi Aramco Saudi Arabia	672,520	22,400.0	881.40
8	9	**HPC5** - PowerEdge C4140, Xeon Gold 6252 24C 2.1GHz, NVIDIA Tesla V100, Mellanox HDR Infiniband, DELL EMC Eni S.p.A. Italy	669,760	35,450.0	860.32
9	13	**Wisteria/BDEC-01 (Odyssey)** - PRIMEHPC FX1000, A64FX 48C 2.2GHz, Tofu interconnect D, Fujitsu Information Technology Center, The University of Tokyo Japan	368,640	22,121.0	817.58
10	39	**Earth Simulator -SX-Aurora TSUBASA** - SX-Aurora TSUBASA B401-8, Vector Engine Type20B 8C 1.6GHz, Infiniband HDR200, NEC Japan Agency for Marine-Earth Science and Technology Japan	43,776	9,990.7	747.80

Today I learned

What surprised you that might benefit other users to hear about?

(and might help NERSC identify documentation improvements!)

Eg:

- Something you got stuck on, hit a dead end, or turned out to be wrong about
  - Give others the benefit of your experience!
  - Opportunity to improve NERSC documentation
- A tip for using NERSC
- Something you learned that might benefit other NERSC users

"If we knew what it was we were doing, it would not be called research, would it?" - Einstein

Announcements and CFPs

See weekly email for these and more:

- Annual User Survey is now open!
  - Look for an email from NERSC@nbriresearch.com
- In related news: you can see the NERSC 2020 Annual Report online (one thing the survey contributes to). Also the 2020 User Demographics
- Heads up: User information is transmitted to DOE Office of Science at end of year
  - Includes name, institutional affiliation(s), and project title(s)
- **New default python module coming in January**

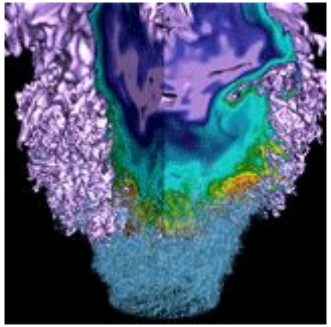
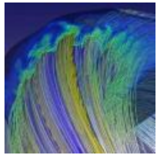
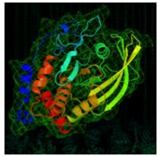
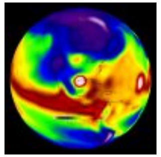
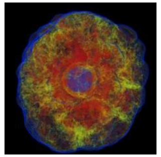
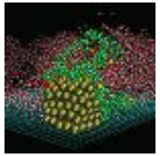
Perlmutter Announcements

- **Breaking news**: Perlmutter User Training originally scheduled for early December ~~will most likely be~~ **has been rescheduled** to January 5-7, 2022
- Users with GPU-ready workloads can request access to Perlmutter by filling out this form
- Prepare your dotfiles for Perlmutter!
  - $HOME is shared across both systems, but each system has its own modules (and module system), etc
  - Check $NERSC_HOST before making system-specific settings in your .bashrc / .cshrc

Announcements and CFPs

- Apply for Prestigious **Alvarez** & **Hopper Postdoctoral Fellowships** in Computing Sciences at Berkeley Lab & NERSC **by Next Monday, November 22**
- Applications for DOE Computational Science Graduate Fellowship are now open
  - For first- and second- year PhD students
- Call for Proposals: Quantum Information Science on Perlmutter
- Nominate someone for the James Corones Award in Leadership, Community Building & Communication
  - Mid-career scientist or engineer making an impact in leadership, community building, and scientific communication
- Others?

Spack at NERSC

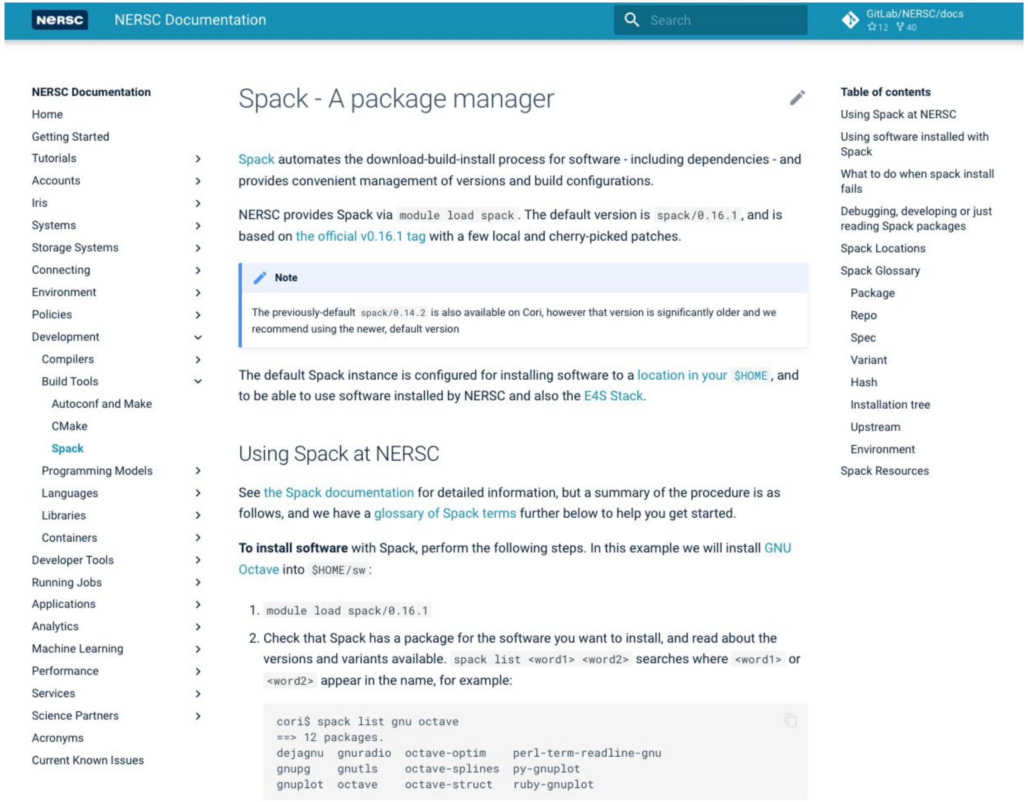
What is it?

"Supercomputing Package Manager"

Automates software installation

Tip: **All of today's topic is in our docs**! =>

(and the docs have links to other Spack information)



Spack - A package manager

Spack automates the download-build-install process for software - including dependencies - and provides convenient management of versions and build configurations.

NERSC provides Spack via `module load spack`. The default version is `spack/0.16.1`, and is based on the official v0.16.1 tag with a few local and cherry-picked patches.

Note

The previously-default `spack/0.14.2` is also available on Cori, however that version is significantly older and we recommend using the newer, default version

The default Spack instance is configured for installing software to a location in your `$HOME`, and to be able to use software installed by NERSC and also the E4S Stack.

Using Spack at NERSC

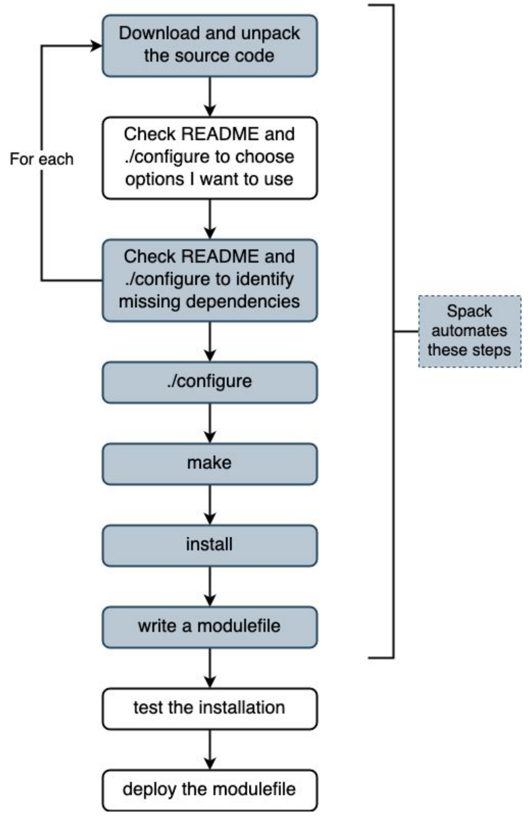
See the Spack documentation for detailed information, but a summary of the procedure is as follows, and we have a glossary of Spack terms further below to help you get started.

To install software with Spack, perform the following steps. In this example we will install GNU Octave into `$HOME/sw`:

1. `module load spack/0.16.1`
2. Check that Spack has a package for the software you want to install, and read about the versions and variants available. `spack list <word1> <word2>` searches where `<word1>` or `<word2>` appear in the name, for example:

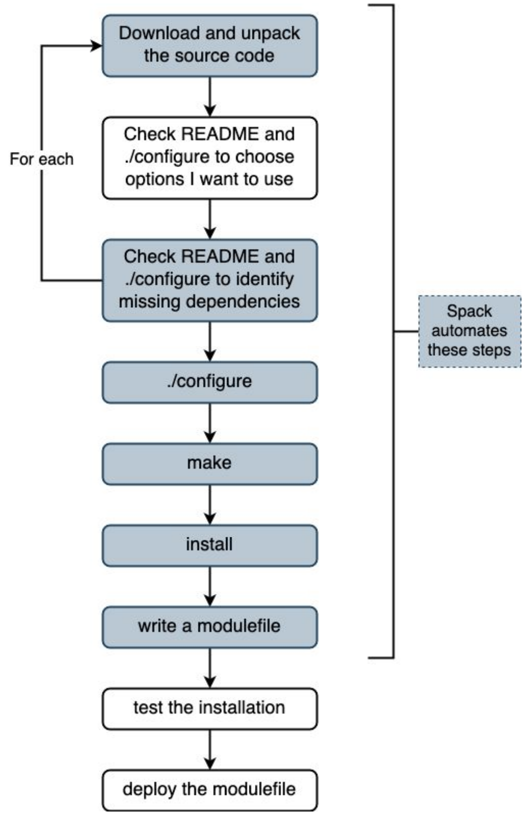
```
cori$ spack list gnu octave
==> 12 packages.
dejagnu  gnuradio  octave-optim    perl-term-readline-gnu
gnupg    gnutls    octave-splines  py-gnuplot
gnuplot  octave    octave-struct   ruby-gnuplot
```

Why use it?


Download and unpack the source code
Check README and ./configure to choose options I want to use
Check README and ./configure to identify missing dependencies
For each
./configure
make
install
write a modulefile
test the installation
deploy the modulefile
Spack automates these steps

- Building software is laborious
  - And error prone
- Spack automates a lot of the busy-work
  - Including the details of getting the right invocation to build the software with the options you want
- Recommendation: Use Spack as the first option for installing software

Why not use it?


Download and unpack the source code
Check README and ./configure to choose options I want to use
Check README and ./configure to identify missing dependencies
For each
./configure
make
install
write a modulefile
test the installation
deploy the modulefile
Spack automates these steps

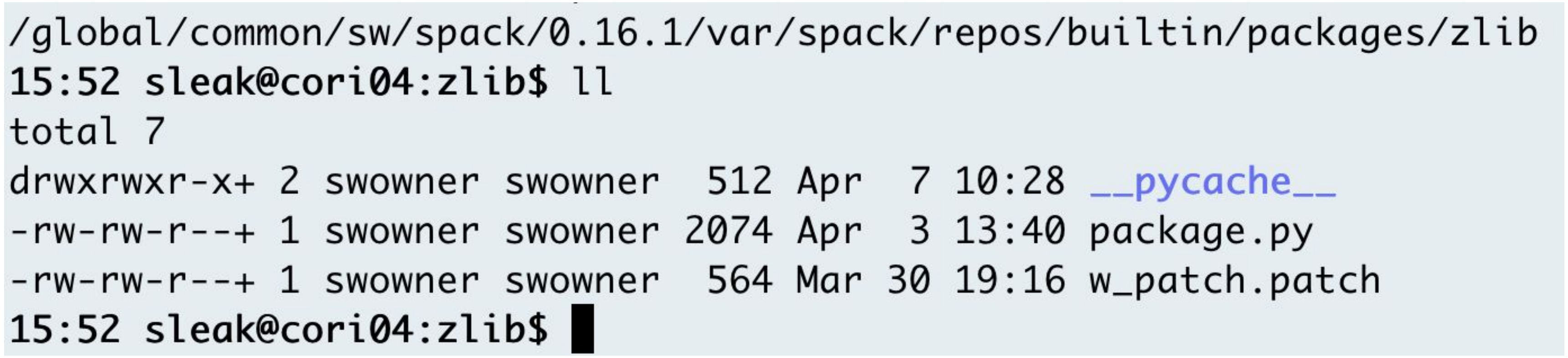
- Scientific software is complex
- Automation is complex
- .. complex^2 !
- The details of what Spack did, and why, are often opaque
- When something fails, finding why (and fixing it) is usually absurdly difficult
- Recommendation: If the fix isn't easy, stop digging
  - Move to a different build method

Today's goal: Demystify Spack and describe NERSC's Spack setup

- Jargon dictionary
- Essential Spack commands
- ~~Working with environments~~
- ~~How Spack decides what to install~~
- ~~Spack idiosyncrasies~~
- ~~Spack setup at NERSC~~
- ~~Workflow for installing software with Spack~~
- ~~What to do when it doesn't work~~
- Q&A

Spack words, and what they mean

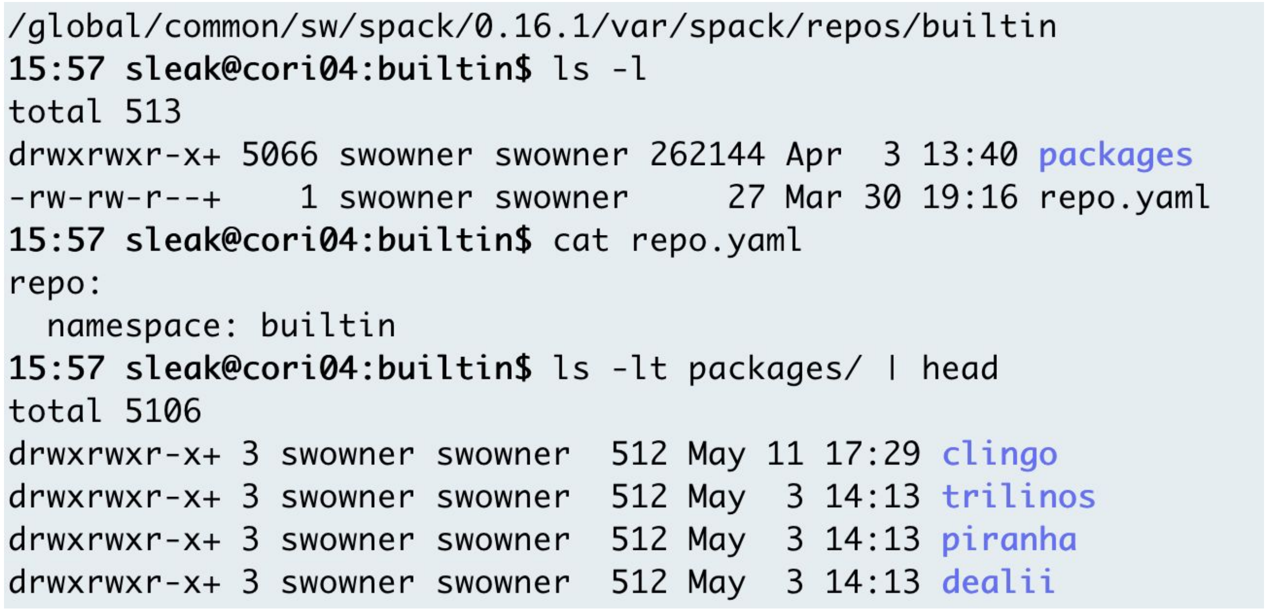
Package: "Source code" describing a piece of software and how to build it (actually a Python class), along with any patches etc that might need to be applied first



```
/global/common/sw/spack/0.16.1/var/spack/repos/builtin/packages/zlib
15:52 sleak@cori04:zlib$ ll
total 7
drwxrwxr-x+ 2 swowner swowner  512 Apr  7 10:28 __pycache__
-rw-rw-r--+ 1 swowner swowner 2074 Apr  3 13:40 package.py
-rw-rw-r--+ 1 swowner swowner  564 Mar 30 19:16 w_patch.patch
15:52 sleak@cori04:zlib$ 
```

Spack words, and what they mean

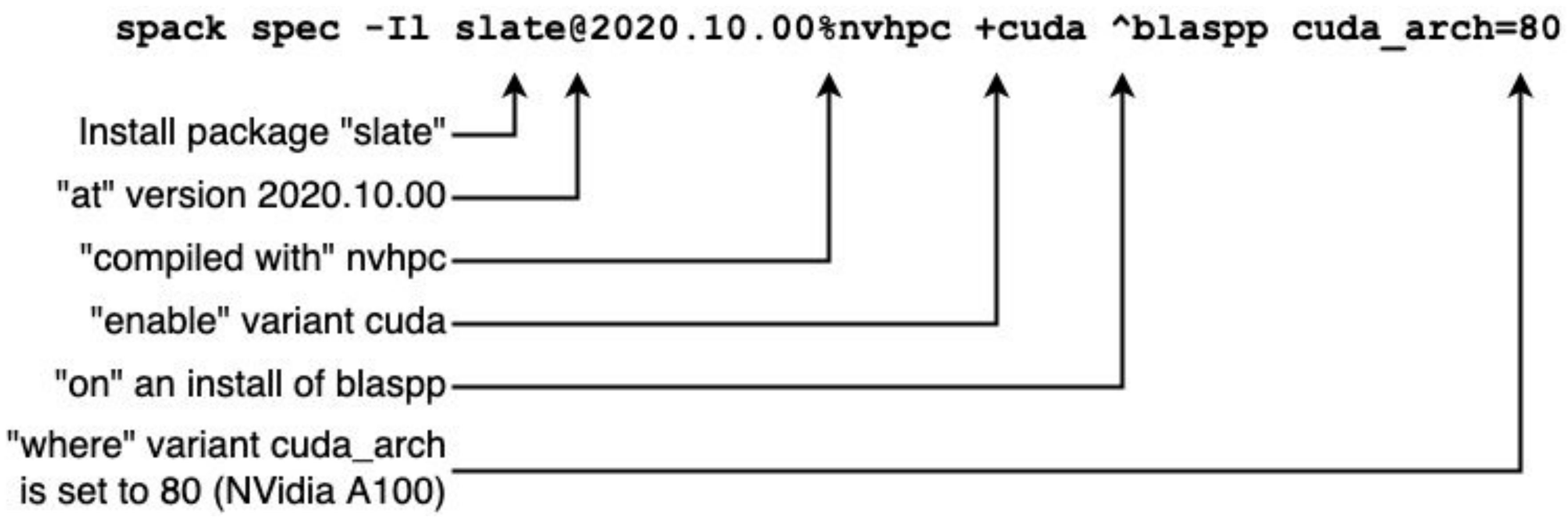
Repo: A collection ("repository") of packages. Pretty much everything is in the "builtin" repo, but Spack has a "repos" config section where you can specify locations and order of repos to search



```
/global/common/sw/spack/0.16.1/var/spack/repos/builtin
15:57 sleak@cori04:builtin$ ls -l
total 513
drwxrwxr-x+ 5066 swowner swowner 262144 Apr  3 13:40 packages
-rw-rw-r--+    1 swowner swowner     27 Mar 30 19:16 repo.yaml
15:57 sleak@cori04:builtin$ cat repo.yaml
repo:
  namespace: builtin
15:57 sleak@cori04:builtin$ ls -lt packages/ | head
total 5106
drwxrwxr-x+ 3 swowner swowner  512 May 11 17:29 clingo
drwxrwxr-x+ 3 swowner swowner  512 May  3 14:13 trilinos
drwxrwxr-x+ 3 swowner swowner  512 May  3 14:13 piranha
drwxrwxr-x+ 3 swowner swowner  512 May  3 14:13 dealii
```

Spack words, and what they mean

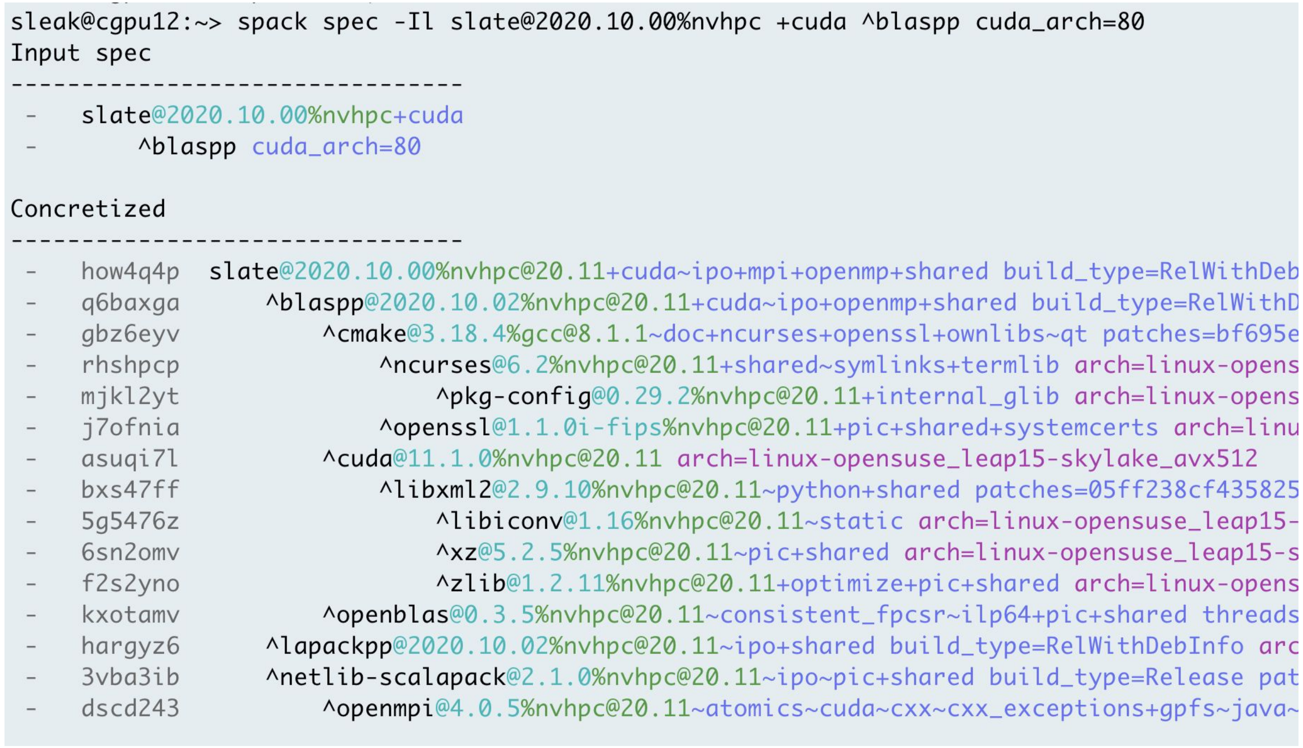
Spec: Spack has a DSL for declaratively describing the parameters with which a package should be (or was) built



```
spack spec -Il slate@2020.10.00%nvhpc +cuda ^blaspp cuda_arch=80
```

Spack words, and what they mean

Spec: (cont'd)
Given a partial spec, and defaults from packages and Spack configuration, Spack uses an ASP solver to work out a detailed ("concrete") spec

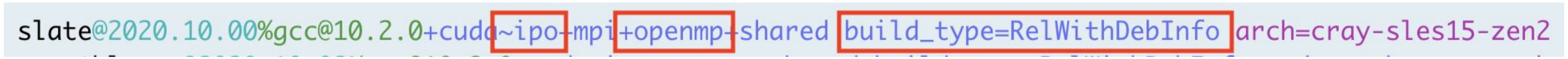


```
sleak@cgpu12:~> spack spec -Il slate@2020.10.00%nvhpc +cuda ^blaspp cuda_arch=80
Input spec
--------------------------------
 -   slate@2020.10.00%nvhpc+cuda
 -       ^blaspp cuda_arch=80

Concretized
--------------------------------
 -   how4q4p  slate@2020.10.00%nvhpc@20.11+cuda~ipo+mpi+openmp+shared build_type=RelWithDeb
 -   q6baxga      ^blaspp@2020.10.02%nvhpc@20.11+cuda~ipo+openmp+shared build_type=RelWithD
 -   gbz6eyv          ^cmake@3.18.4%gcc@8.1.1~doc+ncurses+openssl+ownlibs~qt patches=bf695e
 -   rhshpcp              ^ncurses@6.2%nvhpc@20.11+shared~symlinks+termlib arch=linux-opens
 -   mjkl2yt                  ^pkg-config@0.29.2%nvhpc@20.11+internal_glib arch=linux-opens
 -   j7ofnia              ^openssl@1.1.0i-fips%nvhpc@20.11+pic+shared+systemcerts arch=linu
 -   asuqi7l          ^cuda@11.1.0%nvhpc@20.11 arch=linux-opensuse_leap15-skylake_avx512
 -   bxs47ff              ^libxml2@2.9.10%nvhpc@20.11~python+shared patches=05ff238cf435825
 -   5g5476z                  ^libiconv@1.16%nvhpc@20.11~static arch=linux-opensuse_leap15-
 -   6sn2omv                  ^xz@5.2.5%nvhpc@20.11~pic+shared arch=linux-opensuse_leap15-s
 -   f2s2yno                  ^zlib@1.2.11%nvhpc@20.11+optimize+pic+shared arch=linux-opens
 -   kxotamv          ^openblas@0.3.5%nvhpc@20.11~consistent_fpcsr~ilp64+pic+shared threads
 -   hargyz6      ^lapackpp@2020.10.02%nvhpc@20.11~ipo+shared build_type=RelWithDebInfo arc
 -   3vba3ib      ^netlib-scalapack@2.1.0%nvhpc@20.11~ipo~pic+shared build_type=Release pat
 -   dscd243          ^openmpi@4.0.5%nvhpc@20.11~atomics~cuda~cxx~cxx_exceptions+gpfs~java~
```

Spack words, and what they mean

Variant: A selectable build option for a package (defined in Spack in the package definition). Usually corresponds to a **`./configure`** option or an optional dependency.

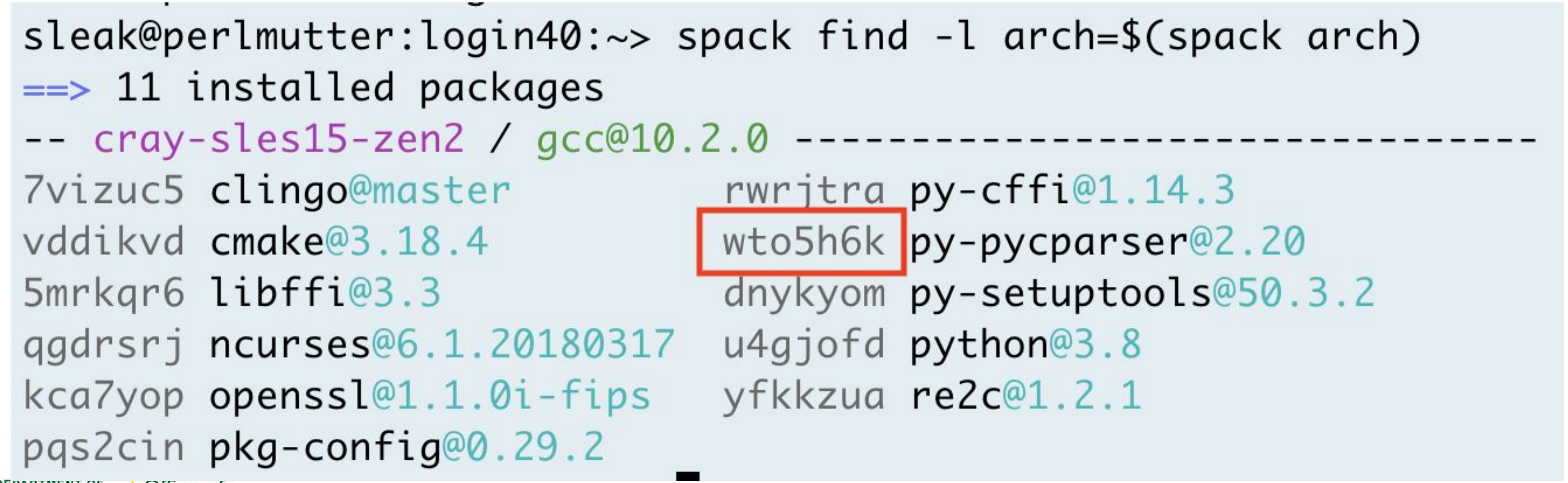
You can enable (+), disable (- or ~) or set (name=value) a variant



```
slate@2020.10.00%gcc@10.2.0+cuda~ipo+mpi+openmp+shared build_type=RelWithDebInfo arch=cray-sles15-zen2
```

Spack words, and what they mean

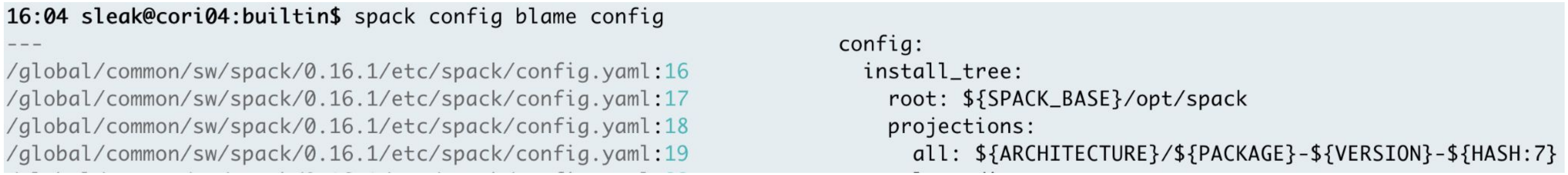
Hash: Spack calculates a hash of each full concrete spec (including dependencies), and uses it as a key to identify the install. This turns out very handy for distinguishing between the many possible builds of the same software



```
sleak@perlmutter:login40:~> spack find -l arch=$(spack arch)
==> 11 installed packages
-- cray-sles15-zen2 / gcc@10.2.0 --------------------------------
7vizuc5 clingo@master               rwrjtra py-cffi@1.14.3
vddikvd cmake@3.18.4                wto5h6k py-pycparser@2.20
5mrkqr6 libffi@3.3                  dnykyom py-setuptools@50.3.2
qgdrsrj ncurses@6.1.20180317        u4gjofd python@3.8
kca7yop openssl@1.1.0i-fips         yfkkzua re2c@1.2.1
pqs2cin pkg-config@0.29.2
```

Spack words, and what they mean

Install_tree: The location Spack will install stuff in. The location and the directory-naming convention are defined in a config file



```
16:04 sleak@cori04:builtin$ spack config blame config
---                                                              config:
/global/common/sw/spack/0.16.1/etc/spack/config.yaml:16            install_tree:
/global/common/sw/spack/0.16.1/etc/spack/config.yaml:17              root: ${SPACK_BASE}/opt/spack
/global/common/sw/spack/0.16.1/etc/spack/config.yaml:18              projections:
/global/common/sw/spack/0.16.1/etc/spack/config.yaml:19                all: ${ARCHITECTURE}/${PACKAGE}-${VERSION}-${HASH:7}
```

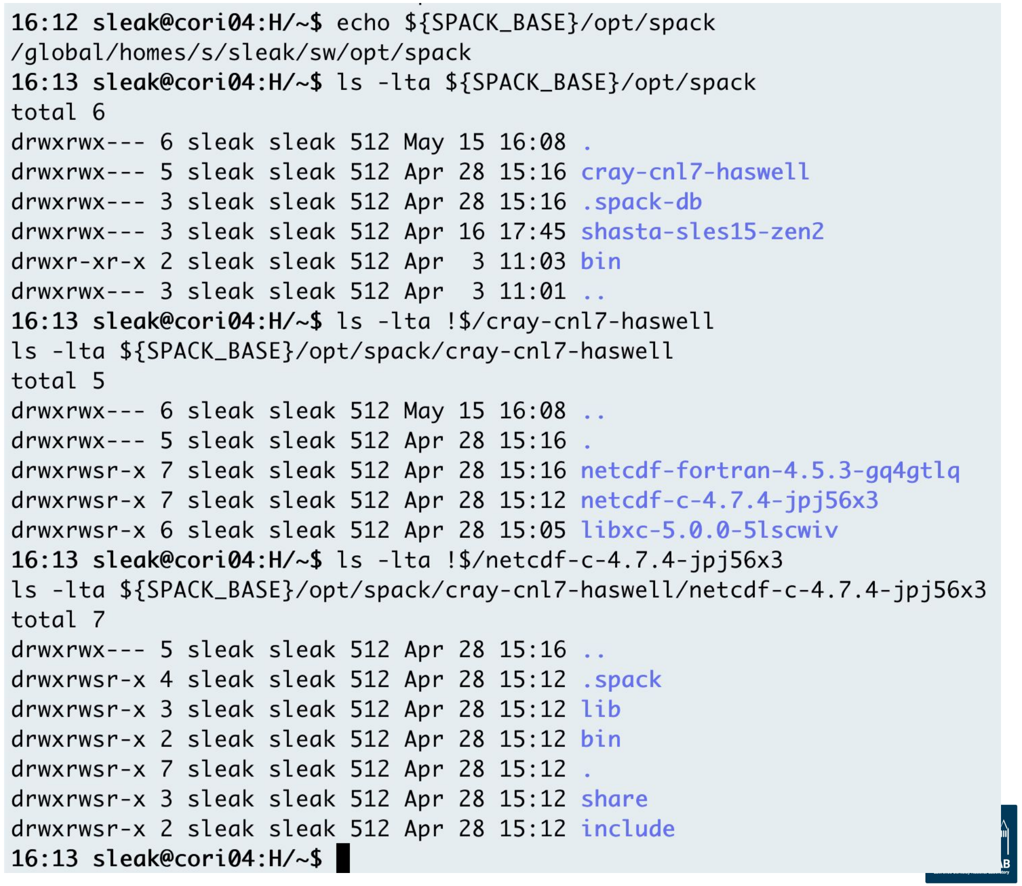
Tip: You can override the location for $SPACK_BASE by setting it in your .bashrc or .bash_profile (default is $HOME/sw)

Spack words, and what they mean

Install_tree: (cont'd)

NERSC setup puts the install_tree in the user's **`$HOME`**, and organizes installs by target architecture

Spack adds some indexing information to each install, and caches it in **`.spack_db/`** at the install_tree root



```
16:12 sleak@cori04:H/~$ echo ${SPACK_BASE}/opt/spack
/global/homes/s/sleak/sw/opt/spack
16:13 sleak@cori04:H/~$ ls -lta ${SPACK_BASE}/opt/spack
total 6
drwxrwx--- 6 sleak sleak 512 May 15 16:08 .
drwxrwx--- 5 sleak sleak 512 Apr 28 15:16 cray-cnl7-haswell
drwxrwx--- 3 sleak sleak 512 Apr 28 15:16 .spack-db
drwxrwx--- 3 sleak sleak 512 Apr 16 17:45 shasta-sles15-zen2
drwxr-xr-x 2 sleak sleak 512 Apr  3 11:03 bin
drwxrwx--- 3 sleak sleak 512 Apr  3 11:01 ..
16:13 sleak@cori04:H/~$ ls -lta !$/cray-cnl7-haswell
ls -lta ${SPACK_BASE}/opt/spack/cray-cnl7-haswell
total 5
drwxrwx--- 6 sleak sleak 512 May 15 16:08 ..
drwxrwx--- 5 sleak sleak 512 Apr 28 15:16 .
drwxrwsr-x 7 sleak sleak 512 Apr 28 15:16 netcdf-fortran-4.5.3-gq4gtlq
drwxrwsr-x 7 sleak sleak 512 Apr 28 15:12 netcdf-c-4.7.4-jpj56x3
drwxrwsr-x 6 sleak sleak 512 Apr 28 15:05 libxc-5.0.0-5lscwiv
16:13 sleak@cori04:H/~$ ls -lta !$/netcdf-c-4.7.4-jpj56x3
ls -lta ${SPACK_BASE}/opt/spack/cray-cnl7-haswell/netcdf-c-4.7.4-jpj56x3
total 7
drwxrwx--- 5 sleak sleak 512 Apr 28 15:16 ..
drwxrwsr-x 4 sleak sleak 512 Apr 28 15:12 .spack
drwxrwsr-x 3 sleak sleak 512 Apr 28 15:12 lib
drwxrwsr-x 2 sleak sleak 512 Apr 28 15:12 bin
drwxrwsr-x 7 sleak sleak 512 Apr 28 15:12 .
drwxrwsr-x 3 sleak sleak 512 Apr 28 15:12 share
drwxrwsr-x 2 sleak sleak 512 Apr 28 15:12 include
16:13 sleak@cori04:H/~$
```

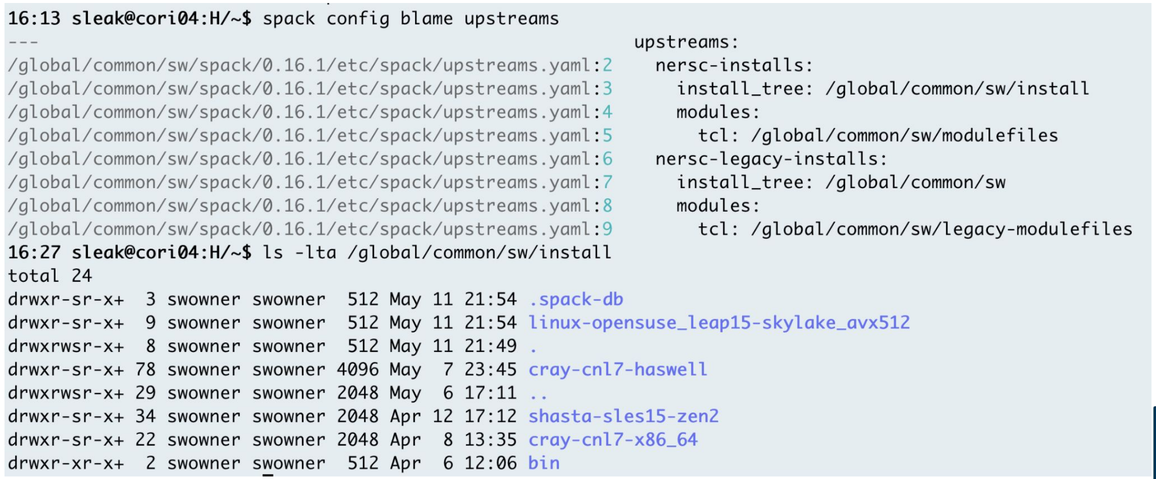
U.S. DEPARTMENT OF ENERGY | Office of Science

Spack words, and what they mean

Upstream: Another install_tree (but read-only) that Spack is allowed to use. Eg if "netcdf" requires "hdf5", and "hdf5" is installed upstream, Spack does not need to build "hdf5" in order to build "netcdf", it can use the upstream install.

NERSC config has an upstream in **`/global/common/sw/install`**, so users can build on software that we (via swowner) install

Tip: one upstream is the E4S deployment at NERSC

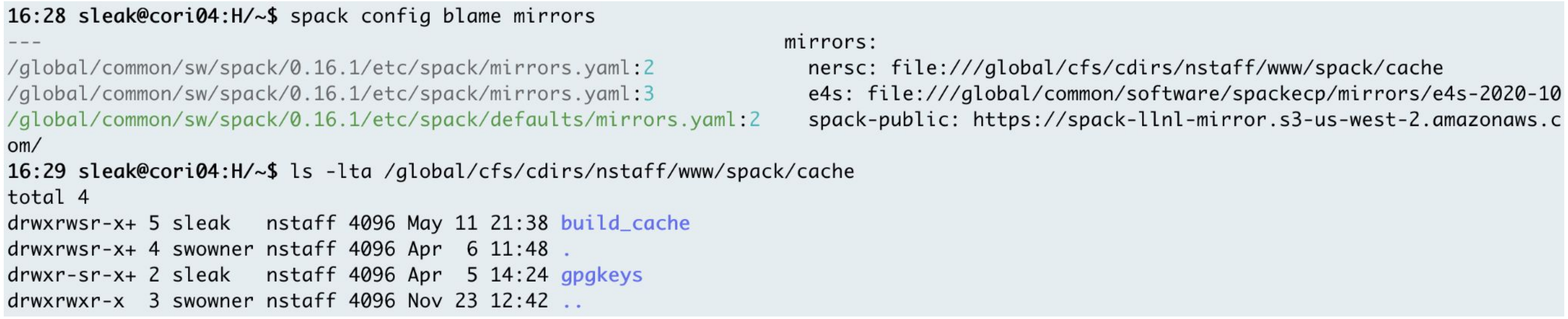


```
16:13 sleak@cori04:H/~$ spack config blame upstreams
---                                                        upstreams:
/global/common/sw/spack/0.16.1/etc/spack/upstreams.yaml:2    nersc-installs:
/global/common/sw/spack/0.16.1/etc/spack/upstreams.yaml:3      install_tree: /global/common/sw/install
/global/common/sw/spack/0.16.1/etc/spack/upstreams.yaml:4      modules:
/global/common/sw/spack/0.16.1/etc/spack/upstreams.yaml:5        tcl: /global/common/sw/modulefiles
/global/common/sw/spack/0.16.1/etc/spack/upstreams.yaml:6    nersc-legacy-installs:
/global/common/sw/spack/0.16.1/etc/spack/upstreams.yaml:7      install_tree: /global/common/sw
/global/common/sw/spack/0.16.1/etc/spack/upstreams.yaml:8      modules:
/global/common/sw/spack/0.16.1/etc/spack/upstreams.yaml:9        tcl: /global/common/sw/legacy-modulefiles
16:27 sleak@cori04:H/~$ ls -lta /global/common/sw/install
total 24
drwxr-sr-x+  3 swowner swowner  512 May 11 21:54 .spack-db
drwxr-sr-x+  9 swowner swowner  512 May 11 21:54 linux-opensuse_leap15-skylake_avx512
drwxrwsr-x+  8 swowner swowner  512 May 11 21:49 .
drwxr-sr-x+ 78 swowner swowner 4096 May  7 23:45 cray-cnl7-haswell
drwxrwsr-x+ 29 swowner swowner 2048 May  6 17:11 ..
drwxr-sr-x+ 34 swowner swowner 2048 Apr 12 17:12 shasta-sles15-zen2
drwxr-sr-x+ 22 swowner swowner 2048 Apr  8 13:35 cray-cnl7-x86_64
drwxr-xr-x+  2 swowner swowner  512 Apr  6 12:06 bin
```

Spack words, and what they mean

Buildcache: An installed software package, tarred up and GPG-signed, allowing someone else to install it without redoing the ./configure and build steps

E4S makes builds available via a buildcache, and NERSC has one set up on CFS (where it can be served via https)



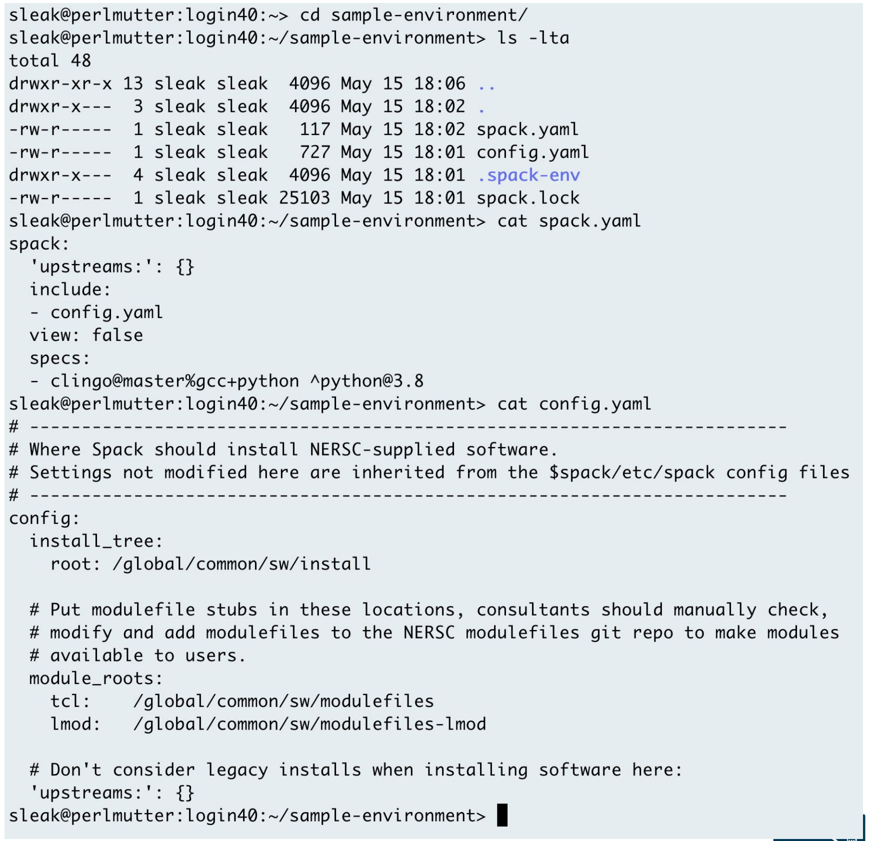
```
16:28 sleak@cori04:H/~$ spack config blame mirrors
---                                                                 mirrors:
/global/common/sw/spack/0.16.1/etc/spack/mirrors.yaml:2               nersc: file:///global/cfs/cdirs/nstaff/www/spack/cache
/global/common/sw/spack/0.16.1/etc/spack/mirrors.yaml:3               e4s: file:///global/common/software/spackecp/mirrors/e4s-2020-10
/global/common/sw/spack/0.16.1/etc/spack/defaults/mirrors.yaml:2      spack-public: https://spack-llnl-mirror.s3-us-west-2.amazonaws.c
om/
16:29 sleak@cori04:H/~$ ls -lta /global/cfs/cdirs/nstaff/www/spack/cache
total 4
drwxrwsr-x+ 5 sleak   nstaff 4096 May 11 21:38 build_cache
drwxrwsr-x+ 4 swowner nstaff 4096 Apr  6 11:48 .
drwxr-sr-x+ 2 sleak   nstaff 4096 Apr  5 14:24 gpgkeys
drwxrwxr-x  3 swowner nstaff 4096 Nov 23 12:42 ..
```

Spack words, and what they mean

Environment: A declarative description of an "environment" (ie collection of software specs alongside build/install preferences) that Spack should make available.

(So, more like a purchase order than a conda environment)

In practice: a directory with a **`spack.yaml`** file in it, in which you can declare local Spack preferences and desired installs



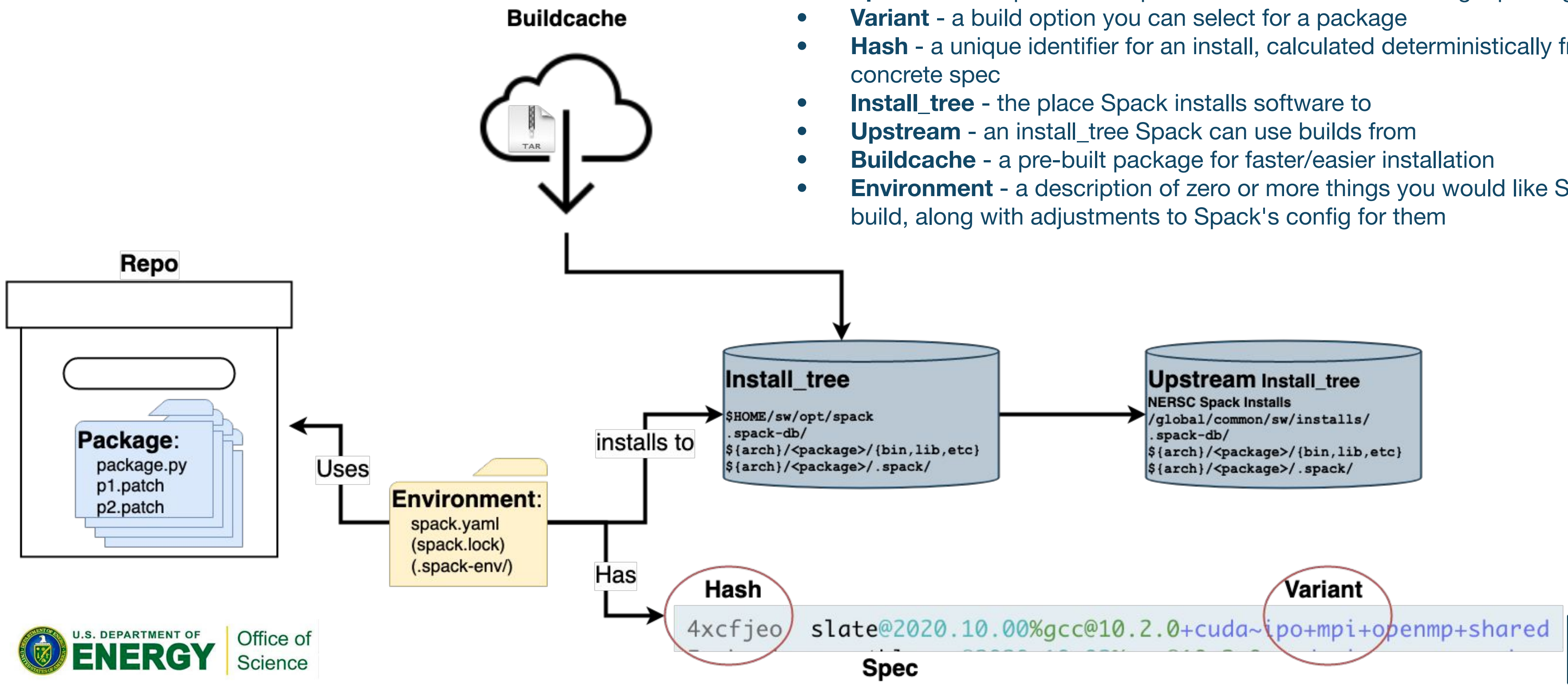
```
sleak@perlmutter:login40:~> cd sample-environment/
sleak@perlmutter:login40:~/sample-environment> ls -lta
total 48
drwxr-xr-x 13 sleak sleak  4096 May 15 18:06 ..
drwxr-x---  3 sleak sleak  4096 May 15 18:02 .
-rw-r-----  1 sleak sleak   117 May 15 18:02 spack.yaml
-rw-r-----  1 sleak sleak   727 May 15 18:01 config.yaml
drwxr-x---  4 sleak sleak  4096 May 15 18:01 .spack-env
-rw-r-----  1 sleak sleak 25103 May 15 18:01 spack.lock
sleak@perlmutter:login40:~/sample-environment> cat spack.yaml
spack:
  'upstreams:': {}
  include:
  - config.yaml
  view: false
  specs:
  - clingo@master%gcc+python ^python@3.8
sleak@perlmutter:login40:~/sample-environment> cat config.yaml
# ---------------------------------------------------------------------------
# Where Spack should install NERSC-supplied software.
# Settings not modified here are inherited from the $spack/etc/spack config files
# ---------------------------------------------------------------------------
config:
  install_tree:
    root: /global/common/sw/install

  # Put modulefile stubs in these locations, consultants should manually check,
  # modify and add modulefiles to the NERSC modulefiles git repo to make modules
  # available to users.
  module_roots:
    tcl:    /global/common/sw/modulefiles
    lmod:   /global/common/sw/modulefiles-lmod

  # Don't consider legacy installs when installing software here:
  'upstreams:': {}
sleak@perlmutter:login40:~/sample-environment> █
```

Summary of Spack jargon

- **Package** - a unit of software that Spack can build and install
- **Repo** - a collection of packages
- **Spec** - a description of the parameters used when building a package
- **Variant** - a build option you can select for a package
- **Hash** - a unique identifier for an install, calculated deterministically from the concrete spec
- **Install_tree** - the place Spack installs software to
- **Upstream** - an install_tree Spack can use builds from
- **Buildcache** - a pre-built package for faster/easier installation
- **Environment** - a description of zero or more things you would like Spack to build, along with adjustments to Spack's config for them

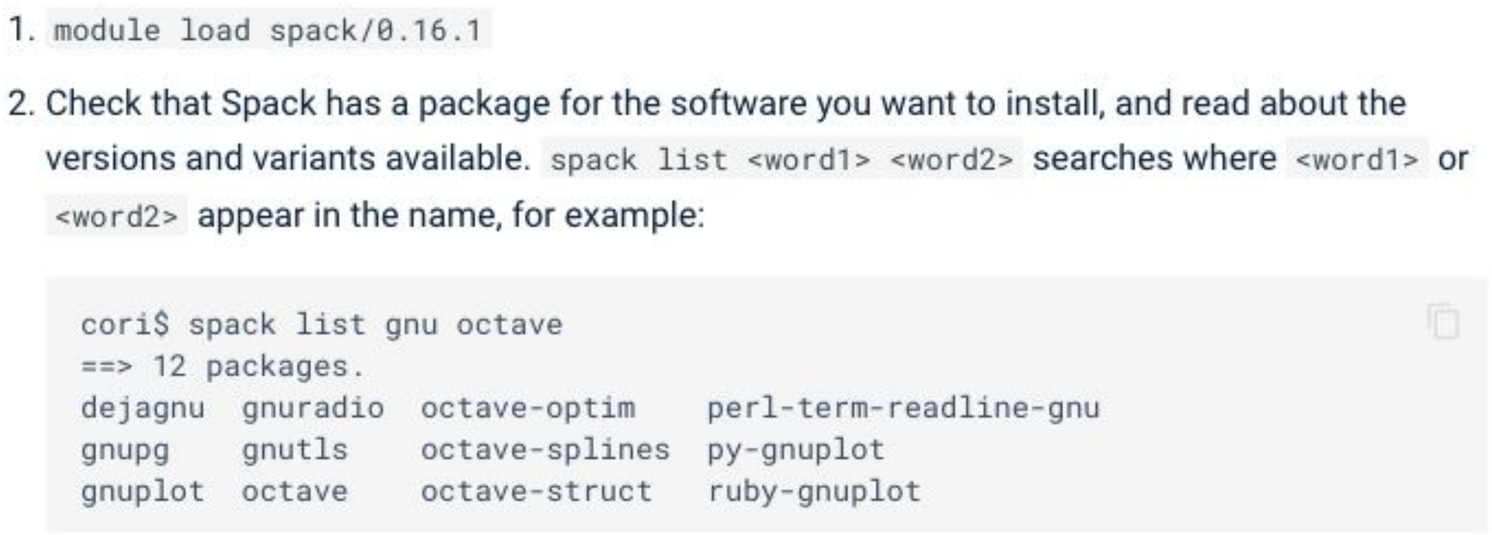
Summary of Spack jargon

- **Package** - a unit of software that Spack can build and install
- **Repo** - a collection of packages
- **Spec** - a description of the parameters used when building a package
- **Variant** - a build option you can select for a package
- **Hash** - a unique identifier for an install, calculated deterministically from the concrete spec
- **Install_tree** - the place Spack installs software to
- **Upstream** - an install_tree Spack can use builds from
- **Buildcache** - a pre-built package for faster/easier installation
- **Environment** - a description of zero or more things you would like Spack to build, along with adjustments to Spack's config for them



Essential commands

- **`spack find`** lists packages installed in the install_tree and upstreams
- **`spack list`** finds packages (installed or not) in repos - i.e. packages that Spack *can* build

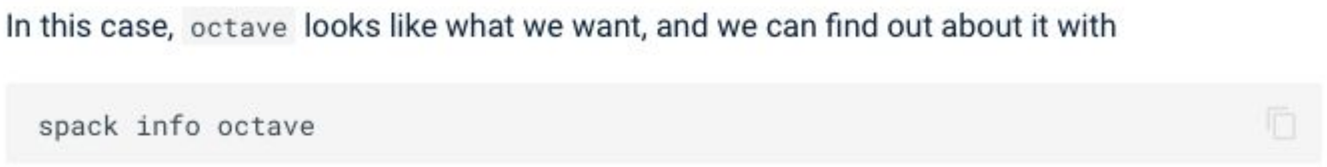


1. `module load spack/0.16.1`
2. Check that Spack has a package for the software you want to install, and read about the versions and variants available. `spack list <word1> <word2>` searches where `<word1>` or `<word2>` appear in the name, for example:

```
cori$ spack list gnu octave
==> 12 packages.
dejagnu  gnuradio  octave-optim    perl-term-readline-gnu
gnupg    gnutls    octave-splines  py-gnuplot
gnuplot  octave    octave-struct   ruby-gnuplot
```

Essential commands

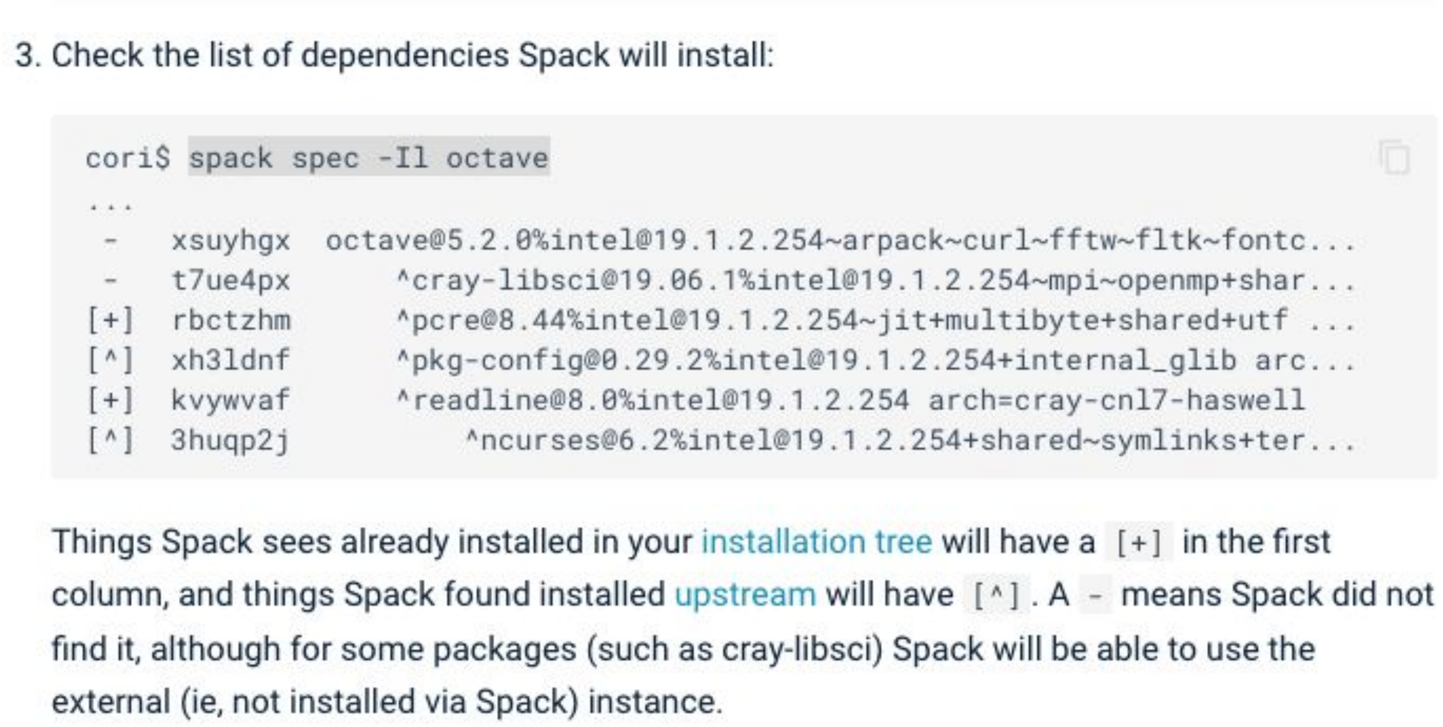
- **`spack info`** shows what Spack knows about a package


In this case, `octave` looks like what we want, and we can find out about it with

```
spack info octave
```

Essential commands

- **`spack spec`** computes and shows a concrete spec (given a partial one)



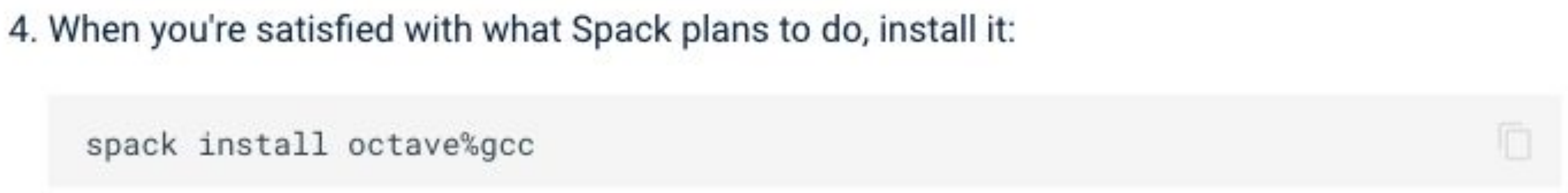
3. Check the list of dependencies Spack will install:

```
cori$ spack spec -Il octave
...
 -   xsuyhgx  octave@5.2.0%intel@19.1.2.254~arpack~curl~fftw~fltk~fontc...
 -   t7ue4px      ^cray-libsci@19.06.1%intel@19.1.2.254~mpi~openmp+shar...
[+]  rbctzhm      ^pcre@8.44%intel@19.1.2.254~jit+multibyte+shared+utf ...
[^]  xh3ldnf      ^pkg-config@0.29.2%intel@19.1.2.254+internal_glib arc...
[+]  kvywvaf      ^readline@8.0%intel@19.1.2.254 arch=cray-cnl7-haswell
[^]  3huqp2j          ^ncurses@6.2%intel@19.1.2.254+shared~symlinks+ter...
```

Things Spack sees already installed in your installation tree will have a `[+]` in the first column, and things Spack found installed upstream will have `[^]`. A `-` means Spack did not find it, although for some packages (such as cray-libsci) Spack will be able to use the external (ie, not installed via Spack) instance.

Essential commands

- **`spack install -v <spec>`** performs the necessary download, configure, build and install steps to install the package and any needed dependencies



4. When you're satisfied with what Spack plans to do, install it:

```
spack install octave%gcc
```

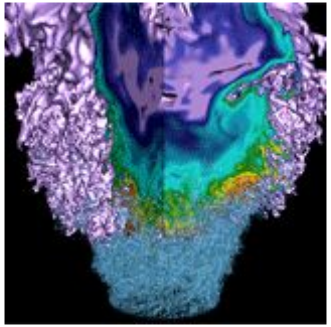
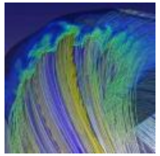
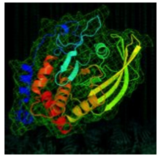
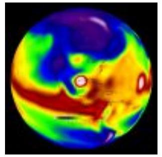
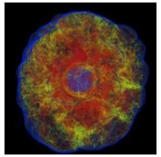
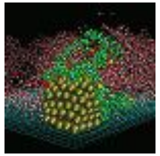
Summary

- When it goes smoothly, Spack saves much time and effort
- (When it doesn't, it can be hard to fix, often best to find an alternative)
- NERSC setup defaults to installing software in your $HOME/sw
  - And can use upstream installs such as E4S

- We ran through some of the jargon you'll encounter related to Spack
- And a basic recipe for using it to install software

One final tip: The Spack Slack (http://spackpm.slack.com/) is a really helpful forum!

Q&A

Coming up

December: AY Transition - what to expect

January: (tentatively) A presentation of some of the work of one of our regular participants

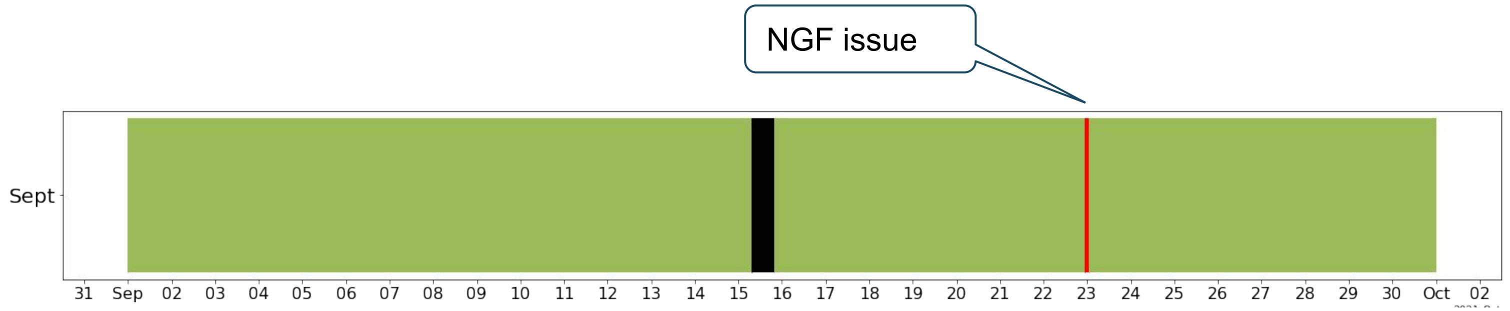
Also coming soon:

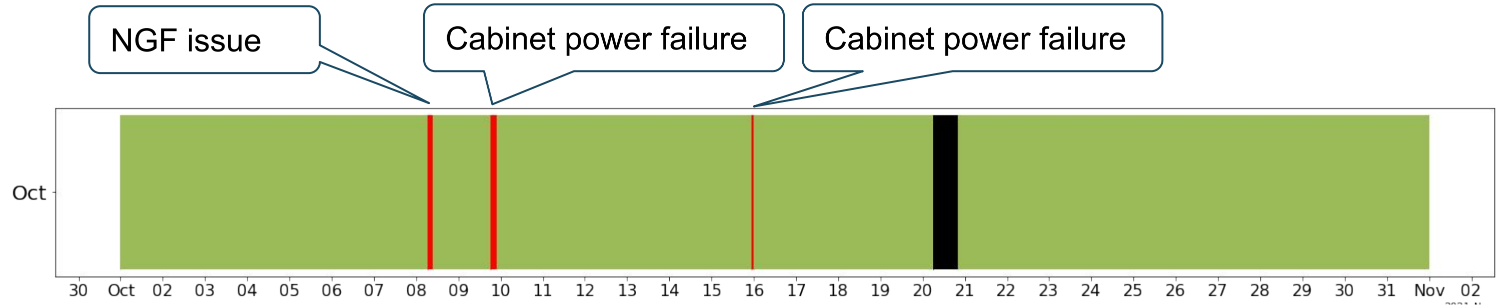
- NERSC docs
- Queue wait time findings (from one of our regular participants)

We'd love to hear some lightning talks **from NERSC users** about the research you use NERSC for!

Last month's numbers - Sept/Oct

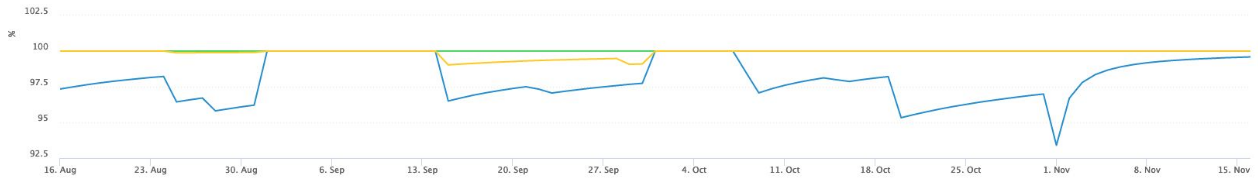
Cori: 6 outages (4 unscheduled totaling 6 hrs 0 min)



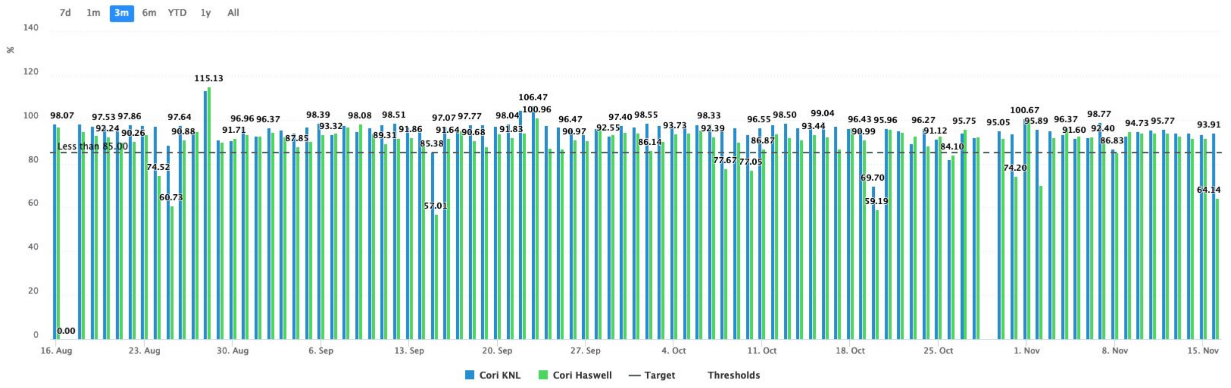


Last month's numbers - Sept/Oct

Cori daily availability:



Cori daily utilization:



New Tickets: 570 (Sept),

Closed Tickets: 490 (Sept), 724 (Oct)

Backlog at 1 Nov: 642

Thank You