Spack Tutorial

Patrick Gartung, Chris Green,
Marc Mengel

Overview

- What is Spack?
- Tutorial (play along!)
  - Setting up, Seeing what is installed
  - Making software available to use
  - Build/Binary Caches
  - Spack and CVMFS
  - Making and using a development instance

What is Spack?

- **S**upercomputing **Pack**age manager
  - system for organizing and building software
  - multiple platforms
  - multiple versions
- Avoids LD_LIBRARY_PATH explosion
  - RPATH in binaries
  - => relocating binary packages (patchelf)
  - Path length / Padding
- Avoiding PATH explosion
  - Environments + Views
  - forest of symlinks

Using Spack – getting started and spack find

- Access Spack via a setup script: (exercise 1)

```
EXP=mu2e
source /cvmfs/$EXP.opensciencegrid.org/packages/setup-env.sh
spack --help
```

- See what packages are installed (exercise 2)

```
spack find
```

- spack find takes a "spec" -- package specification (used all over Spack)

```
spack find python target=x86_64_v2
spack find python target=x86_64
spack find python@3.8:
```

- more spec details

```
spack help --spec
```

Using spack find: cont.

- can show packages unique hash:

```
spack find --long  python
spack find --very-long python
```

- can show dependencies:

```
spack find --deps python
```

- can show paths:

```
spack find --paths python
```

Making software available

- Spack (un)load puts packages in your PATH, etc.

```
python3 -V
spack load python@3.9.16
python3 -V
spack unload python
python3 -V
spack load python/avsjsvp
python3 -V
```

Making Software Available

- Spack “environments” access many packages at once

```
spack find
spack env list
spack env activate ${EXP}_externals_current
spack find
which root
spack env deactivate
```

Using Spack (cont)

- spack cd

```
spack cd -i python@3.9.15
pwd
ls
./bin/python3 -V
```

```
spack cd --env ${EXP}_externals_current
ls -a
cp spack.lock $HOME/saved_spack.lock  # for later…
cd .spack-env/view
ls -l bin | more
```

- Note that there are a bunch of other spack cd options, use --help to see them

Starting a new instance

While you *can* just checkout a copy of spack from Git, and configure it yourself, we recommend our “bootstrap” script from fermi-spack-tools:

```
wget https://github.com/FNALssi/fermi-spack-tools/raw/v2_20_0/bin/bootstrap
chmod +x bootstrap
sh ./bootstrap /path/to/new/instance
```

Or for a build instance

```
Sh ./bootstrap --with_padding /path/to/new/instance
```

Spack recipes

A tour of spack recipes…

Run `spack edit` on the following to examine their recipes:

- ‘watch’ : simple case: Versions, A few dependencies, determine_version (for `spack external find`), executables for recipes that depend on it
- ‘python’: above plus: setting phases, lots of variants, setup_build_environment, setup_dependent_environment, flag_handler, configure_args, @run_after decorators…
- ‘art-root-io’: a representative case for us

SPACK and ups

Our Fermi version of spack has ups compatibility features

```
. /grid/fermiapp/products/common/etc/setups
ups list -az /cvmfs/$EXP.opensciencegrid.org/packages
setup -z /cvmfs/$EXP.opensciencegrid.org/packages \
bzip2 1.0.6
```

You can also convert existing ups packages to spack packages

```
spack load fermi-spack-tools
ups_to_spack htgettoken v1_16_1
```

```
(Run this later in our own Spack area)
```

Your own spack area

- Make a spack instance “chained” to the other ("test release" equivalent)
- we have a script for that...[in fermi-spack-tools]

```
mkdir /build/$USER
cd /build/$USER
spack load fermi-spack-tools
make_subspack --with_padding /cvmfs/$EXP.opensciencegrid.org/packages \
 $PWD/my_spack
```

- The `--with-padding` enables directory padding/relocatability

```
spack unload fermi-spack-tools
. my_spack/setup-env.sh
spack cd -r
more etc/spack/config.yaml etc/spack/upstreams.yaml
```

Our own spack area (cont.)

- We can install something in our spack area;
  first: see what would be installed:

```
spack spec --install-status py-black ^python@3.9.16
```

- Note that packages are labelled:

   - not installed

  [+] installed here

  [^] installed in upstream spack instance)

- Actually install it:

```
spack install py-black ^python@3.9.16
spack spec --install-status py-black ^python@3.9.16
```

Installing prebuilt packages from buildcache

In theory, plain spack install will get things from a buildcache, but it is difficult to give a command line spec that matches…

Recommend

- using spack buildcache install by hash:

```
spack buildcache list -al watch
spack buildcache install -oa /tyd3og5
```

- Installing an environment with a spack.lock file (in a later slide)

Developing sw in your environment

Setting up and use a build environment in our chained instance

```
spack env create myenv1 $HOME/saved_spack.lock
spack env activate myenv1
spack develop art@develop
spack develop art-root-io@develop
spack config edit
```

...change version of art and art-root-io to "develop"

```
spack cd --env
cd art
cd ../art-root-io
spack concretize --force
spack install
```

Adding packages to a buildcache

Add a gpg signing key if needed

```
spack gpg list
gpg-agent --homedir=/dir/from/above --daemon &
gpg --gen-key --homedir=/dir/from/above
cp /dir/from/above/secring.gpg /some/place/safe
```

Put signed packages in a local buildcache directory

```
spack buildcache create -k gpg-key ./bc spec
```

Copy them to a distribution area & reindex (needs permissions added)

```
scp -r ./bc/build_cache products@fifeutilgpvm01:/spack_cache/
ssh products@fifeutilgpvm01 sh /spack_cache/.mkindex.html
```

Installing in cvmfs

Installing into cvmfs: use (only) pre-built packages

- Login into cvmfs node,

```
ssh cvmfsmu2e@oasiscfs01
. /cvmfs/$EXP.opensciencegrid.org/packages/setup-env.sh
```

- Start a cvmfs transaction

```
cvmfs_server transaction $EXP.opensciencegrid.org
```

- Install with buildcache intall hash

```
spack buildcache install -oa /hash1
```

- …or install an environment from a lock file

```
spack env create newenv /path/to/spack.lock
spack -e newenv install
```

- End the cvmfs transaction

```
touch xyz/.cvmfscatalog # to partition cvmfs catalogs
cvmfs_server publish  $EXP.opensciencegrid.org
```

Building with upstream packages

Most experiments will be building their own packages against toolsets like the Art suite or LArSoft.

Recommendation:

- Install exact env: `spack env create name spack.lock`
- `spack add new_package` to environment
- `spack install`
- Use resulting spack.lock and buildcache to distribute

Working around the “concretizer”

Sometimes, Spack will just not concretize a new package depending on an existing one. You can “change its mind”:

- `spack spec -yaml new_package_spec > file1.yaml`
  `Spack spec -yaml existing/hash > file2.yaml`
  - Save output of concretizing and existing spec
  - Create combined file with upper package from file1 with hashes replaced and lower packages from file2
- `spack install -f combined_file.yaml`
  - Install already concretized package

After the class

To learn more

- Run commands with --help
- Read the documentation at spack.readthedocs.io