A noob test: Spack "vs" EasyBuild

TECHNISCHE UNIVERSITÄT DARMSTADT

7th EasyBuild User Meeting

26th January 2022, Virtual

NHR4CES

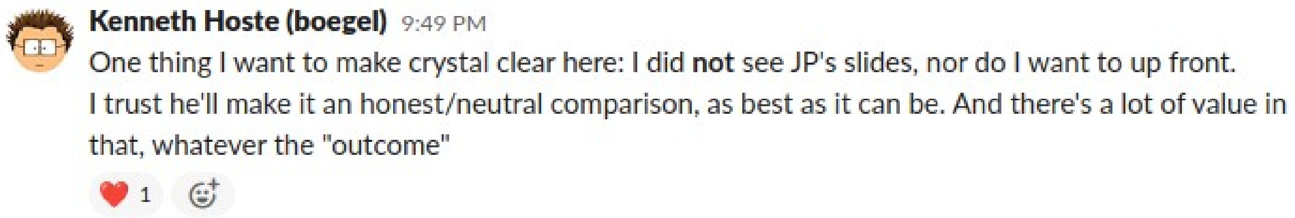
Jan-Patrick Lehr @jplehr

jan-patrick.lehr@tu-darmstadt.de

SF 4.0

Software-Factory 4.0

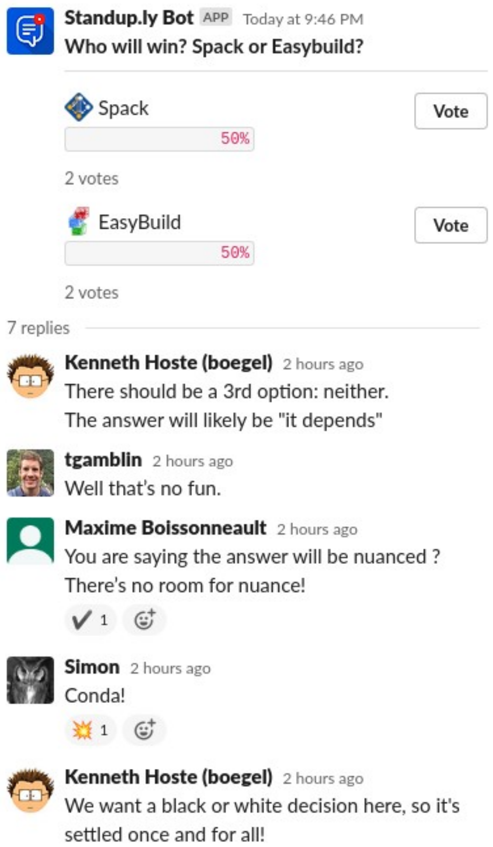
Spack “vs” EasyBuild



Kenneth Hoste (boegel) 9:49 PM

One thing I want to make crystal clear here: I did **not** see JP's slides, nor do I want to up front.
I trust he'll make it an honest/neutral comparison, as best as it can be. And there's a lot of value in that, whatever the "outcome"

❤️ 1



Standup.ly Bot APP Today at 9:46 PM

Who will win? Spack or Easybuild?

Spack 50% Vote

2 votes

EasyBuild 50% Vote

2 votes

7 replies

Kenneth Hoste (boegel) 2 hours ago

There should be a 3rd option: neither.
The answer will likely be "it depends"

tgamblin 2 hours ago

Well that’s no fun.

Maxime Boissonneault 2 hours ago

You are saying the answer will be nuanced ?
There’s no room for nuance!

✓ 1

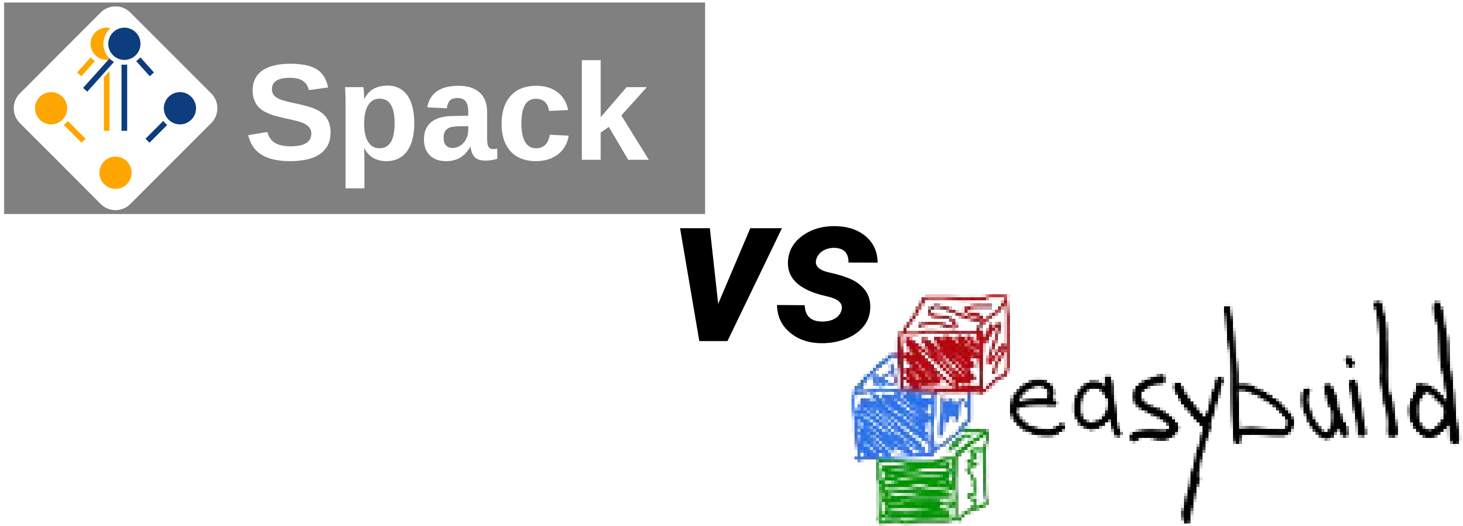
Simon 2 hours ago

Conda!

💥 1

Kenneth Hoste (boegel) 2 hours ago

We want a black or white decision here, so it's settled once and for all!



My Background

- **No** system administrator
- **No** Linux guru

- In the larger field of HPC since 2011 (student, PhD student, postdoc)
- Work on performance measurement and program analysis tools
- I attended a Spack tutorial at ISC several years ago (and never used it)

- Started streaming on twitch to motivate myself during final year of PhD
- Opportunity to reserve time for “trying out stuff of my want-to-try list”

ToolThursday

- Weekly “show” on twich to approach a tool / library / technology that I think
  - Helps me with a particular task
  - Is relevant to my field
  - May be fun to see
  - Was suggested as something cool to look at by someone else

- First two tools on the list: **Spack** and **EasyBuild**
  - **Spack**: 4 episodes so far, totaling ~7 hours
  - **EasyBuild**: 2 episodes so far, totaling ~4 hours

- Every Thursday from 9.30pm CET – 11.00pm CET (sometimes even midnight!)

Goal

- Familiarize myself a little bit with both tools for educational purposes
  - Our compute center decided to go with Spack

- Set up the development environment on my home machine
  - Running Manjaro Linux with Cinnamon DE
  - Currently using containers as development environment

- Create a package file / easyconfig for my tool MetaCG [1] as a starting point
  - Potentially creating these for all software we develop in our group

MetaCG

- Collection of tools and libraries for call-graph construction and analysis for heuristic profiling hook placement within PIRA [2].
  - More info: PhD Presentation [3], FOSDEM’21 talk, FOSDEM’22 talk on Feb. 6th

- Dependencies
  - GCC 9, Clang / LLVM 10
  - OpenMPI
  - nlohmann json
  - cxxopts
  - Score-P / Cube
  - Extra-P 3.0

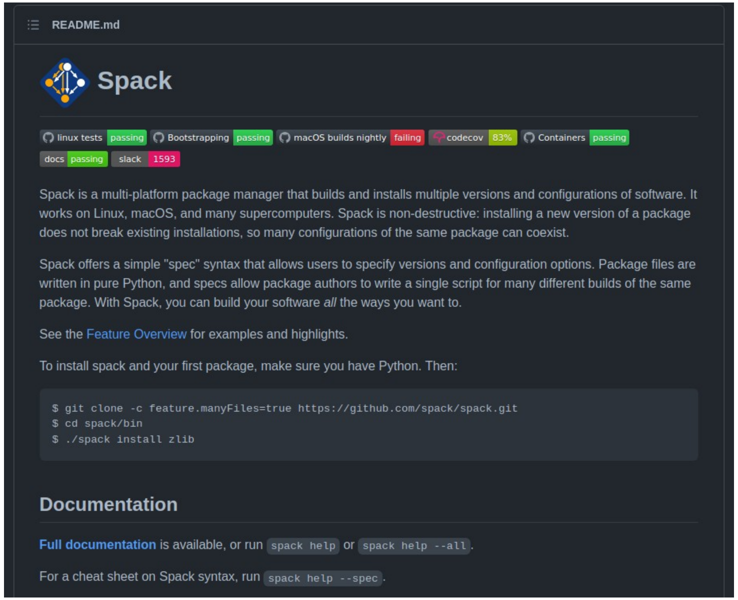
https://github.com/tudasc/metacg

Spack

- https://github.com/spack/spack

- “Spack is a multi platform package manager that builds and installs multiple versions and configurations of software.”

- **My understanding:** I type “install gcc” and Spack resolves dependencies and installs them if they are not already there. Like apt-get on Ubuntu.



Spack

Spack is a multi-platform package manager that builds and installs multiple versions and configurations of software. It works on Linux, macOS, and many supercomputers. Spack is non-destructive: installing a new version of a package does not break existing installations, so many configurations of the same package can coexist.

Spack offers a simple "spec" syntax that allows users to specify versions and configuration options. Package files are written in pure Python, and specs allow package authors to write a single script for many different builds of the same package. With Spack, you can build your software *all* the ways you want to.

See the Feature Overview for examples and highlights.

To install spack and your first package, make sure you have Python. Then:

```
$ git clone -c feature.manyFiles=true https://github.com/spack/spack.git
$ cd spack/bin
$ ./spack install zlib
```

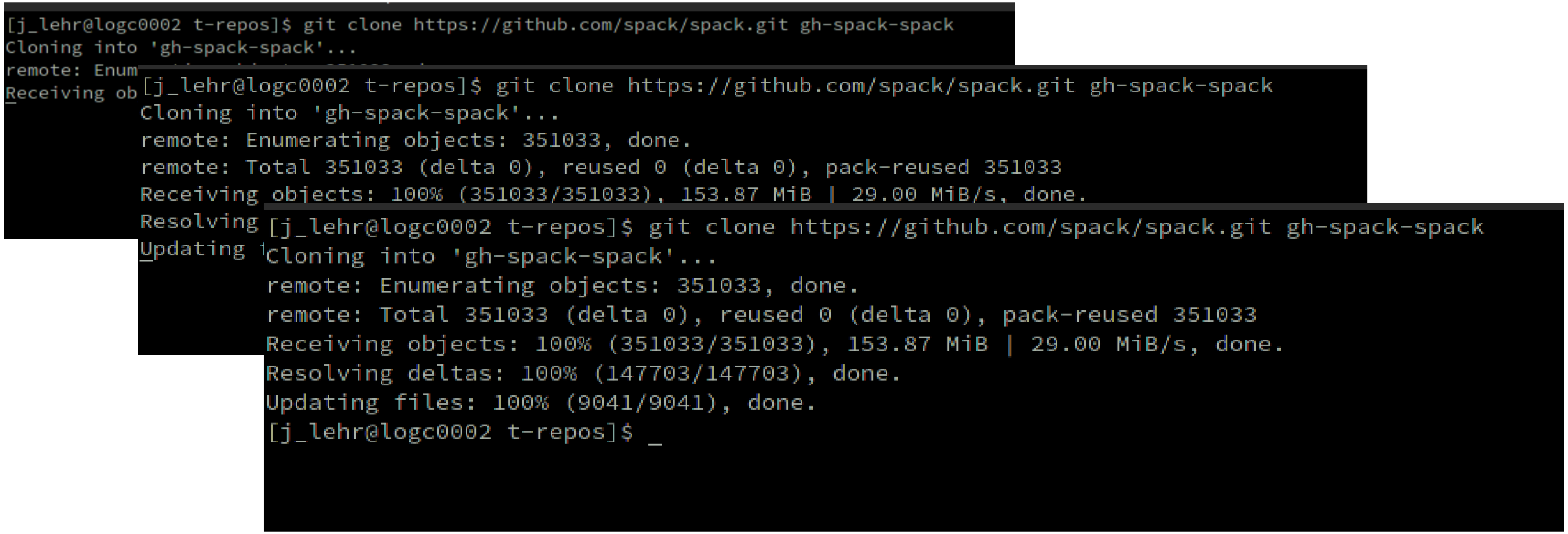
Documentation

Full documentation is available, or run `spack help` or `spack help --all`.

For a cheat sheet on Spack syntax, run `spack help --spec`.

Installation

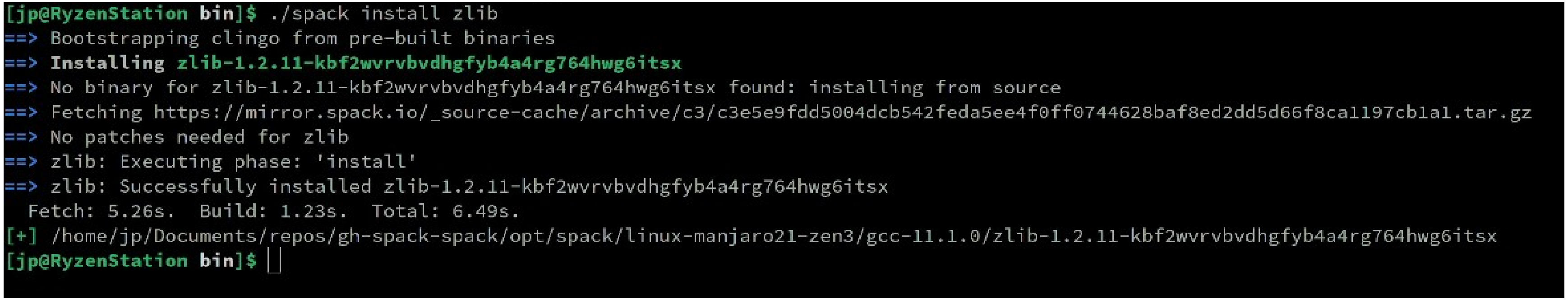
- Clone the git repository
- Source a script, if you want to use things like **`spack load cmake`**



```
[j_lehr@logc0002 t-repos]$ git clone https://github.com/spack/spack.git gh-spack-spack
Cloning into 'gh-spack-spack'...
remote: Enumerating objects: 351033, done.
remote: Total 351033 (delta 0), reused 0 (delta 0), pack-reused 351033
Receiving objects: 100% (351033/351033), 153.87 MiB | 29.00 MiB/s, done.
Resolving deltas: 100% (147703/147703), done.
Updating files: 100% (9041/9041), done.
[j_lehr@logc0002 t-repos]$ _
```

First Build

- **Installing `zlib`**

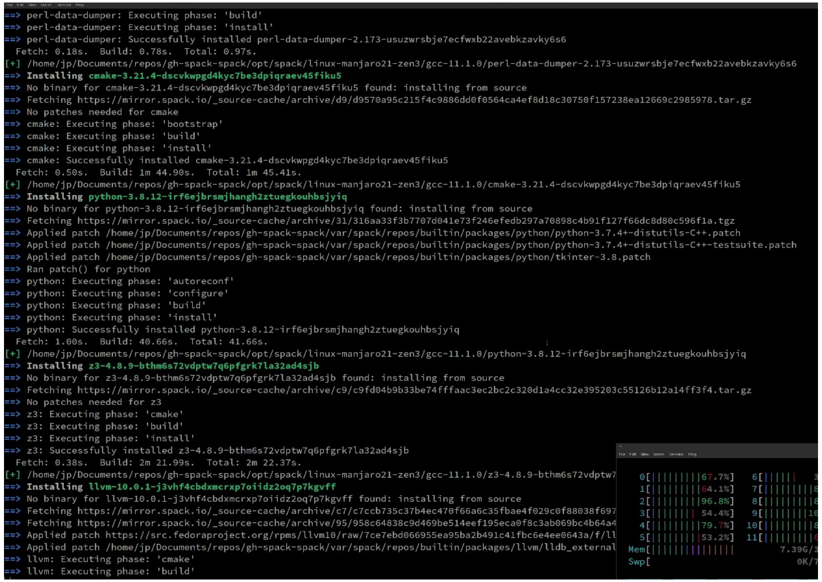


```
[jp@RyzenStation bin]$ ./spack install zlib
==> Bootstrapping clingo from pre-built binaries
==> Installing zlib-1.2.11-kbf2wvrvbvdhgfyb4a4rg764hwg6itsx
==> No binary for zlib-1.2.11-kbf2wvrvbvdhgfyb4a4rg764hwg6itsx found: installing from source
==> Fetching https://mirror.spack.io/_source-cache/archive/c3/c3e5e9fdd5004dcb542feda5ee4f0ff0744628baf8ed2dd5d66f8ca1197cb1a1.tar.gz
==> No patches needed for zlib
==> zlib: Executing phase: 'install'
==> zlib: Successfully installed zlib-1.2.11-kbf2wvrvbvdhgfyb4a4rg764hwg6itsx
  Fetch: 5.26s.  Build: 1.23s.  Total: 6.49s.
[+] /home/jp/Documents/repos/gh-spack-spack/opt/spack/linux-manjaro21-zen3/gcc-11.1.0/zlib-1.2.11-kbf2wvrvbvdhgfyb4a4rg764hwg6itsx
[jp@RyzenStation bin]$
```

Building the Toolchain

- Spack didn’t realize that the system GCC can compile Fortran (gfortran not being installed on the system)
  - The error message wasn’t particularly clear

- Installed GCC version 9.1 (w/ Fortran)
- Installed all other dependencies
  - Except Extra-P



```
==> perl-data-dumper: Executing phase: 'build'
==> perl-data-dumper: Executing phase: 'install'
==> perl-data-dumper: Successfully installed perl-data-dumper-2.173-usuzwrsbje7ecfwxb22avebkzavky6s6
  Fetch: 0.18s.  Build: 0.78s.  Total: 0.97s.
[+] /home/jp/Documents/repos/gh-spack-spack/opt/spack/linux-manjaro21-zen3/gcc-11.1.0/perl-data-dumper-2.173-usuzwrsbje7ecfwxb22avebkzavky6s6
==> Installing cmake-3.21.4-dscvkwpgd4kyc7be3dpiqraev45fiku5
==> No binary for cmake-3.21.4-dscvkwpgd4kyc7be3dpiqraev45fiku5 found: installing from source
==> Fetching https://mirror.spack.io/_source-cache/archive/d9/d9570a95c215f4c9886dd0f0564ca4ef8d18c30750f157238ea12669c2985978.tar.gz
==> No patches needed for cmake
==> cmake: Executing phase: 'bootstrap'
==> cmake: Executing phase: 'build'
==> cmake: Executing phase: 'install'
==> cmake: Successfully installed cmake-3.21.4-dscvkwpgd4kyc7be3dpiqraev45fiku5
  Fetch: 0.50s.  Build: 1m 44.90s.  Total: 1m 45.41s.
[+] /home/jp/Documents/repos/gh-spack-spack/opt/spack/linux-manjaro21-zen3/gcc-11.1.0/cmake-3.21.4-dscvkwpgd4kyc7be3dpiqraev45fiku5
==> Installing python-3.8.12-irf6ejbrsmjhangh2ztuegkouhbsjyiq
==> No binary for python-3.8.12-irf6ejbrsmjhangh2ztuegkouhbsjyiq found: installing from source
==> Fetching https://mirror.spack.io/_source-cache/archive/31/316aa33f3b7707d041e73f246efedb297a70898c4b91f127f66dc8d80c596f1a.tgz
==> Applied patch /home/jp/Documents/repos/gh-spack-spack/var/spack/repos/builtin/packages/python/python-3.7.4+-distutils-C++.patch
==> Applied patch /home/jp/Documents/repos/gh-spack-spack/var/spack/repos/builtin/packages/python/python-3.7.4+-distutils-C++-testsuite.patch
==> Applied patch /home/jp/Documents/repos/gh-spack-spack/var/spack/repos/builtin/packages/python/tkinter-3.8.patch
==> Ran patch() for python
==> python: Executing phase: 'autoreconf'
==> python: Executing phase: 'configure'
==> python: Executing phase: 'build'
==> python: Executing phase: 'install'
==> python: Successfully installed python-3.8.12-irf6ejbrsmjhangh2ztuegkouhbsjyiq
  Fetch: 1.00s.  Build: 40.66s.  Total: 41.66s.
[+] /home/jp/Documents/repos/gh-spack-spack/opt/spack/linux-manjaro21-zen3/gcc-11.1.0/python-3.8.12-irf6ejbrsmjhangh2ztuegkouhbsjyiq
==> Installing z3-4.8.9-bthm6s72vdptw7q6pfgrk7la32ad4sjb
==> No binary for z3-4.8.9-bthm6s72vdptw7q6pfgrk7la32ad4sjb found: installing from source
==> Fetching https://mirror.spack.io/_source-cache/archive/c9/c9fd04b9b33be74fffaac3ec2bc2c320d1a4cc32e395203c55126b12a14ff3f4.tar.gz
==> No patches needed for z3
==> z3: Executing phase: 'cmake'
==> z3: Executing phase: 'build'
==> z3: Executing phase: 'install'
==> z3: Successfully installed z3-4.8.9-bthm6s72vdptw7q6pfgrk7la32ad4sjb
  Fetch: 0.38s.  Build: 2m 21.99s.  Total: 2m 22.37s.
[+] /home/jp/Documents/repos/gh-spack-spack/opt/spack/linux-manjaro21-zen3/gcc-11.1.0/z3-4.8.9-bthm6s72vdptw7
==> Installing llvm-10.0.1-j3vhf4cbdxmcrxp7oiidz2oq7p7kgvff
==> No binary for llvm-10.0.1-j3vhf4cbdxmcrxp7oiidz2oq7p7kgvff found: installing from source
==> Fetching https://mirror.spack.io/_source-cache/archive/c7/c7ccb735c37b4ec470f66a6c35fbae4f029c0f88038f697
==> Fetching https://mirror.spack.io/_source-cache/archive/95/958c64838c9d469be514eef195eca0f8c3ab069bc4b64a4
==> Applied patch https://src.fedoraproject.org/rpms/llvm10/raw/7ce7ebd066955ea95ba2b491c41fbc6e4ee0643a/f/ll
==> Applied patch /home/jp/Documents/repos/gh-spack-spack/var/spack/repos/builtin/packages/llvm/lldb_external
==> llvm: Executing phase: 'cmake'
==> llvm: Executing phase: 'build'
```

```
  0[|||||||||67.7%]   6[|||||
  1[|||||||||64.1%]   7[||||||||
  2[|||||||||96.8%]   8[||||||||
  3[|||||||  54.4%]   9[||||||||
  4[|||||||||79.7%]  10[||||||||
  5[|||||||||53.2%]  11[||||||||
Mem[||||||||||||||            7.39G/
Swp[                             0K/
```

spack install llvm@10.0.1 %gcc@9.1.0

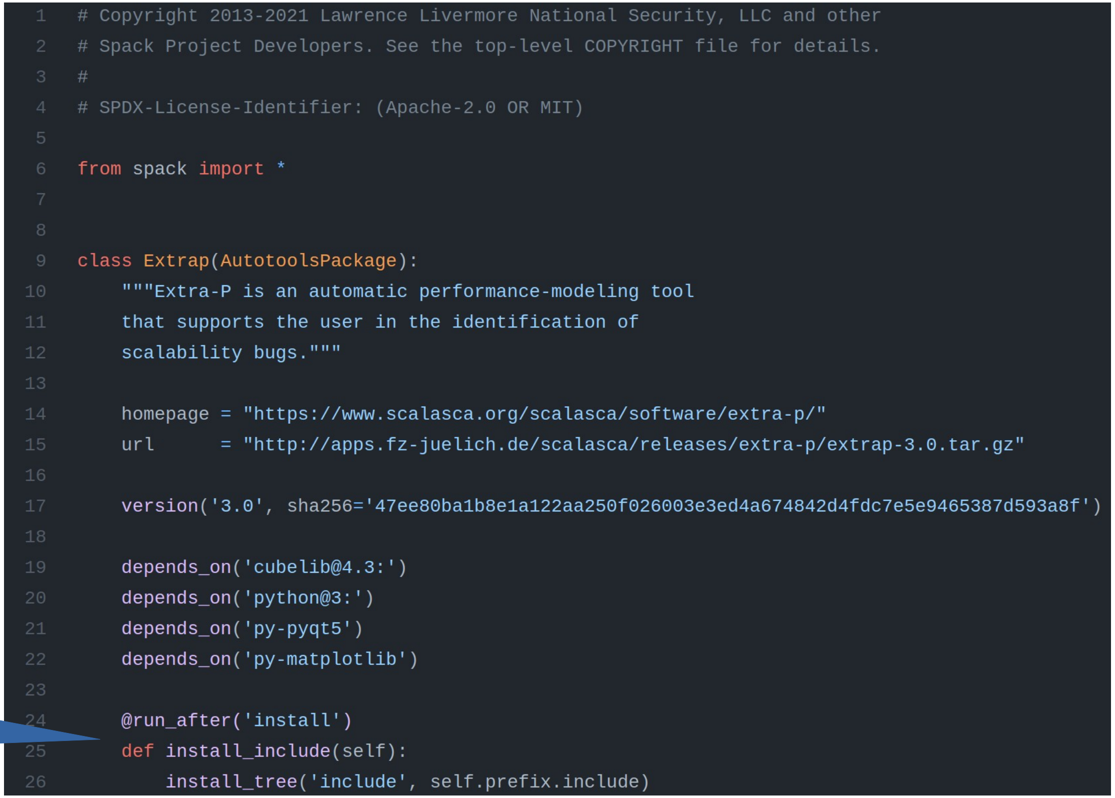
spack install cubelib@4.5 %gcc@9.1.0

Building the Toolchain

- Created the Extra-P package (shout out to twitch user Babumts)
  - Documentation would have gotten me there eventually, but Babumts was faster

- This was the biggest obstacle

Need to copy the ‘include’ from source tree to install



```python
# Copyright 2013-2021 Lawrence Livermore National Security, LLC and other
# Spack Project Developers. See the top-level COPYRIGHT file for details.
#
# SPDX-License-Identifier: (Apache-2.0 OR MIT)

from spack import *


class Extrap(AutotoolsPackage):
    """Extra-P is an automatic performance-modeling tool
    that supports the user in the identification of
    scalability bugs."""

    homepage = "https://www.scalasca.org/scalasca/software/extra-p/"
    url      = "http://apps.fz-juelich.de/scalasca/releases/extra-p/extrap-3.0.tar.gz"

    version('3.0', sha256='47ee80ba1b8e1a122aa250f026003e3ed4a674842d4fdc7e5e9465387d593a8f')

    depends_on('cubelib@4.3:')
    depends_on('python@3:')
    depends_on('py-pyqt5')
    depends_on('py-matplotlib')

    @run_after('install')
    def install_include(self):
        install_tree('include', self.prefix.include)
```

Building MetaCG

Fitting for this meeting

- Did not succeed due to defficiencies in MetaCG’s CMake. :)

- It currently assumes (or requires) to have the source-tree for cxxopts
  - This is due to an older version of cxxopts being used
  - Reason has been fixed in current cxxopts

What I enjoyed w/ Spack

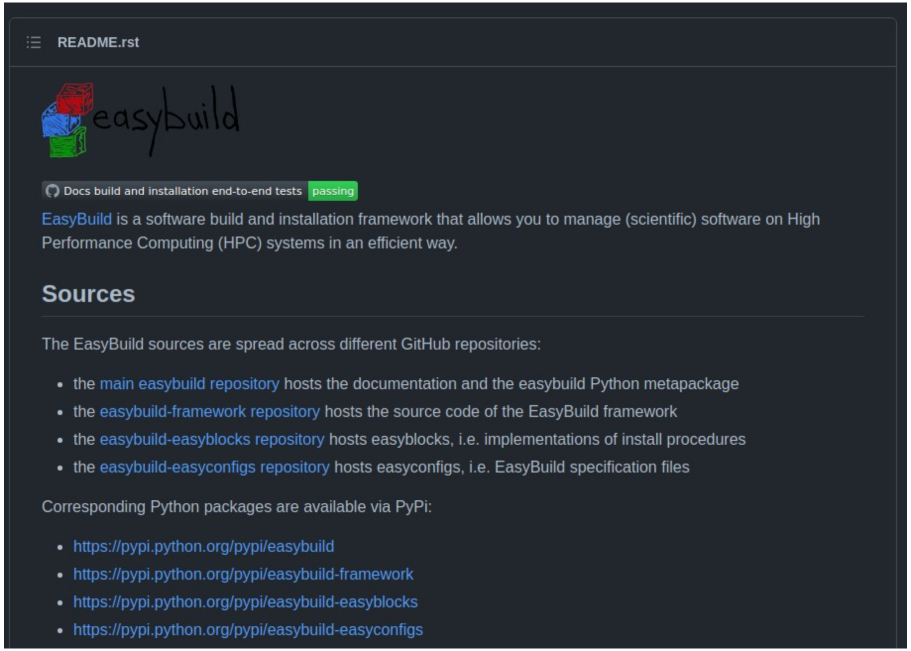
- I weirdly “enjoy” installing software by checking out a git repository
- The documentation told me what I needed to start
  - No *additional information* that I may want to know once I’m a Spack guru
- Once I got over the syntax initially, things worked out nicely and I was able to recall it faster than I expected
- The packages I needed built with the versions I set to be used

My Take Away

- For my limited use case, I did not encounter any **real** obstacle for using Spack
- **`spack find`** and **`list`** behave differently when giving partial names
- Spack does not display what it will build (at least per default)
- I don’t know how many versions of Perl I have by now installed on my system
- I felt there was too much documentation once the initial steps are done
  - I usually wasn’t sure where to look for what I searched for

EasyBuild

- https://github.com/easybuilders/easybuild
- “EasyBuild is a software build and installation framework that allows you to manage (scientific) software on High Performance Computing (HPC) systems in an efficient way.”
- **My understanding:** I have a list of packages that I can install with ease and configure for other versions when needed.



README.rst

EasyBuild is a software build and installation framework that allows you to manage (scientific) software on High Performance Computing (HPC) systems in an efficient way.

Sources

The EasyBuild sources are spread across different GitHub repositories:

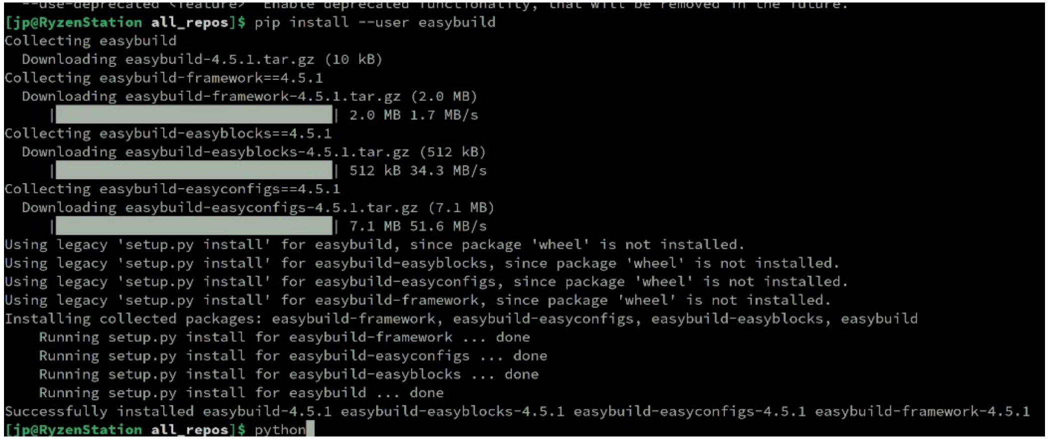
- the main easybuild repository hosts the documentation and the easybuild Python metapackage
- the easybuild-framework repository hosts the source code of the EasyBuild framework
- the easybuild-easyblocks repository hosts easyblocks, i.e. implementations of install procedures
- the easybuild-easyconfigs repository hosts easyconfigs, i.e. EasyBuild specification files

Corresponding Python packages are available via PyPi:

- https://pypi.python.org/pypi/easybuild
- https://pypi.python.org/pypi/easybuild-framework
- https://pypi.python.org/pypi/easybuild-easyblocks
- https://pypi.python.org/pypi/easybuild-easyconfigs

Installation

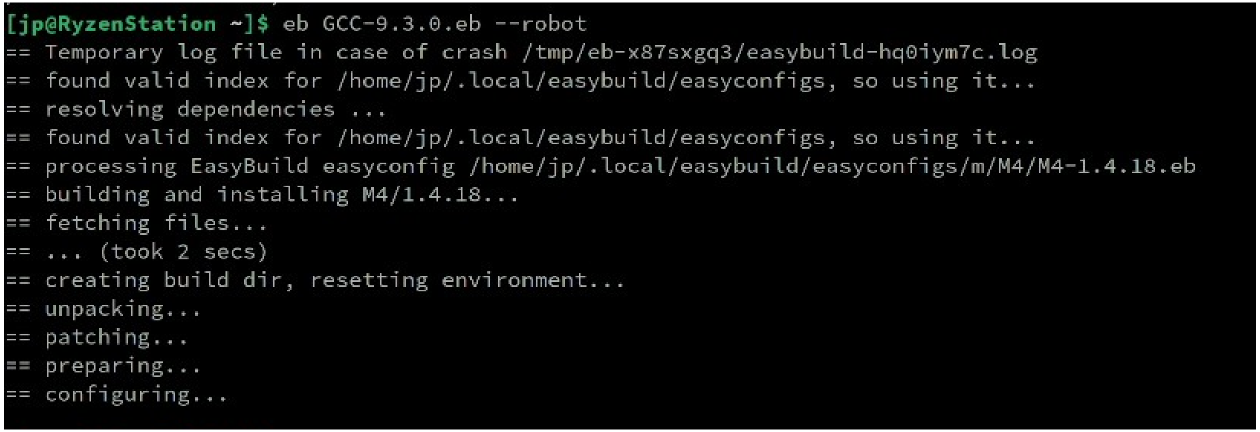
- Installed using pip
  - At the time of installation (remember I’m on Manjaro) was Python v3.9
  - At the time of use Manjaro updated the Python version to 3.10
  - Re-installed for Python v2.7 (thanks for the help Kenneth)
- I confused my machine with the cluster and that I did not have LMod installed
  - Sanity checks worked, but **`eb --search matplotlib`** failed



```
  --use-deprecated <feature>  Enable deprecated functionality, that will be removed in the future.
[jp@RyzenStation all_repos]$ pip install --user easybuild
Collecting easybuild
  Downloading easybuild-4.5.1.tar.gz (10 kB)
Collecting easybuild-framework==4.5.1
  Downloading easybuild-framework-4.5.1.tar.gz (2.0 MB)
     |████████████████████████████████| 2.0 MB 1.7 MB/s
Collecting easybuild-easyblocks==4.5.1
  Downloading easybuild-easyblocks-4.5.1.tar.gz (512 kB)
     |████████████████████████████████| 512 kB 34.3 MB/s
Collecting easybuild-easyconfigs==4.5.1
  Downloading easybuild-easyconfigs-4.5.1.tar.gz (7.1 MB)
     |████████████████████████████████| 7.1 MB 51.6 MB/s
Using legacy 'setup.py install' for easybuild, since package 'wheel' is not installed.
Using legacy 'setup.py install' for easybuild-easyblocks, since package 'wheel' is not installed.
Using legacy 'setup.py install' for easybuild-easyconfigs, since package 'wheel' is not installed.
Using legacy 'setup.py install' for easybuild-framework, since package 'wheel' is not installed.
Installing collected packages: easybuild-framework, easybuild-easyconfigs, easybuild-easyblocks, easybuild
    Running setup.py install for easybuild-framework ... done
    Running setup.py install for easybuild-easyconfigs ... done
    Running setup.py install for easybuild-easyblocks ... done
    Running setup.py install for easybuild ... done
Successfully installed easybuild-4.5.1 easybuild-easyblocks-4.5.1 easybuild-easyconfigs-4.5.1 easybuild-framework-4.5.1
[jp@RyzenStation all_repos]$ python
```

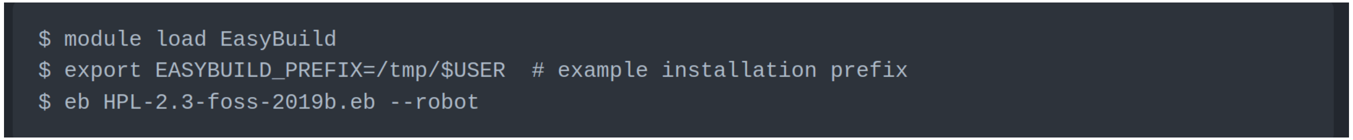
First Build

- Built GCC v9.3
  (initially forgot --robot)
- Went smoothly



```
[jp@RyzenStation ~]$ eb GCC-9.3.0.eb --robot
== Temporary log file in case of crash /tmp/eb-x87sxgq3/easybuild-hq0iym7c.log
== found valid index for /home/jp/.local/easybuild/easyconfigs, so using it...
== resolving dependencies ...
== found valid index for /home/jp/.local/easybuild/easyconfigs, so using it...
== processing EasyBuild easyconfig /home/jp/.local/easybuild/easyconfigs/m/M4/M4-1.4.18.eb
== building and installing M4/1.4.18...
== fetching files...
== ... (took 2 secs)
== creating build dir, resetting environment...
== unpacking...
== patching...
== preparing...
== configuring...
```

- The EB documentation didn’t help me from the get-go how to install something
  - Examples has **some** .eb file w/ versions in it
  - “I don’t know where that comes from.”

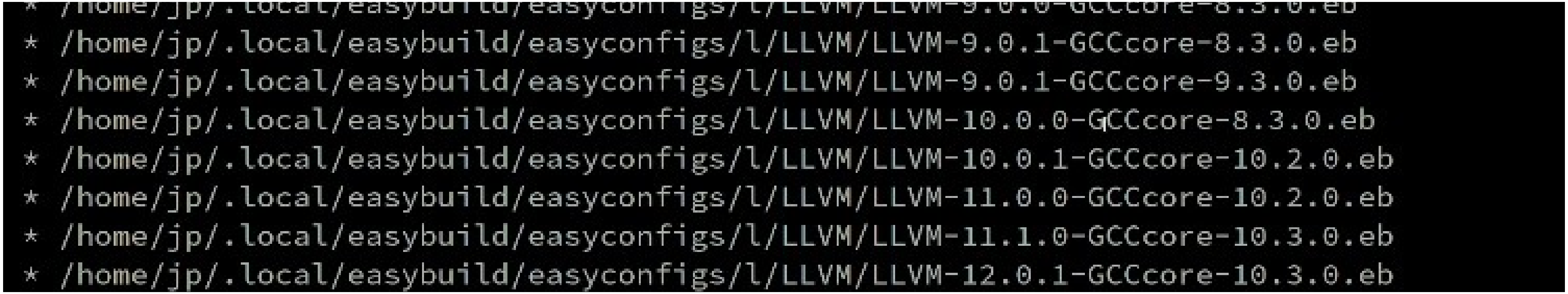


```
$ module load EasyBuild
$ export EASYBUILD_PREFIX=/tmp/$USER  # example installation prefix
$ eb HPL-2.3-foss-2019b.eb --robot
```

Building the Toolchain

- So then, just compile LLVM 10 with GCC 9.3

- Well, it’s missing a config with the right GCC version



```
 * /home/jp/.local/easybuild/easyconfigs/l/LLVM/LLVM-9.0.0-GCCcore-8.3.0.eb
 * /home/jp/.local/easybuild/easyconfigs/l/LLVM/LLVM-9.0.1-GCCcore-8.3.0.eb
 * /home/jp/.local/easybuild/easyconfigs/l/LLVM/LLVM-9.0.1-GCCcore-9.3.0.eb
 * /home/jp/.local/easybuild/easyconfigs/l/LLVM/LLVM-10.0.0-GCCcore-8.3.0.eb
 * /home/jp/.local/easybuild/easyconfigs/l/LLVM/LLVM-10.0.1-GCCcore-10.2.0.eb
 * /home/jp/.local/easybuild/easyconfigs/l/LLVM/LLVM-11.0.0-GCCcore-10.2.0.eb
 * /home/jp/.local/easybuild/easyconfigs/l/LLVM/LLVM-11.1.0-GCCcore-10.3.0.eb
 * /home/jp/.local/easybuild/easyconfigs/l/LLVM/LLVM-12.0.1-GCCcore-10.3.0.eb
```

Building the Toolchain

- Initially tried one of the LLVM 10 packages using
  - **`eb --software-name=LLVM-10.0.1 --toolchain=GCC,9.3.0`**
  - No success
  - “ERROR: Toolchain template not found, available toolchains: …, GCC, …”

- Then copied **`LLVM-10.0.1-GCC-10.2.0.eb`** to **`LLVM-10.0.1-GCC-9.3.0.eb`**
  - Changed the **`‘version’`** of GCC in **`‘toolchain’`** to 9.3.0
  - No success

- Ended with missing OS dependencies (which I now know, I should’ve ignored with **`--ignore-osdeps`**)

Building the Toolchain

- Next: using the **`--try-toolchain=GCCcore,9.3.0`**
  - Some more progress
  - LDAP version on Manjaro too new for cURL → fixing it with **`--without-ldap`**
  - Suggested by Kenneth

Building the Toolchain

- Next: using the `--try-software-version=10.0.1`
  - LDAP version on Manjaro too new for cURL → fixing it with `--without-ldap`

- Errors out on obscure CMake error
  - Support for C++17
  - Cannot make `std::unique_ptr<T>`

- Maybe (Actually?) a problem with the binutils version (and the Manjaro host?)
  - Change binutils version dependency in Cmake easy config

Building the Toolchain

- At this point I stopped (for now)
  - Had to prepare slides for this talk
  - Continue on Thursday on twitch

Obstacles

- **I want LLVM 10 built with GCC 9.3**
  - It felt as if I was working against the system (and my host system?)

- Documentation
  - The docs feel written for a different audience than me

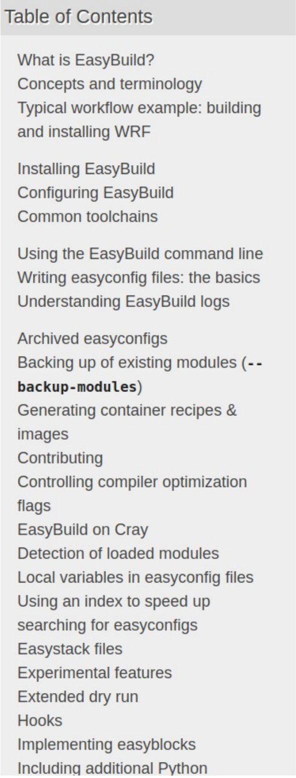


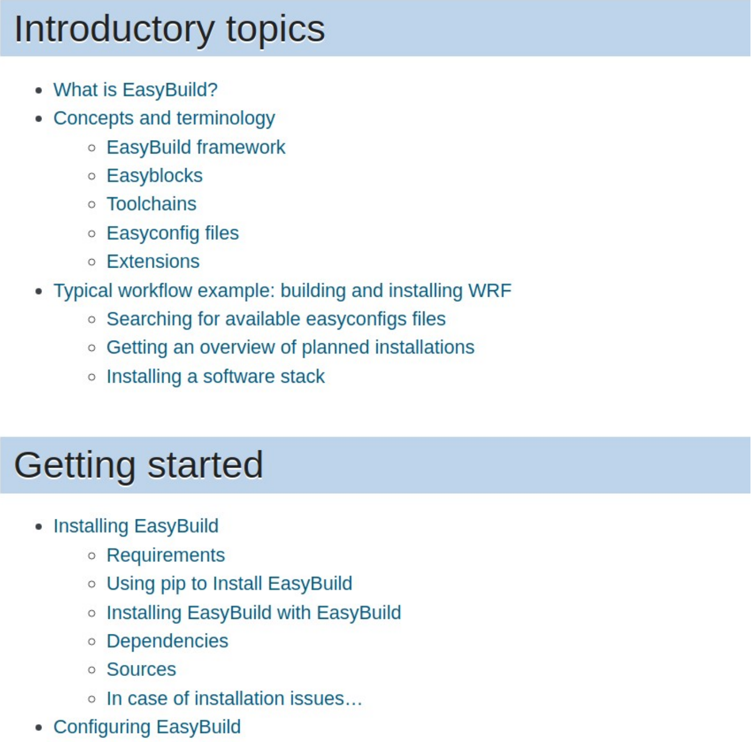
Table of Contents

What is EasyBuild?
Concepts and terminology
Typical workflow example: building and installing WRF

Installing EasyBuild
Configuring EasyBuild
Common toolchains

Using the EasyBuild command line
Writing easyconfig files: the basics
Understanding EasyBuild logs

Archived easyconfigs
Backing up of existing modules (`--backup-modules`)
Generating container recipes & images
Contributing
Controlling compiler optimization flags
EasyBuild on Cray
Detection of loaded modules
Local variables in easyconfig files
Using an index to speed up searching for easyconfigs
Easystack files
Experimental features
Extended dry run
Hooks
Implementing easyblocks
Including additional Python



Introductory topics

- What is EasyBuild?
- Concepts and terminology
  - EasyBuild framework
  - Easyblocks
  - Toolchains
  - Easyconfig files
  - Extensions
- Typical workflow example: building and installing WRF
  - Searching for available easyconfigs files
  - Getting an overview of planned installations
  - Installing a software stack

Getting started

- Installing EasyBuild
  - Requirements
  - Using pip to Install EasyBuild
  - Installing EasyBuild with EasyBuild
  - Dependencies
  - Sources
  - In case of installation issues…
- Configuring EasyBuild

My Take Away

- Available configs built smoothly and without issues
- It doesn’t go ahead and download / compile truck loads of stuff (unless **`--robot`**)
  - Although I’m also annoyed by me, forgetting giving it that flag
- I can see a lot of value in common toolchains
  - May be inspired from that for my own stuff / CI support
- The documentation didn’t help me a whole lot, but the twitch chat did

Comparison

Spack

- Clone git repository
- **`spack install gcc@9`**
- Compile LLVM 10 with GCC 9

- Docs tell me how to immediately use it
  - May not be the most efficient usage

- I got lost in the large documentation when I wanted to start with my own package file

EasyBuild

- Installation via pip
- **`eb GCC-9.3.0.eb --robot`**
- LLVM 10 with GCC 9 didn’t work despite changes

- Docs tell me a lot of details
  - I’m impatient → want to install stuff

- I did not yet get to the point where I have to implement an easy config

Summary / Conclusion

- For my daily work, i.e., I need some library for my development, I chose Spack
  - Recently needed CMake v3.22.x on our cluster and wanted to straightforwardly use module system compiler → **`spack install cmake@3.22`**

- For creating a user-facing software stack that can have a little bit of slack w.r.t. versions I may choose EasyBuild
  - Common toolchains for me most interesting thing in this regard

- Ad: Come and join me Thursday's for #ToolThursday on twitch.tv/jplehr ;) [4]

References / Links

- [1] **MetaCG**
  Repository: https://github.com/tudasc/metacg
  Paper: https://doi.org/10.1145/3427764.3428320
- [2] **PIRA**
  Repository: https://github.com/tudasc/pira
  Paper: https://doi.org/10.1145/3281070.3281071
- [3] **PhD Defence**
  YouTube: https://youtu.be/m_SD-BhJ924
- [4] **ToolThursday**
  Thursdays from 9.30 pm CET to ~11 pm CET on Twitch: https://twitch.tv/jplehr

- Paper preprints are on researchgate