SITE PRESENTATION

VUB (VRIJE UNIVERSITEIT BRUSSEL) - BELGIUM

CINTIA WILLEMYNS

OUTLINE

- ▶ Personal background
- ▶ Discovery of EasyBuild
- ▶ What Makes EasyBuild Easy to learn (and what is challenging)
- ▶ Description of VUB Hydra Infrastructure
- ▶ VUB Workflow for Software Installation @VUB
- ▶ Conclusion

INTRODUCTION

PERSONAL BACKGROUND

- Background
  - PhD physics (the pen and paper kind)
- Present
  - HPC team @VUB (since 2023) → Tier2 (academic) + Tier1 (industry)
- EasyBuild, HPC and even IT was very new to me.
  - Linux, Python, Fortran
  - No formal programming education
  - Little knowledge about hardware (still lots to learn here)

@WilleBell

DISCOVERY OF EASYBUILD

WHAT MAKES EASYBUILDE IT EASY FOR BEGINNERS

- Having some knowledge of Python -> gives a starting point (or any other language)

- Getting to know the way into EasyBuild can be made very gradually:
  - do (copy) very easy easyconfigs (where the “magic happens” in the background)
  - Getting to know some simple options
  - Going to installations in other languages (Python->R -> Ocaml)
  - Using different easyblocks
  - Getting to understand the background (looking at easyblocks)
  - Tweaks to work with particular installations, patching
  - …

Steep learning curve

DISCOVERY OF EASYBUILD

WHAT MAKES EASYBUILDE IT EASY FOR BEGINNERS

- Having some knowledge of Python -> gives a starting point (or any other language)
- Getting to know the way into EasyBuild can be made very gradually
- Little need for hardware knowledge
- EB Community

EasyBuild
easybuild.slack.com

EASYBUILD.io
building software with ease

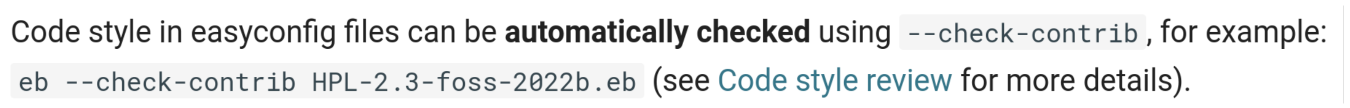
Documentation
Tutorial

HPC.social
hpcsocial.slack.com

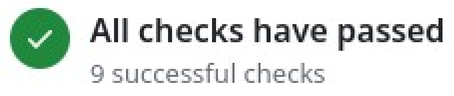
DISCOVERY OF EASYBUILD

INITIAL CHALLENGES

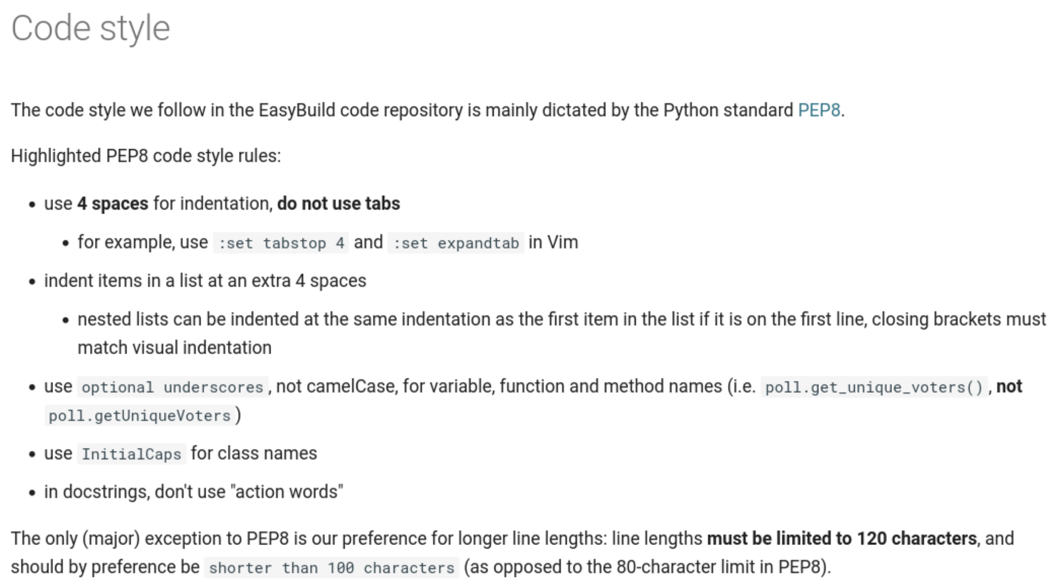
- Understanding all the different way to install a software (different languages, practices change overtime)
- Hard to understand what is standard/good practices



Code style in easyconfig files can be **automatically checked** using `--check-contrib`, for example: `eb --check-contrib HPL-2.3-foss-2022b.eb` (see Code style review for more details).



All checks have passed
9 successful checks



Code style

The code style we follow in the EasyBuild code repository is mainly dictated by the Python standard PEP8.

Highlighted PEP8 code style rules:

- use **4 spaces** for indentation, **do not use tabs**
    - for example, use `:set tabstop 4` and `:set expandtab` in Vim
- indent items in a list at an extra 4 spaces
    - nested lists can be indented at the same indentation as the first item in the list if it is on the first line, closing brackets must match visual indentation
- use `optional underscores`, not camelCase, for variable, function and method names (i.e. `poll.get_unique_voters()`, **not** `poll.getUniqueVoters`)
- use `InitialCaps` for class names
- in docstrings, don't use "action words"

The only (major) exception to PEP8 is our preference for longer line lengths: line lengths **must be limited to 120 characters**, and should by preference be `shorter than 100 characters` (as opposed to the 80-character limit in PEP8).

- e.g. Suffix or no suffix? extensions depending on extensions?

INFRASTRUCTURE AND SOME STATS

HYDRA (VUB TIER-2 CLUSTER WITHIN THE VSC)

Heterogeneous cluster → different CPU micro architectures, different interconnects

CPU-only nodes

- Partition: broadwell
  - 26 nodes → each node: 2x 14-core INTEL E5-2680v4 (Broadwell) and 256 GB per node
- Partition: broadwell_himem
  - 1 node: 4x 10-core INTEL E7-8891v4 and 1.5 TB
- Partition: skylake
  - 22 nodes → each node: 2x 20-core INTEL Xeon Gold 6148 (Skylake) and 192 GB per node
- Partition: skylake_mpi
  - 49 nodes → each node: 2x 20-core INTEL Xeon Gold 6148 (Skylake) and 192 GB per node and IB

GPU nodes

- Partition: pascal
  - 4 nodes → each node:
    - GPUs:2x Nvidia Tesla P100 (Pascal)
    - Processors: 2x 12-core INTEL E5-2650v4 (Broadwell)
- Partition: ampere
  - 10 nodes → each node:
    - 2x Nvidia A100 (Ampere)
    - 2x 16-core AMD EPYC 7282 (Zen2)

Soon + 20 zen4 nodes

INSTALLATION WORKFLOW

OVERVIEW OF SOFTWARE INSTALLATION

1 Write easyconfig (and/or easyblock)

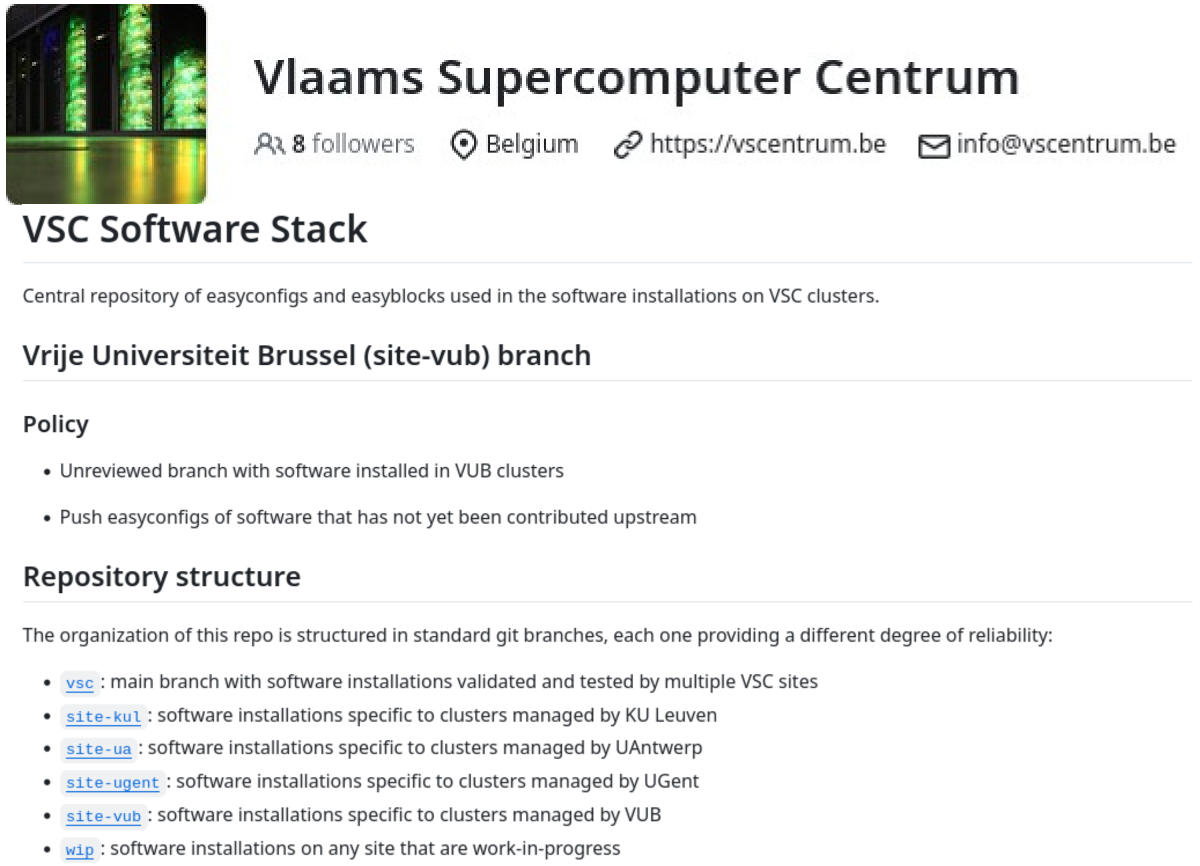
2 Test it locally

3 Create a PR in Easybuilders repo

4 Create a commit in our `site-vub` branch

5 Install in Hydra → installation script

6 Git revert on our `site-vub` repo when PR is merged


Vlaams Supercomputer Centrum

8 followers · Belgium · https://vscentrum.be · info@vscentrum.be

VSC Software Stack

Central repository of easyconfigs and easyblocks used in the software installations on VSC clusters.

Vrije Universiteit Brussel (site-vub) branch

Policy

- Unreviewed branch with software installed in VUB clusters
- Push easyconfigs of software that has not yet been contributed upstream

Repository structure

The organization of this repo is structured in standard git branches, each one providing a different degree of reliability:

- `vsc` : main branch with software installations validated and tested by multiple VSC sites
- `site-kul` : software installations specific to clusters managed by KU Leuven
- `site-ua` : software installations specific to clusters managed by UAntwerp
- `site-ugent` : software installations specific to clusters managed by UGent
- `site-vub` : software installations specific to clusters managed by VUB
- `wip` : software installations on any site that are work-in-progress

INSTALLATION WORKFLOW

OVERVIEW OF SOFTWARE INSTALLATION

1 Write easyconfig (and/or easyblock)

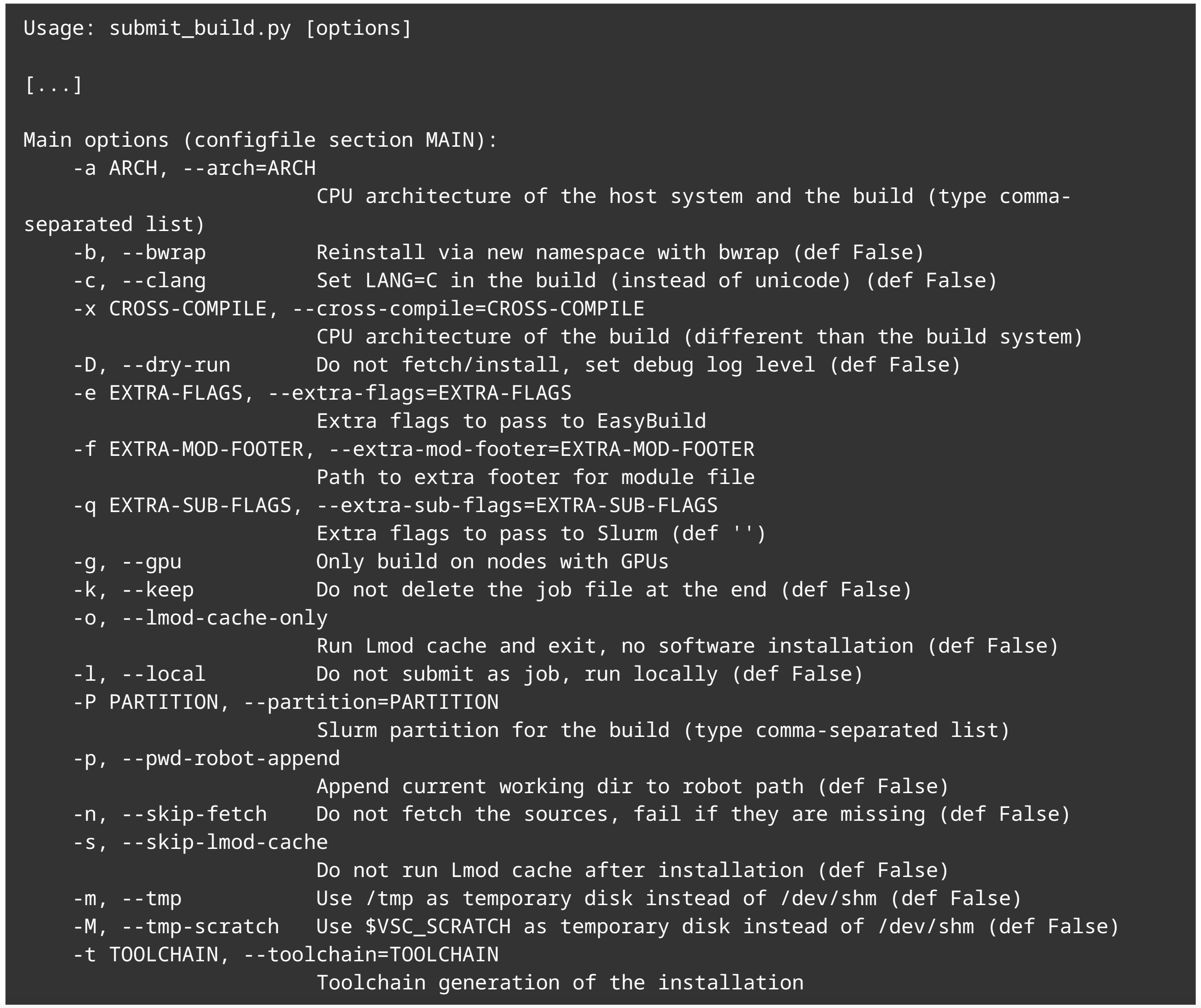
2 Test it locally

3 Create a PR in Easybuilders repo

4 Create a commit in our `site-vub` branch

5 Install in Hydra → installation script

6 Git revert on our `site-vub` repo when PR is merged



```
Usage: submit_build.py [options]

[...]

Main options (configfile section MAIN):
    -a ARCH, --arch=ARCH
                        CPU architecture of the host system and the build (type comma-
separated list)
    -b, --bwrap         Reinstall via new namespace with bwrap (def False)
    -c, --clang         Set LANG=C in the build (instead of unicode) (def False)
    -x CROSS-COMPILE, --cross-compile=CROSS-COMPILE
                        CPU architecture of the build (different than the build system)
    -D, --dry-run       Do not fetch/install, set debug log level (def False)
    -e EXTRA-FLAGS, --extra-flags=EXTRA-FLAGS
                        Extra flags to pass to EasyBuild
    -f EXTRA-MOD-FOOTER, --extra-mod-footer=EXTRA-MOD-FOOTER
                        Path to extra footer for module file
    -q EXTRA-SUB-FLAGS, --extra-sub-flags=EXTRA-SUB-FLAGS
                        Extra flags to pass to Slurm (def '')
    -g, --gpu           Only build on nodes with GPUs
    -k, --keep          Do not delete the job file at the end (def False)
    -o, --lmod-cache-only
                        Run Lmod cache and exit, no software installation (def False)
    -l, --local         Do not submit as job, run locally (def False)
    -P PARTITION, --partition=PARTITION
                        Slurm partition for the build (type comma-separated list)
    -p, --pwd-robot-append
                        Append current working dir to robot path (def False)
    -n, --skip-fetch    Do not fetch the sources, fail if they are missing (def False)
    -s, --skip-lmod-cache
                        Do not run Lmod cache after installation (def False)
    -m, --tmp           Use /tmp as temporary disk instead of /dev/shm (def False)
    -M, --tmp-scratch   Use $VSC_SCRATCH as temporary disk instead of /dev/shm (def False)
    -t TOOLCHAIN, --toolchain=TOOLCHAIN
                        Toolchain generation of the installation
```

CONCLUSIONS

EASYBUILD.io

building software with ease

- EasyBuild is easy
- Very steep learning curve (at the beginning)
- Helpful community (nice Slack channels, patient colleagues and maintainers)
- Allows for efficient workflow for installations in our cluster
- I hopefully to contribute further to the EasyBuild community

QUESTIONS?
SUGGESTIONS?