FRALIN LIFE SCIENCES INSTITUTE
VIRGINIA TECH

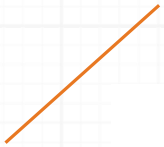
Introduction to EasyBuild

DOMINIK BORKOWSKI

May 15, 2019 – Maryland HPC User Group

About us

- Virginia Bioinformatics Institute at Virginia Tech, opened doors in 2000
- Changed name to Biocomplexity Institute in 2015
- In 2019 started process of growing to become Fralin Life Sciences Institute
- IT portfolio features HPC resources and services



Why EasyBuild?

Problems with traditional methods

High cost

Installing and maintaining software from source requires a lot of time and effort

No code reuse

Installing new versions of software benefits very little from previous efforts

Lack of flexibility

Binary package systems focus on a single version compiled against a single set of compilers and libraries

User adverse

Installing or upgrading software can be performed only system administrators

Brief history

- Started in 2009, EasyBuild reached version 1.0, stable API and GPL license in 2012 (announced at SC'12)
- Developed originally by IT department of Ghent University in Belgium, member of Flemish Supercomputer Center

Most recent version 3.8.3 provides:

- 1,711 different software recipes (easyconfigs) with 30 compiler toolchains
- 200 software specific and 34 generic methods (easyblocks)

What is EasyBuild?

- Software build and installation framework, written in Python
- Supports different compilers and MPI libraries
- Very active and helpful community
- Features necessary to make it work for our environment at FLSI/VT were implemented by the core development team
- Software recipes are written by many people around the world

Terminology

EasyBuild framework

- core: Python modules and packages
- provides supporting functionality for building and installing software

easyblock

- Python module which serves as a build script, ‘plugin’ for the EasyBuild framework
- implements a (generic) software build/install procedure

Terminology (continued)

easyconfig file

- (*.eb): build specification, software name/version, compiler toolchain, source URL, dependency list

(compiler) toolchain

- set of compilers with accompanying libraries

extensions

- additional libraries/packages/modules for a particular application (e.g., Python, R)

EasyBuild install stages

Check easyconfig

Check validity of easyconfig, including presence of dependencies

Download source

Download, unpack, and prepare source or installation files

Build

Run configure, make, and test, or relevant recipe for given software

Install

Verify installation and generate modules

Toolchains

Most easyconfigs target either FOSS or INTEL toolchains, we picked FOSS:

- foss/2018a, foss/2018b, foss/2019a, etc

foss toolchain comprises of:

- compiler - GCC
- MPI implementation - OpenMPI
- libraries - OpenBLAS/LAPACK, ScaLAPACK(/BLACS), FFTW

EasyBuild at FLSI

What's different at FLSI?

Environment modules on each cluster have pre-set variables for:

- HPC administrators for managing globally installed software
- Users and their local installations

Users' environment allows access to ALL global software, and build locally any additional/missing items

- Users can create their own easyconfigs, or override global ones

Directory structure takes cluster name and architecture into account

- a single home directory or /apps dir can be shared between many clusters

Our workflow

- Environment module setup, documentation, and easyconfigs used by us are checked into a central git repository
- After deploying new software new easyconfigs and modules are checked into the repo
- To set up a new cluster we simply check out repository and build all configs

Repository

- https://github.com/dominikborkowski/biocomplexity_easybuild

Additional resources

Documentation

EasyBuild has excellent documentation

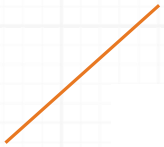
- https://easybuild.readthedocs.io/

Presentations

- https://users.ugent.be/~kehoste/ceci_20190425.pdf
- https://users.ugent.be/~kehoste/EasyBuild_20190510_CSCS_software_management.pdf

Friendly support is available via multiple venues

- Slack, GitHub, IRC, mailing list



Demo time!