Software installation on Compute Canada clusters using EasyBuild

Introduction to EasyBuild for users.

Ali Kerrache,

High Performance Computing Analyst,
University of Manitoba, Winnipeg, MB
WestGrid, Compute Canada.

Outline

- ❑ Introduction to Compute Canada software stack
- ❑ Local installation (user’s directory):
  - ➢ R, Python and Perl packages
  - ➢ Open source programs, … etc.
- ❑ Introduction to EasyBuild
  - ➢ Concept of EasyBuild
  - ➢ Basics of EasyBuild
  - ➢ Examples
  - ➢ Demonstration

Software installation and distribution

Operating system package managers / repos:

- ❑ **Ubuntu:** ~$ ***sudo apt-get install <package>***
- ❑ **CentOS:** ~$ ***sudo yum install <package>***
- ❑ **On HPC:** users do not have **sudo**! (**DO NOT ASK FOR IT**)

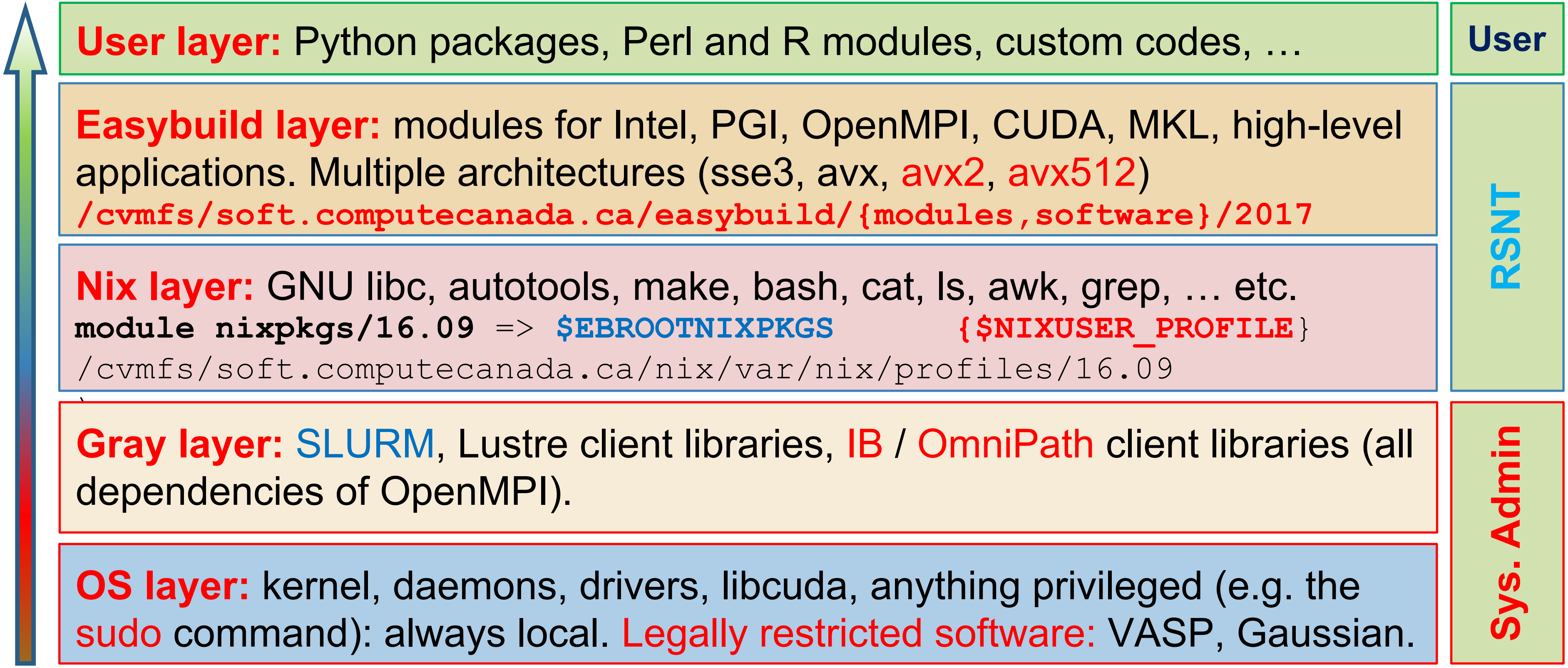
Local installation: usually to $HOME or $PROJECT

- ➢ **Get the code:** download the sources/binaries: wget, git clone, … etc.
- ➢ **Settings:** load dependencies, set environment variables, … etc.
- ➢ **Build:** ./configure {cmake ..} +opts; make; make test {check}; make install

Using a centralized HPC software stack:

- ❖ **Software distributed via CVMFS:** CC software stack (CC clusters), …
- ❖ **Local software:** legally restricted software (VASP, Gaussian, …)

Software Environment on CC clusters



User layer: Python packages, Perl and R modules, custom codes, …

Easybuild layer: modules for Intel, PGI, OpenMPI, CUDA, MKL, high-level applications. Multiple architectures (sse3, avx, avx2, avx512)
`/cvmfs/soft.computecanada.ca/easybuild/{modules,software}/2017`

Nix layer: GNU libc, autotools, make, bash, cat, ls, awk, grep, … etc.
`module nixpkgs/16.09 => $EBROOTNIXPKGS {$NIXUSER_PROFILE}`
`/cvmfs/soft.computecanada.ca/nix/var/nix/profiles/16.09`

Gray layer: SLURM, Lustre client libraries, IB / OmniPath client libraries (all dependencies of OpenMPI).

OS layer: kernel, daemons, drivers, libcuda, anything privileged (e.g. the sudo command): always local. Legally restricted software: VASP, Gaussian.

User

RSNT

Sys. Admin

RSNT: Software installation

- RSNT: Research and Support National Team
  - Installs and maintains software stack on Compute Canada clusters.
  - Write and maintain the documentation. +other contributions from CC-Staff.

- What software we install?
- Number-crunching software environment:
  - Compilers (GCC, Intel, PGI), BLAS, LAPACK, MKL, PETSc, GSL, HDF5, NetCDF, MPI, OpenMP, profilers, debuggers and other build tools, ... etc.
- Dynamic languages and interpreters: R, Python, Perl, Julia, ...
- Domain-specific applications and packages:
  - Engineering, Chemistry, Machine-Learning, Biomolecular, genomics, … etc.
- Some commercial and licensed software:
  - ANSYS, … controlled by POSIX groups, User license, … etc.

CC software stack: documentation

Available software:

730+ scientific applications

5,200+ permutations of version/arch/toolchain

- ❑ List of available modules:
  https://docs.computecanada.ca/wiki/Available_software
- ❑ List of Python wheels:
  https://docs.computecanada.ca/wiki/Available_wheels
- ❑ Main Compute Canada documentation:
  https://docs.computecanada.ca/wiki

How to find a given software?

- module avail; module purge
- module spider soft; module spider soft/version
- module load soft/version; module unload soft/version
- module show soft/version; module help soft/version
- module list
- module use ~/modulefiles; module unuse ~/modulefiles
- Documentation: https://lmod.readthedocs.io/
  - https://docs.computecanada.ca/wiki/Utiliser_des_modules/en

Local installation: user's directory

- ❑ Compute Canada provide a minimal installation of:
  - ❑ R and r-bundle-bioconductor as modules:
    - ✓ users can install the packages needed in their home directory.
  - ❑ Python as modules: python and scipy-stack
    - ✓ users can install the packages needed in their home directory.
    - ✓ Most used packages are provided as wheels.
  - ❑ Perl and bioperl as modules:
    - ✓ users can install the packages needed in their home directory.
- ❑ Other software installed locally:
  - ❑ Home made programs
  - ❑ Restricted and licensed software that can not be distributed via CVMFS.
  - ❑ Custom software: patch from a user, changing parts of the code, … etc.
  - ❑ Development version of a code, … etc.

https://docs.computecanada.ca/wiki/Installing_software_in_your_home_directory

Local installation: R packages

- ❑ R packages:

  rgdal, adegenet, stats, rjags, dplyr, … etc.

- ❑ Choose your module: module spider r
- ❑ Load R and dependencies (gdal, jags, gsl, udunits… etc):

  module load gcc/7.3.0 r/3.6.0

- ➢ Launch R and install the packages:

```
~$ R
> install.packages(”sp")
'lib =/cvmfs/soft.computecanada.ca/easybuild/{..}/R/library"' is not writable
Would you like to use a personal library instead? (yes/No/cancel) yes
Would you like to create a personal library ‘~/R/{…}’ to install packages into? (yes/No/cancel) yes

--- Please select a CRAN mirror for use in this session ---
> install.packages(”dplyr")
```

Local installation: python packages

- ❑ Check available wheels: avail_wheels <package>

  https://docs.computecanada.ca/wiki/Available_Python_wheels/en

- ❑ Chose your module: module spider python
- ❑ Load Python and dependencies; scipy-stack, … if needed:

```
~ $ module load gcc/7.3.0 python/3.7.4 scipy-stack/2019b
```

- ❑ Create & activate a virtual environment, install and test:

```
~ $ virtualenv /home/$USER/cutadapt_env
~ $ source /home/$USER/cutadapt_env/bin/activate
```

```
(cutadapt_env) ~ $ pip install cutadapt --no-index
(cutadapt_env) ~ $ python -c "import cutadapt" ; cutadapt --help
```

- ❑ For other programs: download, unpack and install using:

```
~$ pip install -r requirements.txt; python setup.py install
```

Local installation: Perl modules

- Example: Hash::Merge; Logger::Simple; MCE::Mutex; threads …
- Load Perl module: module load perl
- Install the first package using cpan:

```
~$ cpan install YAML
Would you like to configure as much as possible automatically? [yes] yes
What approach do you want?  (Choose 'local::lib', 'sudo' or 'manual')
[local::lib] local::lib
Would you like me to append that to /home/$USER/.bashrc now? [yes] yes
```

- Install the rest of the packages:

```
~$ cpan install Hash::Merge
~$ cpan install Logger::Simple
~$ cpan install MCE::Mutex
~$ cpan install threads
```

To make the changes available in your environment, run:

```
. ~/.bashrc
```

or logout and login again

Local installation: configure / cmake

Steps for building a software:

- Download the source files
- Load a compiler + dependencies; set environment variables if needed.
- Configure, build, test and install the code, set a module.

Using **configure**:

- Configure the code: ./configure --prefix=<path-to-install-dir> <+options>
- Build, test, install: make; make test {check}; make install

Using **cmake**:

- Create a build directory: mkdir build && cd build
- Configure the code:

cmake .. -DCMAKE_INSTALL_PREFIX=<path-to-install-dir> <+options>

- Build, test, install: make; make test {check}; make install

Introduction to EasyBuild: concept

- EasyBuild: a software build and installation framework.

  http://hpcugent.github.io/easybuild/

  - automates much of what you now do manually.
  - originated from Ghent University, Belgium.
  - Now, used by various sites worldwide:
    - including Compute Canada clusters.

- Three components:
  - framework: high level Python scripts.
  - easyblocks: is it configure; make; make install, cmake, custom?

    Python scripts ➔ used for more complexe software (WRF, … etc.)
  - easyconfigs: what are the configure parameters? (configuration files).

Introduction to EasyBuild: concept

- ❑ framework:
  - ❖ Core of easybuild that provide the main functions for building software
  - ❖ Unpacking sources, configuration, build, install, set the module, …etc.

- ❑ easyblocks: eb --list-easyblocks
  - ❖ Python scripts used for building a particular software.
  - ❖ Rely on framework: execute shells, run commands, obtain output, exit.

- ❑ extensions: additional add-ons: R, Python, Perl, Ruby.

- ❑ easyconfigs: eb --avail-easyconfig-params; eb -a; eb --list-software
  - ❖ Text files that contain values of key parameters supplied by the framework.
  - ❖ Provide module names (dependencies) that are loaded by the framework.
  - ❖ A copy of easyconfig is stored in the installation directory (successful inst.)

Working with EasyBuild: basics

- What do you need?
  - Access to eb command: already installed on all CC clusters (CVMFS).
  - Toolchains: compiler, MKL, OpenMPI, CUDA, … ~$ eb --list-toolchains
  - EasyBuild recipe: search for a recipe using ~$ eb -S <software name>
    - examples available on Compute Canada GitHub.
    - official GitHub for easybuild (may need to be adapted to CC environment).
  - Access to source files via network or locally:
    - EasyBuild can download the sources (if possible) or use the files from local directory.

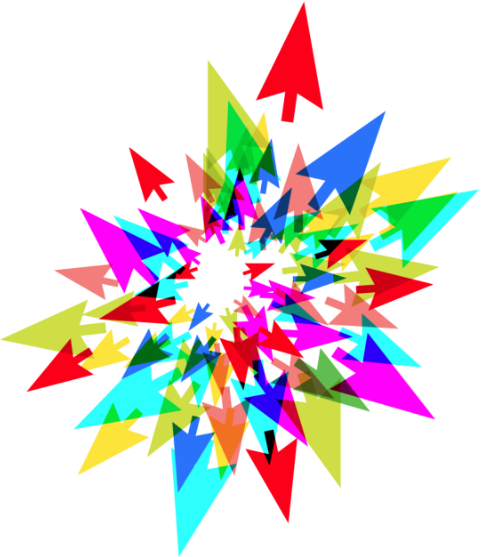
- Compiling with EasyBuild:
  - Use existing recipe (and customize it if needed); if not: write your own.
  - One recipe: for multiple software versions and different toolchains

- Syntax:
  - ~$ eb <recipe> <+options>

For more options: ~$ eb -- help

EasyBuild: Toolchains

- **Toolchains:** core modules in easybuild concept.
- **Combination of:**
  - Compiler: **g**cc, **i**ntel, **p**gi.
  - MPI implementation: openmpi, intel mpi
  - Math libraries: intel mkl, BLACS, ScaLAPACK, FFTW, …
  - CUDA: for GPU applications.
- **Available toolchains:**
  - iccifort, iompi, iompic, iimkl, iomkl, iomklc, … etc.
  - GCC, gmkl, gompi, pompi, … etc.



- **Most used toolchains on CC software stack:**
  - GCC,5.4.0, iccifort/iimkl/gmkl,2016.4, iompi/iomkl/gompi/gomkl, 2016.4.11 →(StdEnv/2016.4).
  - GCC,7.3.0, iccifort/iimkl/gmkl,2018.3, iompi/iomkl/gompi/gomkl, 2018.3.312 →(StdEnv/2018.3).

EasyBuild: easyconfig template

software-version-toolchain-toolchainversion.eb; GSL-2.4-GCC-7.3.0.eb

```
easyblock = 'ConfigureMake'

name = 'NAME'
version = 'VERSION’

homepage = 'http://www.example.com'
description = """TEMPLATE DESCRIPTION"""

toolchain = SYSTEM
sources = ['%(name)s-%(version)s.tar.gz']
source_urls = ['http://www.example.com']
patches = []
checksums = []
dependencies = []
sanity_check_paths = {
    'files': ['/bin/%(namelower)s'],
    'dirs': ["lib"]
}
moduleclass = 'phys'
```

ConfigureMake, CMakeMake, MakeCp, CmdCp, Binary, PackedBinary, Tarball, Bundle …

Software name + software version

Link to the home page + short description

Toolchain, Toolchain version, Toolchain options

sources, URL, patch, checksums, …

HDF5, FFTW, Boost, NetCDF, GSL, …

Sanity check on the installation directory

Category of the program: chem, bio, geo, data, …

Options: eb --avail-easyconfig-params; eb -a

```
toolchainopts = {}
builddependencies = []

preconfigopts = ' '
configopts    = [' ']
configopts += [' ']

skipsteps = [' ']

install_cmd = " "

postinstallcmds = [' ']

modextrapaths = {' '}
```

```
toolchainopts = {'usempi': True, 'openmp': True, 'cstd': 'c++11’}

builddependencies = [('CMake', '3.4.1')]

preconfigopts = ' export MKLPATH=$MKLROOT && '

configopts    = '-DBUILD_SHARED_LIBS=ON '
configopts += '-DBUILD_UTILITIES=ON '

skipsteps = ['install']

install_cmd = " ./install.sh && cp -r bin lib tests %(installdir)s"

postinstallcmds =  ['cp -r bin lib examples %(installdir)s']

modextrapaths = {'CPATH': 'include/voro++'}
```

Where to find EB recipe if there is any?

- **Online:**
  - https://github.com/ComputeCanada/easybuild-easyconfigs
  - https://github.com/easybuilders/easybuild-easyconfigs
  - Other contributors (online search)
- **Locally:**
  - Clone GitHub repository to explore the different recipes.
  - Search for a recipe using the command: `~$ eb -S <name of the program>`

If not, write your own:

- Check the documentation: https://easybuild.readthedocs.io/en/latest/
- Start using existing recipes to familiarize yourself with EB concept.
- If there is no recipe to use or to customize: write your own.

EasyBuild: checksums

- Easybuild syntax:
  - ~$ eb <recipe> <+opts>

  For more options: ~$ eb --help

- Build with disabling checksums:
  - Syntax: ~$ eb <recipe> --disable-enforce-checksums

- Add checksums manually:
  - **Use: sha256sum <sources>**
  - **Works also with: md5sum <sources>**
  - Add checksums = ['37dae3281b21213df237ca5e2973766c'] to your <recipe>.

- Add checksums with EB:
  - Syntax: ~$ eb <recipe> --inject-checksums (Does not build: it adds checksums).

Compile with EasyBuild: one or more options

- ❑ Build using: eb <recipe> <+options>
- ❑ Change a toolchain:
  - ✓ ~$ eb <recipe> --try-toolchain=GCC,7.3.0
  - ✓ ~$ eb <recipe> --try-toolchain=gmkl,2018.3
- ❑ Change the software version:
  - ✓ ~$ eb <recipe> --try-software-version=1.2.0
  - ✓ ~$ eb <recipe> --try-software-version=1.4.2
- ❑ Force the installation:
  - ✓ ~$ eb <recipe> --force
  - ✓ ~$ eb --rebuild <recipe>
- ❑ Keep the build directory:
  - ✓ ~$ eb <recipe> --disable-cleanup-builddir

--parallel = 8
--force
--rebuild
--robot
--disable-enforce-checksums
--inject-checksums
--fix-deprecated-easyconfigs

--installpath-modules=${}
--installpath-software=${}
--prefix=${install-dir}
--sourcepath=${path to src}

Custom path for the installation directory

Stata-15.eb

```
name = 'Stata'
version = '15'

homepage = 'https://www.stata.com/'
description = """Stata is a complete, integrated statistical software package."""

toolchain = SYSTEM

sources = ['Stata%(version)sLinux64.tar.gz']

dependencies = [('libpng', '1.2.58')]

postinstallcmds = ["/cvmfs/soft.computecanada.ca/easybuild/bin/setrpaths.sh --path %(installdir)s/"]

moduleclass = 'data'
```

Installation steps for Stata under project space

- ❑ By default: ~/.local/easybuild {modules; software; sources}
- ❑ In this example, the program STATA will be installed under project space and the module under home directory:

```
~$ installdir=/project/6012345/$USER
~$ moduledir=/home/$USER/.local/easybuild/modules/2017
~$ pathtosrc=/home/$USER/software
~$ eb Stata-15.eb --installpath-modules=${moduledir}
--prefix{--installpath-software}=${installdir} --sourcepath=${pathtosrc}
```

- ❑ Set the module for other members of the group:
  - ❖ share the installation directory (read and exec access).
  - ❖ copy '~/.local/easybuild/modules' to home directory of other members of the group.

EasyBuild by examples

- ADMIXTURE-1.3.0.eb
- BLAST+-2.10.0-GCC-7.3.0.eb
- Circos-0.69-6.eb
- DALTON-2018-iomkl-2016.4.11.eb
- DIAMOND-0.8.36-GCC-5.4.0.eb
- fastStructure-1.0-GCC-5.4.0.eb
- FastTree-2.1.10-GCC-5.4.0.eb
- GSL-2.4-GCC-5.4.0.eb
- Octave-5.1.0-gmkl-2018.3.eb
- PfamScan-1.6-GCC-7.3.0.eb
- RAxML-8.2.11-gompi-2016.4.11.eb
- Siesta-4.1-b2-iomkl-2016.4.11.eb
- Stata-15.eb

```
~$ eb ADMIXTURE-1.3.0.eb
~$ eb BLAST+-2.10.0-GCC-7.3.0.eb
~$ eb Circos-0.69-6.eb
~$ eb DALTON-2018-iomkl-2016.4.11.eb
~$ eb DIAMOND-0.8.36-GCC-5.4.0.eb
~$ eb fastStructure-1.0-GCC-5.4.0.eb
~$ eb FastTree-2.1.10-GCC-5.4.0.eb
~$ eb GSL-2.4-GCC-5.4.0.eb
~$ eb Octave-5.1.0-gmkl-2018.3.eb
~$ eb PfamScan-1.6-GCC-7.3.0.eb
~$ eb RAxML-8.2.11-gompi-2016.4.11.eb
~$ eb Siesta-4.1-b2-iomkl-2016.4.11.eb
~$ eb Stata-15.eb
```

GSL-2.4-GCC-5.4.0.eb

```
easyblock = 'ConfigureMake'

name = 'GSL'
version = '2.4’

homepage = 'http://www.gnu.org/software/gsl/'
description = """GNU Scientific Library (GSL)."""

toolchain = {'name': 'GCC', 'version': '7.3.0'}
toolchainopts = {'unroll': True, 'pic': True}

source_urls = [GNU_SOURCE]
sources = [SOURCELOWER_TAR_GZ]

moduleclass = 'numlib'
```

```
eb GSL-2.4-GCC-5.4.0.eb --force

eb GSL-2.4-GCC-5.4.0.eb --inject-checksums

eb GSL-2.4-GCC-5.4.0.eb

eb GSL-2.4-GCC-5.4.0.eb --try-toolchain=GCC,7.3.0

eb GSL-2.4-GCC-5.4.0.eb --try-toolchain=iccifort,2016.4

eb GSL-2.4-GCC-5.4.0.eb --try-toolchain=iccifort,2018.3

eb GSL-2.4-GCC-5.4.0.eb --try-toolchain=iccifort,2018.3 --try-software-version=2.5
```

DIAMOND-0.8.36-GCC-5.4.0.eb

```
easyblock = "CMakeMake"
name = 'DIAMOND'
version = "0.8.36"

homepage = https://github.com/bbuchfink/diamond'
description = """Accelerated BLAST"""

toolchain = {'name': 'GCC', 'version': '5.4.0'}
source_urls = ['https://github.com/bbuchfink/diamond/archive/']
sources = ['v%(version)s.tar.gz']

separate_build_dir = True

sanity_check_paths = {
    'files': ['bin/diamond'],
    'dirs': [],
}
moduleclass = 'bio'
```

```
eb DIAMOND-0.8.36-GCC-5.4.0.eb

eb DIAMOND-0.8.36-GCC-5.4.0.eb
--try-toolchain=GCC,7.3.0

eb DIAMOND-0.8.36-GCC-5.4.0.eb
--try-software-version=0.9.22

eb DIAMOND-0.8.36-GCC-5.4.0.eb
--try-toolchain=GCC,7.3.0 –try-
software-version=0.9.8
```

fastStructure-1.0-GCC-5.4.0.eb

```
easyblock = 'CmdCp'
name = 'fastStructure'
version = '1.0'

homepage = 'http://rajanil.github.io/fastStructure/'
description = """fastStructure is an algorithm for…"""

toolchain = {'name': 'GCC', 'version': '5.4.0'}
source_urls = ['https://github.com/rajanil/fastStructure/archive/']
sources = ['v%(version)s.tar.gz']
dependencies = [
    ('Python', '2.7.14'),
    ('SciPy-Stack', '2017b'),
    ('GSL', '2.3'),
]

cmds_map = [('.*', 'cd vars && python setup.py build_ext --
inplace && cd .. && python setup.py build_ext --inplace')]
```

```
files_to_copy = ['*']

postinstallcmds = [
    'echo "#!/bin/env python" | cat -
%(installdir)s/structure.py > temp && mv
temp %(installdir)s/structure.py',
    'chmod +x %(installdir)s/structure.py']

modextrapaths = {
    'PATH': [''],
    'PYTHONPATH': [''],
}

sanity_check_paths = {
    'files': ['structure.py'],
    'dirs': ['vars'],
}
moduleclass = 'bio'
```

DALTON-2018-iomkl-2016.4.11.eb

```
easyblock = 'CMakeMake'

name = 'DALTON'
version = "2018"

homepage = 'http://daltonprogram.org/'
description = """The Dalton suite consists of two separate
executables, Dalton and LSDalton."""

toolchain = {'name': 'iomkl', 'version': '2016.4.11'}
toolchainopts = {'usempi': True, 'openmp': True, 'pic': True}

sources = [{
    'filename': '%(namelower)s-release-%(version)s.tar.gz',
    'git_config': {
        'url': 'https://gitlab.com/dalton/',
        'repo_name': 'dalton',
        'commit': '07a00c83',
        'recursive': True,
    },
}]
```

```
separate_build_dir = True

configopts  = '-DCMAKE_BUILD_TYPE=release '
configopts += ' {followed by a long list of options …} '

postinstallcmds =  ['cd %(installdir)s/dalton && mkdir -p ../bin
&& mv dalton dalton.x ../bin/ && mv GIT_HASH VERSION
basis tools ../ && cd ../ && rm -rf dalton && chmod u+x tools/*
&& cp -r %(builddir)s/easybuild_obj/test %(installdir)s/']

sanity_check_paths = {
    'files': ['bin/dalton', 'bin/dalton.x', 'GIT_HASH'],
    'dirs': ['test', 'basis', 'tools'],
}

modextrapaths = {'PATH': ['basis', 'tools’]}
modextravars = {'BASLIB': '%(installdir)s/basis'}

moduleclass = 'chem'
```

RAxML-8.2.11-gompi-2016.4.11.eb

```
easyblock = 'MakeCp'

name = 'RAxML'
version = '8.2.11'

homepage = 'https://github.com/stamatak/standard-RAxML'
description = "RAxML search algorithm for maximum
likelihood based inference of phylogenetic trees."

toolchain = {'name': 'gompi', 'version': '2016.4.11'}
toolchainopts = {'usempi': True}

sources = ['v%(version)s.zip']
source_urls = ['https://github.com/stamatak/standard-
RAxML/archive/']

buildopts = '-f Makefile.MPI.gcc CC="$CC"'
```

```
files_to_copy = [(["raxmlHPC-MPI"], "bin"),
"usefulScripts", "README", "manual"]

postinstallcmds =  ['ln -sf
%(installdir)s/bin/raxmlHPC-MPI
%(installdir)s/bin/raxmlHPC && chmod u+x
%(installdir)s/usefulScripts/*.*']

modextrapaths = {'PATH': 'usefulScripts'}

sanity_check_paths = {
    'files': ["bin/raxmlHPC-MPI"],
    'dirs': [],
}

moduleclass = 'bio'
modluafooter = """
depends_on("perl")
"""
```

Short demonstration on a cluster

- ADMIXTURE-1.3.0.eb
- BLAST+-2.10.0-GCC-7.3.0.eb
- Circos-0.69-6.eb
- DALTON-2018-iomkl-2016.4.11.eb
- DIAMOND-0.8.36-GCC-5.4.0.eb
- fastStructure-1.0-GCC-5.4.0.eb
- FastTree-2.1.10-GCC-5.4.0.eb
- GSL-2.4-GCC-5.4.0.eb
- Octave-5.1.0-gmkl-2018.3.eb
- PfamScan-1.6-GCC-7.3.0.eb
- RAxML-8.2.11-gompi-2016.4.11.eb
- Siesta-4.1-b2-iomkl-2016.4.11.eb
- Stata-15.eb

- Some useful EB commands:
  - search for recipe
  - list of parameter
  - help
- Install GSL-2.4 with GCC-5.4.0
- Install GSL-2.4 with GCC-7.3.0
- Install GSL-2.5 with GCC-5.4.0
- Install RAxML-8.2.11 with gompi-2016.4.11
- Install RAxML-8.2.11 with iompi-{2016.4.11,2018.3.312}
- Install ADMIXTURE-1.3.0
- Install DIAMOND-0.8.36
- Install DIAMOND-0.9.22

Some links and documentation

- https://github.com/ComputeCanada/easybuild-easyconfigs
- https://github.com/ComputeCanada/easybuild-easyblocks
- https://github.com/ComputeCanada/easybuild-framework

- https://github.com/easybuilders/easybuild-easyconfigs
- https://github.com/easybuilders/easybuild-easyblocks
- https://github.com/easybuilders/easybuild-framework

- http://hpcugent.github.io/easybuild/
- https://easybuild.readthedocs.io/en/latest/

- https://lmod.readthedocs.io/

- https://docs.computecanada.ca/wiki/Utiliser_des_modules/en
- https://docs.computecanada.ca/wiki/Compute_Canada_Documentation

Getting support

EasyBuild:

Website: https://easybuilders.github.io/easybuild/
Mailing list: https://lists.ugent.be/wws/info/easybuild

Compute Canada support contacts:

support@computecanada.ca for the general support

Documentation and Training:

Compute Canada: https://docs.computecanada.ca
Westgrid website: https://www.westgrid.ca
Westgrid Training Events calendar: https://www.westgrid.ca/events
Westgrid Training material: https://westgrid.github.io/trainingMaterials/

Acknowledgments

Thanks to: RSNT (Research and Support National Team), CVMFS team, other contributors from Compute Canada

Thanks to EasyBuild: UGent, JSC, Robert Schmidt, ...