EasyBuild: Building Software With Ease.

Jens Timmerman Stijn De Weirdt Kenneth Hoste

Department ICT
Ghent University
easybuild@lists.ugent.be

HEPiX Spring 2012 Workshop

Outline

Outline

UNIVERSITEIT GENT

Not all end-users can build the software they use.

- Compilation requires lots of expertise and specific knowledge.
  - Not all documentation is created equal.
  - Not all software has a straightforward build procedure.
- Collaboration and sharing the knowledge is problematic at best.

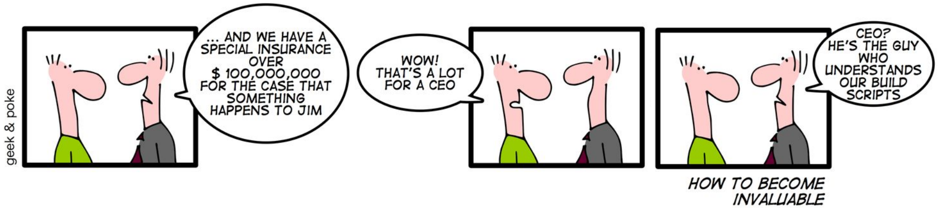
geek & poke
... AND WE HAVE A SPECIAL INSURANCE OVER $ 100,000,000 FOR THE CASE THAT SOMETHING HAPPENS TO JIM
WOW! THAT'S A LOT FOR A CEO
CEO? HE'S THE GUY WHO UNDERSTANDS OUR BUILD SCRIPTS
HOW TO BECOME INVALUABLE

Figure: http://geekandpoke.typepad.com/

Installations should be automated.

- Building/installing newer versions.
- Using different compilers and/or libraries.
- Using different compiler flags.
- Build on different OS and/or hardware.

Installations should be managed.

The installation should be:

- Reproducable
  - Reproduce the installation on a new system.
  - Share the automated installation procedure and/or allow collaboration.
- Verified
  - Verify executable and libraries.
  - Verify correctness: This requires end-user expertise.
- Able to provide multiple versions of the same package.

Outline

Software installation framework

- Uses specification files (.eb) files which describe a software package.
  - name, version, website, description
  - installation files (sources), patches, download location
  - compiler toolkit to build with
  - optional build parameters
  - ...
- Allows for diverting from default 'configure/ make' through extensions called *easyblocks*.
- Builds/install binaries, creates module files
- Written in Python

Software installation framework

- Uses specification files (.eb) files which describe a software package.
  - name, version, website, description
  - installation files (sources), patches, download location
  - compiler toolkit to build with
  - optional build parameters
  - ...
- Allows for diverting from default 'configure/ make' through extensions called *easyblocks*.
- Builds/install binaries, creates module files
- Written in Python

UNIVERSITEIT GENT

Software installation framework

- Uses specification files (.eb) files which describe a software package.
  - name, version, website, description
  - installation files (sources), patches, download location
  - compiler toolkit to build with
  - optional build parameters
  - ...
- Allows for diverting from default 'configure/ make' through extensions called *easyblocks*.
- Builds/install binaries, creates module files
- Written in Python

Software installation framework

- Uses specification files (.eb) files which describe a software package.
  - name, version, website, description
  - installation files (sources), patches, download location
  - compiler toolkit to build with
  - optional build parameters
  - ...
- Allows for diverting from default 'configure/ make' through extensions called *easyblocks*.
- Builds/install binaries, creates module files
- Written in Python

Key properties

- Allows for sharing and reusing the easyblocks and .eb specification files.
  - Repeat the build process on a separate system with a single command
- Installs different versions of a program next to each other.
- Save .eb specification files under version control.
- Keep track of installation logs and build statistics.

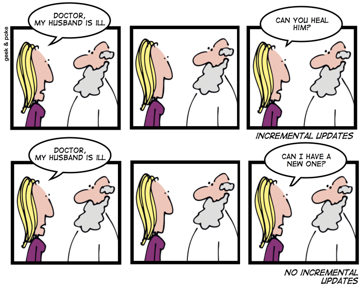

Figure: http://geekandpoke.typepad.com/

Outline

Environment Modules

Definition

Environment modules allows the dynamic modification of a user's environment via modulefiles.
Each modulefile contains the information needed to configure the shell for an application. Typically modulefiles instruct the module command to alter or set shell environment variables such as PATH, MANPATH, etc.
`http://modules.sourceforge.net/`

Used for:

- The ability to use multiple versions of a software package.
- Track and load dependencies
- Making the software available to users in a transparent way

Module file example

```
#%Module
proc ModulesHelp { } {  puts stderr { Python is a programming language ...} }
module-whatis {Python is a programming language ...}
set root    /home/jens/easybuild/software/Python/2.7.3-goalf-1.1.0-no-OFED
conflict    Python
if { ![is-loaded goalf/1.1.0-no-OFED] } {
    module load goalf/1.1.0-no-OFED
}
prepend-path LD_LIBRARY_PATH $root/lib
prepend-path MANPATH $root/man
prepend-path MANPATH $root/share/man
prepend-path PATH $root/bin
setenv SOFTROOTPYTHON $root
setenv SOFTVERSIONPYTHON 2.7.2
```

Prerequisites

- Python ≥ 2.4 and < 3.0 (for now)
- Environment Modules (requires Tcl)
- Optionally:
  - GitPython (recommended)
  - pysvn

Usage

- Create (or download) a .eb specification file and easyblock (if needed).
- Repeat for all dependencies:

```
./easybuild.sh package-version.eb
```

- After the installation has completed successfully you will have:
  - binaries and libraries
  - module files
  - .eb files under version control.

What EasyBuild does for you

- Generate installation path and create build directory
- Resolve dependencies
  - Is there a module for it? If not, build dependency first.
- Unpack source (optionally try to download it first)
- Apply patches
- Configure build
- Build
- Test
- Install
- Install extra packages
  - e.g, add Python packages like scipy and numpy.
- Generate module files
- Clean up (build dir, temporary log files, ...)
- Verify installation (not there yet)

UNIVERSITEIT GENT

Quick demo

After unpacking EasyBuild run:

```
export EBHOME="<path to where you unpacked EasyBuild>"
${EBHOME}/easybuild.sh --robot ${EBHOME}/easybuild/easyconfigs \
${EBHOME}/easybuild/easyconfigs/h/HPL/HPL-2.0-goalf-1.1.0.eb
```

This bootstraps GCC (requires a c/c++ compiler on the system) and builds OpenMPI, LAPACK, ATLAS, BLACS, ScaLAPACK, FFTW (goalf toolkit) and finally HPL.

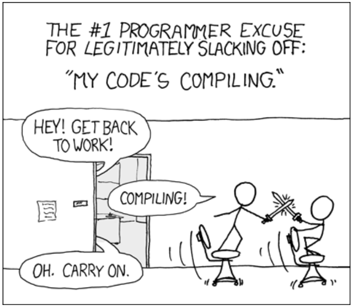

Figure: http://xkcd.com/

Example EasyBuild config

```
buildPath = “/tmp/easybuild/build/”
installPath = “/usr/easybuildapps”
sourcePath = “/usr/easybuildsrc”
#requires GitPython
repositoryType = ’git’
repositoryPath = ("ssh://user@server/path/to/repo.git",
                  "path/inside/repo")
logFormat = ("easybuildlog",
             "easybuild-%(name)s-%(version)s-%(date)s.%(time)s.log")
```

Example .eb File

```
name = 'HPL'
version = '2.0'
homepage = 'http://www.netlib.org/benchmark/hpl/'
description = """HPL is a software package that solves a (random) dense linear
    system in double precision (64 bits) arithmetic on distributed-memory
    computers. It can thus be regarded as a portable as well as freely
    available implementation of the High Performance Computing Linpack
    Benchmark."""
toolkit = {'name':'goalf', 'version':'1.1.0-no-OFED'}
toolkitopts = {'optarch':True, 'usempi':True}
sources = ['%s-%s.tar.gz' % (name.lower(),version)]
sourceURLs = ['http://www.netlib.org/benchmark/%s' % name.lower()]
## fix Make dependencies, so parallel build also works
patches = ['HPL_parallel-make.patch']
```

Current state

- In use for over 3 years, we build **all** end-user software with it
- Framework cleaned up and released as open source begin April 2012.
  - `https://github.com/hpcugent/easybuild`

- Very well tested under Scientific Linux (SL) 5 and 6.
- Framework also tested under:
  - Fedora 16
  - Ubuntu 10.04 and 11.10
  - Suse
  - Arch Linux
  - OS X 10.7.3

UNIVERSITEIT GENT

List of currently supported software packages (>250)

ABAQUS ABINIT ABySS ADF ALLPATHS-LG ARPACK ATSAS AmberTools Armadillo AutoDock-Vina Autoconf Aztec BEAGLE BEAST BLACS BLAST BWA BamTools BiSearch BigDFT Biopython Blat Blitz++ Boost Bowtie CAMFR CAPHE CCfits CD-HIT CFITSIO CLHEP CLooG-PPL CMake COMSOL **CP2K** CPLEX CPMD Cabal Charm++ ClustalW ClustalW2 Cufflinks DIANA DOLFIN Denoiser EMAN EPD ETSFIO F77SPLIT FASTX FFC FFTW FIAT FLAME FLUENT FastTree Ferret Flume GAMESS-US GATE GCC GDAL GDB GEANT4 GEOS GHC GLPK GMP GRASS GROMACS GSL Gambit **Gaussian** GenomeAnalysisTK GotoBLAS HBase HDF5 HOD HPCC Hadoop Harminv Hive Hoard IMOD IPM IPython ISIS Insight Instant IronPython JUnit JasPer Java JumboMem LAPACK LMF LS-DYNA Libint M4 MATLAB MEME METIS MPB MPFR MPITB MUMPS MVAPICH2 Maple Maven Meep Mono Mpc MrBayes MyMediaLite MySQL NAMD NCL NCO NWChem OPT++ ORCA Octave Open64 OpenCV **OpenFOAM** OpenMPI PCRE PEST PETSc PHENIX PHP PPL PROJ PVM ParMETIS Parallel-netCDF Path64 Perl PostgreSQL Primer3 PyNAST Pypar Python QIIME Qhull QuEST Quantum-ESPRESSO R RAPTOR RDP-Classifier ROOT ROSETTA Revo Rpy Ruby SAMtools SAS SCOTCH SCons SGA SHELXL SOAPdenovo SPARSKIT SPIDER SPRNG SWIG Sage ScaLAPACK ScientificPython SeqtrimNEXT Silo SuiteSparse Szip Tcl TclTk Tk TopHat Treefinder Trinity UCLUST UFC UFL UNAFold **VASP** Viper **WIEN2k** WRF Wannier90 Xmipp Y12M YASM ant bcMPI boost-numeric-bindings bzip2 cURL cdbtools fastahack ffmpeg g2lib galib giflib gimkl gmgfl gnuplot gogfl gomkl google-sparsehash gtk-sharp gzip h5utils iSight icc ictce ifort imkl impi ipp itac jobs libctl libgtextutils libmatheval libpng libsmm libxc libxml2 libxslt likwid maui mhash microbiomeutil mpiBLAST mpiGraph netCDF orc picard ploticus pyTables pysam python-meep qrupdate readline schroedinger tbb unixODBC x264 yasm zlib

Summary

- EasyBuild allows to easily reproduce software builds/installations and install multiple versions.
- Feedback is very welcome, contributions even more:
  - easybuild@lists.ugent.be
  - `https://github.com/hpcugent/easybuild`
- Outlook
  - We are currently cleaning up easyblocks and releasing them on github regularly.