Installing software using EasyBuild in your cluster home

Ata Roudgar
Simon Fraser University
Compute Canada

Why do we need to install a software in home?

1. Users who run a software that has some config files and need to be customized
2. Users who needs to upgrade a software
3. Users who need to modify the source code

Types of installation, globally vs. locally

- Globally
  - software is installed on a path where anyone have access to, e.g., CVMFS (/cvmfs directory)
  - by admins
  - users cannot modify or delete it
  - Installed only once for all users
- In user's home
  - Installed in user's home directory (easybuild -> $HOME/.local/easybuild directory)
  - by user or admin
  - User can modify or delete files
  - By default, it is accessible only by the user

Traditional method of installing

- Download the source file
- If software comes with configure file, run ./configure with proper arguments.
- Use commands like "make" and "make install"
- Most of the time it is a complicated process
- Not always reproducible particularly when compiler or dependencies change version.
- Requires a good knowledge about options that are used to compile the software

Easybuild a software build framework

Easybuild

Easybuild a software build and installation framework

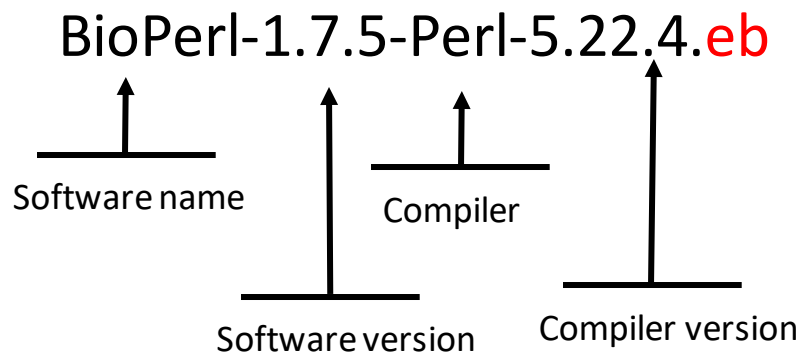
- It is written in Python
- allows for easily reproducing previous builds
- keep the software build recipes simple and human-readable
- **enables sharing with the HPC community**
- supports co-existence of versions/builds via dedicated installation prefix and module files
- retain logs for traceability of the build processes
- It still complicated but complication part is done by admin when install for the first time.

Easybuild

Easybuild recipe (easyconfig)

- Easybuild recipe is a text file (in python syntax) to instruct easybuild how to install a software
- Creating a recipe from scratch could be very challenging
- Compute Canada collected huge number of recipes at https://github.com/ComputeCanada/easybuild-easyconfigs/tree/computecanada-main/easybuild/easyconfigs
- Upon successful installation a copy of the recipe is copied at <installdir>/easybuild.
- Standard easybuild recipe filename format:

<software name>-<version>-<compiler>-<compiler version>.eb



Easybuild recipe

Basic concept of easybuild recipe parameters

`name`: The name of the software. %(name)s to call its value

`Version`: The version of the software. %(version)s to call its value

`Homepage`: The homepage of the software

`Description`: A short description about the software. This description is the same as the one at https://docs.computecanada.ca/wiki/Available_software

`source_urls`: The website where easybuild loo for to download the source file

`Source`: The name of the source file (in zip or tar.gz format)

`Builddependencies`: software dependencies at installation time

`sanity_check_paths`: check to see if such executable files exist after installation. Paths are given with respect to the install directory

`moduleclass`: bio, math, chem, phys, geo, vis

```
easyblock = 'ConfigureMake'

name = 'BLAST+'
version = '2.4.0'

homepage = 'http://blast.ncbi.nlm.nih.gov/'
description = """Basic Local Alignment Search Tool, or BLAST, is an
algorithm for comparing primary biological sequence information, such as
the amino-acid sequences of different proteins or the nucleotides of DNA
sequences."""

toolchain = {'name': 'GCC', 'version': '7.3.0'}

source_urls =
['http://ftp.ncbi.nlm.nih.gov/blast/executables/blast+/%(version)s/']
sources = ['ncbi-blast-%(version)s+-src.tar.gz']
patches = ['%(name)s-%(version)s_boost_backport.patch']

builddependencies = [
('Boost', '1.54.0'),
('Python', '2.7.14')
]

configopts = " --with-64 --with-bin-release --without-debug --with-mt"
sanity_check_paths = {
    'files': ['bin/blastn', 'bin/blastp', 'bin/blastx'],
    'dirs': []
}
moduleclass = 'bio'
```

Easybuild

Process of installation

- Command to run easybuild:

$ eb [path]/<recipe> [options]

- Download the source file
- Compare checksum to check if downloaded files altered or corrupted
- Apply patch files
- Build (compile)
- Copy the executables to installation path, %(installdir)s
- Check to see if expected executable file exist
- Make module
- End

Useful Template names/values

%(name)s	Name of the software
%(namelower)s	lower case of name
%(version)s	Version of the software
%(version_major)s	major version
%(version_minor)s	minor version
%(builddir)s	Build directory
%(installdir)s	Installation directory

Example: name='FastANI'
Version='3.2.1'

%(namelower)s : 'fastani'
%(version_major)s : '3.2'

Module command

- to load a software
  - Add "%(installdir)s/bin" where executable files are located to default searching path (PATH)
  - Add "%(installdir)s/lib" to LIBRARY_PATH variable
  - And some other variables....
- Upon a successful module command the following environmental variable is also setup:

EBROOT<name of the software in capital>

And it will contain the path to install dir.

Example:

```
$ module load gcc/9.3.0  openmpi/4.0.3 gromacs/2021.4
$ echo $EBROOTGROMACS
$/cvmfs/soft.computecanada.ca/easybuild/software/2020/avx2/MPI/gcc9/openmpi4/gromacs/2021.4
```

In this example we loaded gromacs-2021.4 that is installed globally using module command

Exercises

1. Reinstalling a software in user's home
2. Upgrading a software in user's home
3. Modifying a software in user's home

1. Reinstalling a software

- The software is already installed globally in /cvmfs
- Install the software on $HOME/.local/easybuild/ as well
- We need a copy of software recipe and patch files to run easybuild

reinstalling software orthomcl

- **Load the software, orthomcl**

```
$ module spider orthomcl
# the output shows that orthomcl-2.0.9 is the latest
$ module spider orthomcl/2.0.9
# orthomcl/2.0.9 has no dependency modules
$ module load orthomcl/2.0.9
```

- **Find install directory**

```
$ echo $EBROOTORTHOMCL
/cvmfs/soft.computecanada.ca/easybuild/software/2020/Core/orthomcl/2.0.9
```

- **Locating easybuild recipe**

```
$ ls $EBROOTORTHOMCL/easybuild/*.eb
/cvmfs/soft.computecanada.ca/easybuild/software/2020/Core/orthomcl/2.0.9/easybuild/orthomcl-2.0.9.eb
```

- **Creating a directory and copy the recipe and patch files**

```
$ mkdir -p ~/easybuild/reinstall
$ cp /cvmfs/soft.computecanada.ca/easybuild/software/2020/Core/orthomcl/2.0.9/easybuild/orthomcl-2.0.9.eb ~/easybuild/reinstall
$ cp /cvmfs/soft.computecanada.ca/easybuild/software/2020/Core/orthomcl/2.0.9/easybuild/*.patch ~/easybuild/reinstall
```

reinstalling software orthomcl

- run easybuild to build and install the software

```
$ cd ~/easybuild/reinstall
$ eb –f ./orthomcl-2.0.9.eb
# option –f is used to force easybuild to install on home
== Temporary log file in case of crash /tmp/eb-dydsorh6/easybuild-cdf1oxk5.log
== Running parse hook for orthomcl-2.0.9.eb...
== Running parse hook for GCC-9.3.0.eb...
== Running parse hook for GCCcore-9.3.0.eb...
Changing postinstallcmds from: []
New postinstallcmds: ["find %(installdir)s -name '*.la' -delete"]
== Running parse hook for MCL-14.137-GCC-9.3.0.eb...
== Running parse hook for GCC-9.3.0.eb...
== processing EasyBuild easyconfig /home/aroudgar/easybuild-recipes/orthomcl-2.0.9.eb
== building and installing Core/orthomcl/2.0.9...
== fetching files...
== creating build dir, resetting environment...
== unpacking...
== patching...
== preparing...
== Running pre-prepare hook...
== Running post-prepare hook...
== ... (took 3 secs)
== configuring [skipped]
== building [skipped]
== testing...
== installing...
== taking care of extensions...
== restore after iterating...
== postprocessing...
== sanity checking...
== ... (took 7 secs)
== cleaning up...
== creating module...
== Running post-module hook...
== ... (took 7 secs)
== permissions...
== packaging...
== COMPLETED: Installation ended successfully (took 22 secs)
== Results of the build can be found in the log file(s) /home/aroudgar/.local/easybuild/software/2020/Core/orthomcl/2.0.9/easybuild/easybuild-orthomcl-2.0.9-20211117.155541.log
== Build succeeded for 1 out of 1
== Running end hook...
== Temporary log file(s) /tmp/eb-dydsorh6/easybuild-cdf1oxk5.log* have been removed.
== Temporary directory /tmp/eb-dydsorh6 has been removed.
```

reinstalling software orthomcl

We now have two copies of orthomcl, one installed in home, and one installed in cvmfs

We run:

```
$ module unload orthomcl/2.0.9
$ module load orthomcl/2.0.9
```

Which one is loaded?

```
$ echo $EBROOTORTHOMCL
/home/aroudgar/.local/easybuild/software/2020/Core/orthomcl/2.0.9
```

home installed softwares have more priority to be loaded by module command

2. Upgrading a software, FastANI

- Recipe

`FastANI-1.32-GCC-9.3.0.eb`

```
name = 'FastANI'
version = '1.32'

source_urls = ['https://github.com/ParBLiSS/FastANI/archive']
sources = ['v%(version)s.zip']
patches = ['FastANI-1.32-memcpy.patch']
checksums = [
  '06a3bd38840526eabf7a99d4f10b8f05edbf5b96bf4301e37afd89f4c1ff6d81',  # v1.32.zip
  'eebcf0b64c31ee360ca79136f644157064ac69747ed13cff70f5c9932c6bb0d5',  # FastANI-1.32-memcpy.patch
]
```

Upgrading a software

Checksum:

- Every file has a unique digital signature
- Command **sha256sum** is used to calculate the checksum of a file

```
$ sha256sum <file name>
```

- Download the source file. E.g., https://github.com/ParBLiSS/FastANI/archive/v1.32.zip
- Run the command

```
$ sha256sum  v1.32.zip
06a3bd38840526eabf7a99d4f10b8f05edbf5b96bf4301e37afd89f4c1ff6d81  v1.32.zip
```

- Add the output of the command to your recipe
- Checksum in recipe contains the digital signature of source and patch files. The first one belong to the source code, the rest belongs to patch files

Upgrading software, FastANI

- **Load the latest version of software "FastANI"**

```
$ mkdir ~/easybuild/upgrade
$ cd ~/easybuild/upgrade
$ module spider fastani
# the output shows that fastani/1.32 is the latest
$ module spider fastani/1.32
   .
   .
   StdEnv/2020  gcc/9.3.0
   StdEnv/2020  intel/2020.1.217
   .
   .

# FastANI compiled with two compiler, GCC and INTEL
$ module load gcc/9.3.0 fastani/1.32
```

Upgrading software, FastANI

- **Find install directory**

```
$ echo $EBROOTFASTANI
/cvmfs/soft.computecanada.ca/easybuild/software/2020/avx2/Compiler/gcc9/fas
tani/1.32
```

- **Locating easybuild recipe**

```
$ ls $EBROOTSPADES/easybuild/*.eb
/cvmfs/soft.computecanada.ca/easybuild/software/2020/avx2/Compiler/gcc9/fastani/1.32/easybuild
/FastANI-1.32-GCC-9.3.0.eb
```

- **Copy the recipe and patch files**

```
$ cp /cvmfs/soft.computecanada.ca/easybuild/software/2020/avx2/Compiler/gcc9/fastani/1.32/easybuild/*.eb  ./
$ cp /cvmfs/soft.computecanada.ca/easybuild/software/2020/avx2/Compiler/gcc9/fastani/1.32/easybuild/*.patch
 ./
```

Upgrading software, FastANI

- Inspect the recipe using "vi" and modify it. The current version of fastANI software is 1.33
  I. Checksum
     I. Download the source code using **wget** command
        ```
        $ wget https://github.com/ParBLiSS/FastANI/archive/v1.33.zip
        ```
     II. Calculate checksum
        ```
        $ sha256sum v1.33.zip
        fb392ffa3c7942091a06de05c56c8e250523723c39da65ae8f245e65e440a74b   v1.33.zip
        ```
     III. Insert it in the recipe
        ```
        checksums = [
            'fb392ffa3c7942091a06de05c56c8e250523723c39da65ae8f245e65e440a74b',  # v1.33.zip
            'eebcf0b64c31ee360ca79136f644157064ac69747ed13cff70f5c9932c6bb0d5',  # FastANI-1.32 memcpy.patch
        ]
        ```
  II. version=1.32 -> version=1.33
     I. Save it under the name **FastANI-1.33-GCC-9.3.0.eb**

- run easybuild to build and install the software

```
$ eb -f ./FastANI-1.33-GCC-9.3.0.eb
```

3. Modifying and installing a software, FastANI

`FastANI-1.32-GCC-9.3.0.eb`

```
name = 'FastANI'
version = '1.32'

source_urls = ['https://github.com/ParBLiSS/FastANI/archive']
sources = ['v%(version)s.zip']
patches = ['FastANI-1.32-memcpy.patch']
checksums = [
  '06a3bd38840526eabf7a99d4f10b8f05edbf5b96bf4301e37afd89f4c1ff6d81',  # v1.32.zip
  'eebcf0b64c31ee360ca79136f644157064ac69747ed13cff70f5c9932c6bb0d5',  # FastANI-1.32-memcpy.patch
]
```

Patch file

- A patch file is a text file that consists of a list of differences
- Command **`diff`** is used to generate the list of differences with respect to original file name (1st argument)

  `$ diff –u <original file name> <recent file name>`

- Example of a patch file:

  `$ cat FastANI-1.32-memcpy.patch`

Modifying and installing software, FastANI

```
--- Makefile.in.orig    2019-07-25 18:01:28.000000000 +0200
+++ Makefile.in 2019-10-17 15:49:27.034833116 +0200
@@ -6,7 +6,7 @@
ifeq ($(UNAME_S),Darwin)  #macOS clang
       CXXFLAGS += -mmacosx-version-min=10.7 -stdlib=libc++ -Xpreprocessor -fopenmp -lomp
else
-       CXXFLAGS += -include src/common/memcpyLink.h -Wl,--wrap=memcpy
+       CXXFLAGS += -include src/common/memcpyLink.h
       CFLAGS += -include src/common/memcpyLink.h
endif
```

Modifying and installing a software, FastANI

I. Prepare recipe, directory and original source files
II. Modify file(s)
III. Use command **`diff`** to generate a patch file
IV. Calculate checksum of the patch file using `sha256sum command`
V. Add patch file in the recipe
VI. Install the modified version of the software

I. Prepare original recipe, directory and source files

```
$ mkdir ~/easybuild/modify
$ cd ~/easybuild/modify
```

We copy **`FastANI-1.32-GCC-9.3.0.eb`** and **`FastANI-1.32-memcpy.patch`** from install directory to this directory (exactly as we did for exercise 2)

```
$ module load gcc/9.3.0 fastani/1.32
$ cp $EBROOTFASTANI/easybuild/*.eb ./
$ cp $EBROOTFASTANI/easybuild/*.patch ./
```

Modifying and installing software, FastANI

```
#   using command wget to download the source file
$   wget https://github.com/ParBLiSS/FastANI/archive/v1.32.zip
#   unzip the file using command unzip
$   unzip v1.32.zip
$   ls -l
drwxr-x--- 5 aroudgar aroudgar    4096 Sep 30  2020 FastANI-1.32
$   cd FastANI-1.32/src/cgi
$   ls -l
rw-r----- 1 aroudgar aroudgar 4234 Sep 30  2020 core_genome_identity.cpp
```

Modifying and installing software, FastANI

II. Modify the file `core_genome_identity.cpp`

```
# first, we make a copy of the original file
$ cp core_genome_identity.cpp core_genome_identity.cpp.orig

$  vi core_genome_identity.cpp
#  Line 32 we add

unsetenv((char *)"MALLOC_ARENA_MAX");
/* This is just a test*/
 CommandLineProcessing::ArgvParser cmd;

#   Save the file and go back to top directory of the source file
$   cd ../../
```

Modifying and installing software, FastANI

III. Use command diff to generate a patch file

```
$  diff -u src/cgi/core_genome_identity.cpp.orig src/cgi/core_genome_identity.cpp > ../FastANI-test-1.32.patch
$  cd ..
$  ls -l
```

```
drwxr-x--- 5 aroudgar aroudgar    4096 Sep 30  2020 FastANI-1.32
-rw-r----- 1 aroudgar aroudgar     394 Dec  2 15:38 FastANI-test-1.3.patch
-rw-r----- 1 aroudgar aroudgar     519 Dec  5 23:38 FastANI-1.32-memcpy.patch
 -rw-r----- 1 aroudgar aroudgar 2965327 Dec  5 23:41 v1.32.zip
```

Modifying and installing software, FastANI

- $ cat FastANI-test-1.32.patch

```
--- src/cgi/core_genome_identity.cpp.orig       2021-12-05 23:42:18.000000000 -0800
+++ src/cgi/core_genome_identity.cpp    2021-12-05 23:43:30.000000000 -0800
@@ -29,7 +29,7 @@
   * for efficient multi-threaded execution
   */
  unsetenv((char *)"MALLOC_ARENA_MAX");
-
+  /* This is just a test*/
  CommandLineProcessing::ArgvParser cmd;
  using namespace std::placeholders;  // for _1, _2, _3...
```

Modifying and installing software, FastANI

IV. Calculate checksum of the patch file using sha256sum command

```
$  sha256sum FastANI-test-1.3.patch
 e2c71fe73fc66e00d9af2842d7edc2232136ecb5bb99a2a1857796c3b7900fea FastANI-test-1.3.patch
```

Modifying and installing a software, FastANI

V. Add patch file to the recipe

FastANI-1.32-GCC-9.3.0.eb

```
easyblock = 'ConfigureMake'

name = 'FastANI'
version = '1.32'

homepage = "https://github.com/ParBLiSS/FastANI"
description = """FastANI is developed for fast alignment-free computation of
whole-genome Average Nucleotide Identity (ANI). ANI is defined as mean
nucleotide identity of orthologous gene pairs shared between two microbial
genomes. FastANI supports pairwise comparison of both complete and draft
genome assemblies."""

toolchain = {'name': 'GCC', 'version': '9.3.0'}

source_urls = ['https://github.com/ParBLiSS/FastANI/archive']
sources = ['v%(version)s.zip']
patches = ['FastANI-test-1.3.patch', 'FastANI-1.32-memcpy.patch']
checksums = [
  '06a3bd38840526eabf7a99d4f10b8f05edbf5b96bf4301e37afd89f4c1ff6d81', # v1.32.zip
  'e2c71fe73fc66e00d9af2842d7edc2232136ecb5bb99a2a1857796c3b7900fea', #FastANI-test-1.3.patch
  'eebcf0b64c31ee360ca79136f644157064ac69747ed13cff70f5c9932c6bb0d5', # FastANI-1.32-memcpy.patch
]
```

Modifying and installing software, FastANI

VI. Install the modified version of the software

```
$ eb -f ./FastANI-1.32-GCC-9.3.0.eb
```

Questions?