Moving Forward to New Linux Distributions

Mark Ito
GlueX Software Meeting
November 24, 2020

Outline

- Background
- Python-2/Python-3 Compatibility
- Creating Singularity Containers
- Future architectures for running GlueX jobs

Background

- Upgrading our default versions of Geant4, requires upgrading compilers.
  - Currently using GCC 4.8.5 (June 2015)
  - Geant4: we use G4 10.02.p02 (June 2016), Richard has gone forward to 10.06.p01 (February 2020), that needs GCC 4.9.3 (also June 2015) or higher
- We need to start converting to Python 3.
  - Python 2 no longer supported as of January 2020
  - Python 3.0 was released in December 2008
- Upgraded versions of GCC incompatible with ROOT 6.08.06 (March 2017), our current default, and vice versa
  - Latest version of GCC is 10.2 (July 2020)
  - Latest version of ROOT is 6.22.02 (August 2020)

How to Proceed?

1. Deploy upgraded versions of selected software via modules (or similar mechanism)
    - This is how the Computer Center is proceeding
2. Run in containers
    - We are already committed to this approach for off-site production running

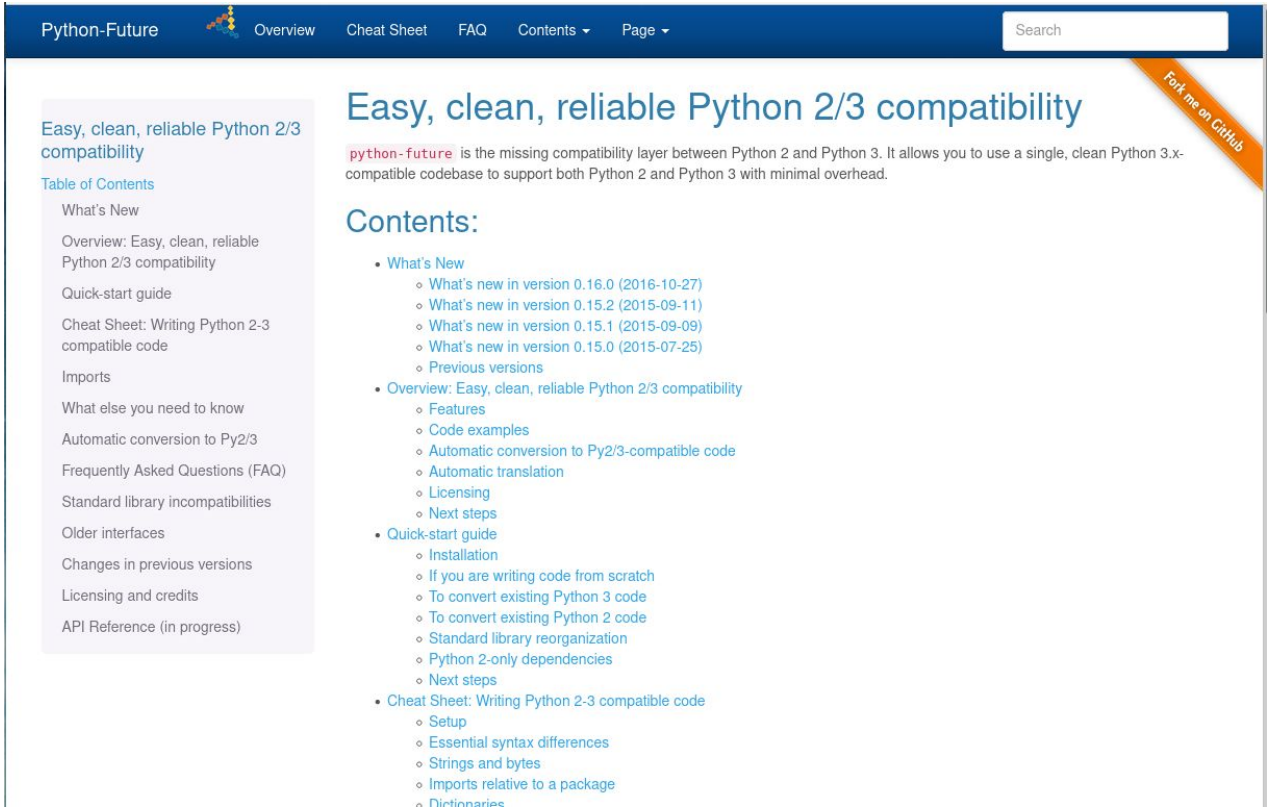
Python-2/Python-3 Compatibility

Issue 1: SCons

- SCons is used for hdds, halld_sim, and halld_recon.
- Cannot do a simple “if python 2, do this, else if python 3, do that”
- Initially tried two copies in repository, e.g., an SConstruct (py2) and a SConstruct3 (py3). Decision at build time on whether to overwrite py2 with py3.
    - Biggest downside: changes would have to go into both.
- Current solution: use the Python-Future module
    - Available as an RPM on RHEL7/CentOS7
    - Available as an APT packages on Ubuntu
    - Already installed on jlabl, farm, and ifarm machines at JLab
    - Now a requirement!

Python-Future Module

python-future is the missing compatibility layer between Python 2 and Python 3. It allows you to use a single, clean Python 3.x-compatible codebase to support both Python 2 and Python 3 with minimal overhead.



To convert existing Python 3 code

To offer backward compatibility with Python 2 from your Python 3 code, you can use the `pasteurize` script. This adds these lines at the top of each module:

```
from __future__ import absolute_import
from __future__ import division
from __future__ import print_function
from __future__ import unicode_literals

from builtins import open
from builtins import str
# etc., as needed

from future import standard_library
standard_library.install_aliases()
```

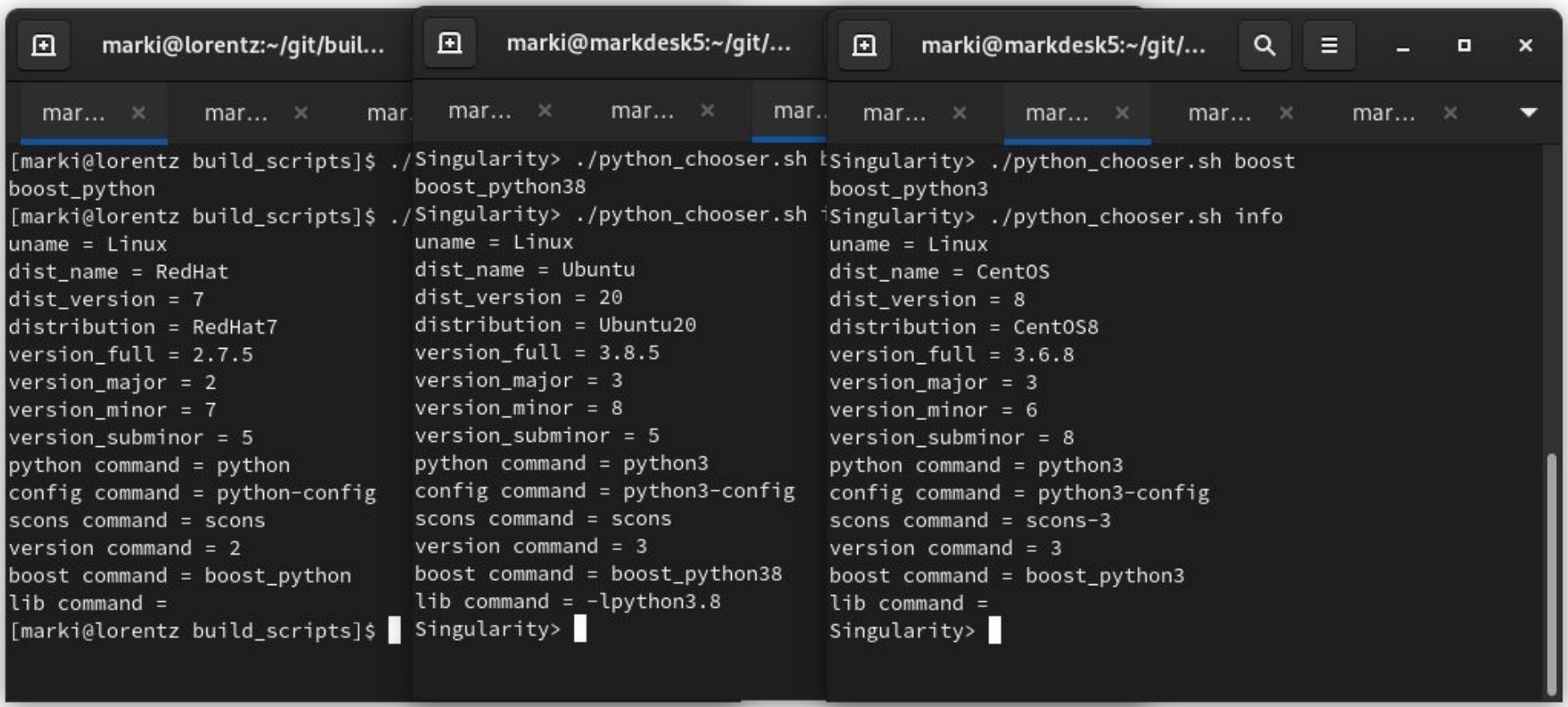
and converts several Python 3-only constructs (like keyword-only arguments) to a form compatible with both Py3 and Py2. Most remaining Python 3 code should simply work on Python 2.

See pasteurize: Py3 to Py2/3 for more details.

Python-2/Python-3 Compatibility

Issue 2: Names of Python Commands

- Depending on the distribution, “python” could be /usr/bin/python, /usr/bin/python2, or /usr/bin/python3.
- Many other distribution-dependent strings, all related to Python
- Introduce new script: python_chooser.sh (in Build Scripts). Takes one-word command line argument from the following possibilities:
  - boost / command / config / info / lib / scons / version
- Feed output to building software (make for scons) as command-line arguments, e.g., for HDGeant4
  - PYTHON_CONFIG = $(shell $(BUILD_SCRIPTS)/python_chooser.sh config)
  - make PYTHON_CONFIG=$(PYTHON_CONFIG)


marki@lorentz:~/git/buil...
[marki@lorentz build_scripts]$ ./
boost_python
[marki@lorentz build_scripts]$ ./
uname = Linux
dist_name = RedHat
dist_version = 7
distribution = RedHat7
version_full = 2.7.5
version_major = 2
version_minor = 7
version_subminor = 5
python command = python
config command = python-config
scons command = scons
version command = 2
boost command = boost_python
lib command =
[marki@lorentz build_scripts]$
marki@markdesk5:~/git/...
Singularity> ./python_chooser.sh b
boost_python38
Singularity> ./python_chooser.sh i
uname = Linux
dist_name = Ubuntu
dist_version = 20
distribution = Ubuntu20
version_full = 3.8.5
version_major = 3
version_minor = 8
version_subminor = 5
python command = python3
config command = python3-config
scons command = scons
version command = 3
boost command = boost_python38
lib command = -lpython3.8
Singularity>
marki@markdesk5:~/git/...
Singularity> ./python_chooser.sh boost
boost_python3
Singularity> ./python_chooser.sh info
uname = Linux
dist_name = CentOS
dist_version = 8
distribution = CentOS8
version_full = 3.6.8
version_major = 3
version_minor = 6
version_subminor = 8
python command = python3
config command = python3-config
scons command = scons-3
version command = 3
boost command = boost_python3
lib command =
Singularity>

Creating Singularity Containers

- New repository: github.com/jeffersonlab/hd_singularity

master | 1 branch | 0 tags | Go to file | Add file | Code

markito3 Rename Fedora 32 recipe. `0dc51ba` 12 days ago | **12** commits

recipes	Rename Fedora 32 recipe.	12 days ago
scripts	rearrange arguments; fix some logic	15 days ago
README.md	update usage message	15 days ago

README.md

hd_singularity

Files to support building and maintenance of Singularity containers for Hall D.

Contains scripts and recipes for creating Singularity containers from scratch.

The main script is scripts/create_gluex_container.sh. Its usage message is as follows:

```
Usage: create_gluex_container.sh [-h] -r <recipe-file> -p <prereqs-script> \
       [-d DIRECTORY] [-t STRING]
```

About

Files to support building and maintenance of Singularity containers for Hall D

Readme

Releases

No releases published
Create a new release

Packages

No packages published
Publish your first package

Languages

Shell 100.0%

```
Usage: create_gluex_container.sh [-h] -r <recipe-file> -p <prereqs-script> \
       [-d DIRECTORY] [-t STRING]

Note: must be run as root

Options:
  -h print this usage message
  -r Singularity recipe file
  -p script that installs gluex software
  -d output directory for containers (default: current working directory)
  -t token to be used to name containers (default = extension in "Singularity.ext")
```

master

hd_singularity / recipes / **Singularity.centos-8.2.2004**

Go to file

markito3 Initial copy from singularity directory of glu… Latest commit 967d539 on Oct 23 **History**

1 contributor

2 lines (2 sloc) | 40 Bytes

Raw Blame

```
1  Bootstrap: docker
2  From: centos:8.2.2004
```

What create_gluex_container.sh does

- Three step process
  a. Create “raw” sandbox container
  b. Install GlueX software (e.g., gfortran, git) into sandbox
  c. Create squashfs container
- Sandbox container: directory tree with usual layout of system software, useable as a container.
- Squashfs container: single binary file, only useable as a container (*.sif file)

```
[marki@markdesk5 ~]$ cat /etc/redhat-release
Fedora release 33 (Thirty Three)
[marki@markdesk5 ~]$ singularity shell /data/gluex/singularity/gluex_centos-8.2.2004.sif
Singularity> cat /etc/redhat-release
CentOS Linux release 8.2.2004 (Core)
Singularity>
```

Future architectures for running GlueX jobs

- Latest master branches will build on CentOS 7 and 8, Ubuntu 20, and Fedora 33.
- Containers give a consistent, easily distributed environment.
- Developing code in a container a bit inconvenient.
- Distributing modules to add to native OS another option
  - Avoids system upgrade.
  - Some of the same inconveniences.
  - Potential problem for users with “different” native OS.
- May want to go to containers everywhere system some day.
- Are we better off settling for RedHat-7-era software tools?

Compiler Upgrade

Site	GCC version
JLab	4.4.7, 4.9.2 available
UConn	4.4.7
Northwestern	4.4, could go to 4.8
MEPhI	4.9
CMU	4.4.7, could go to 4.9.2
Athens	4.8.4
Regina	4.4.7
Open Science Grid	4.4.7, option for 4.9.2

- New C++11 language features available in 4.8 and higher
- Some developers would like to use modern features
- Each computer used to compile code would have to upgrade
- Proposal: go to 4.9.2
- When?