Foreign packages in GNU Guix

Examples from Ruby gems, Python modules and R/CRAN

Pjotr Prins & David Thompson

FOSDEM

January 30th, 2016

UMC Utrecht/UTHSC GeneNetwork.org

GNU Guix

What is it to me?

Holy grail

- Controlled software deployment
- Control the full dependency graph
- Write once and *reproduce any time*
- Understand what is happening

Foreign packages

What are they?

Foreign packages?

- Non-native guix packages: Perl, Python, Ruby, R
- but also emacs, Firefox, etc.
- I.e., any tool that wants to install software locally

Ruby gems

- Unpacks gem in global Ruby directory
- Or in $GEM_HOME
- gem major number versioning
- gems are not isolated, not even between ruby versions
- rvm, rbenv, bundler try to solve the mess
- I got to dislike rvm and bundler with a passion

Solving the mess

- if $GEM_HOME is isolated the gem tool can do the job
- /gnu/store/p8ks1l9cl-ruby-2.2.4/bin/ruby -v

  ruby 2.2.4p230 (2015-12-16 revision 53155) [x86_64-linux]
- export GEM_HOME=~/.gem/p8ks1l9cl-ruby-2.2.4/2.2.0
- export GEM_PATH=~/.gem/p8ks1l9cl-ruby-2.2.4/2.2.0
- **gem install log4r**
- **ruby -e require 'log4r'**

It works!

At this point I am living with a

- reproducible modern ruby
- no need for rvm, rbenv or bundler
- gem works. Gems are isolated but not fully reproducible
- bundler still works
- Great for me

But. . .

I am still not satisfied

For real

- System-wide deployments
- Reproducible: when and where should not matter
- I.e., we want full control over the depency graph
- So, Ruby gems ultimately need to go into GNU Guix
- With help from David I started adding ruby gem support

Guix gem support

A short code walk

- **guix/build-system/ruby.scm** - defines what a package should be like
- **guix/build/ruby-build-system.scm** - unpacks and builds the package
- **guix/import/ruby.scm** - automatically imports gems
- **gnu/package/ruby.scm** - contains Ruby and gems

Import gem

guix import gem log4r

```
(package
  (name "ruby-log4r")
  (version "1.1.10")
  (source
    (origin
      (method url-fetch)
      (uri (rubygems-uri "log4r" version))
      (sha256
        (base32
          "0ri90q0frfmigkirqv5ihyrj59xm8pq5zcmf156cbdv4r4l2jicv"))))
  (build-system ruby-build-system)
  (synopsis
    "See also: http://logging.apache.org/log4j")
  blah, blah)
```

Install gem

- **guix package -i ruby-log4r**
- Installs /gnu/store/409cynfy62-ruby-log4r-1.1.10/lib/
- Symlinks ~/.guix-profile/lib/ruby/gems/2.2.0/gems/
- **guix package - -search-paths**
- export GEM_PATH=~/.guix-profile/lib/ruby/gems/2.2.0
- **ruby -e require 'log4r'**

Done

We have

- A fully reproducible ruby
- A fully reproducible gem
- Write once and deploy anywhere

Python modules

The GNU Guix python build-system

- **guix package -A python**
- **guix package -i python2-parsedatetime**
- Imports from pypi
- Installs anything with a setup.py
- Supports python3 and python2 with one definition:

```
(define-public python2-parsedatetime
  (package-with-python2 python-parsedatetime))
```

R modules

The GNU Guix R build-system

- Imports from CRAN
- Installs anything on CRAN and bioconductor
- R modules are not tied to R releases!

What is Guix to me?

- Dependable software deployment
- Reproducible software deployment
- Incremental software development
  1. As a system administrator (users, versions, transactions)
  2. As a software developer (user support, multiple versions)

Debian etc.

How does GNU Guix compare to other popular distributions?

- Versioning is **NOT** a problem
- Multiple targets are **NOT** a problem (+/- ssl, BLAS/Atlas)
- Foreign imports are *easy*
- Packaging is relatively *easy*
- The only dependency is the Linux kernel **API**

Docker?

- With Docker order and time of software installation matter
- The build is not reproducible
- Sending multiple **GBs** around for a simple thing?
- File systems and network policies? At what level should we care?
- You must be **MAD** to use Docker when there is GNU Guix

Conclusion

- Adding packages is *easy*
- Guix is hackable - with a bit of LISP adding features such as foreign package support is surprisingly *easy*
- GNU Guix has the right level of abstraction

See `https://github.com/pjotrp/guix-notes`

Acknowledgements

- In particular Ludovic Courtès, Ricardo Wurmus, David Thompson & Ben Woodcroft for suporting Ruby packaging and the ruby-build-system
- The GNU and GNU Guix communities (many, many)
- Prof. Rob W. Williams, UTHSC (http://genenetwork.org/)