Growing a GNU with Guix

Ludovic Courtès
ludo@gnu.org

FOSDEM
2 February 2014, Brussels
Howdy!

(guile)
Howdy!

(guile)

❄️ NixOS
Howdy!
the "GNU system", 30 years later
the "GNU system", 30 years later

- protect & enhance computing freedom
the “GNU system”, 30 years later

▶ protect & enhance *computing freedom*
▶ improve *integration* of GNU software, consistency
▶ improve *workflow* among GNU hacker & users
team leader, GNU marketing dept.
Dependable.
per-user, transactional package installation etc.

alice@foo$ guix package --install=gcc
alice@foo$ guix gc --references ‘which gcc’
/nix/store/...-glibc-2.17
/nix/store/...-gcc-4.8.0
...

demo!

bob@foo$ guix package --install=gcc-4.7.3
bob@foo$ guix gc --references ‘which gcc’
/nix/store/...-glibc-2.13
/nix/store/...-gcc-4.7.3
...
Alice@foo$ guix package --install=emacs
The following package will be installed:
  emacs-24.3 out /nix/store/...-emacs-24.3

The following files will be downloaded:
  /nix/store/...-emacs-24.3
  /nix/store/...-libxpm-3.5.10
  /nix/store/...-libxext-1.3.1
  /nix/store/...-libxaw-1.0.11
alice@foo$ guix package --install=emacs
The following package will be installed:
  emacs-24.3 out /nix/store/...-emacs-24.3

The following files will be downloaded:
  /nix/store/...-libxext-1.3.1
  /nix/store/...-libxaw-1.0.11
The following derivations will be built:
  /nix/store/...-emacs-24.3.drv
  /nix/store/...-libxpm-3.5.10.drv
transactional upgrades

$ guix package --upgrade

The following packages will be installed:

<table>
<thead>
<tr>
<th>Package</th>
<th>Type</th>
<th>Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>emacs-24.3</td>
<td>out</td>
<td>/nix/store/...-emacs-24.3</td>
</tr>
<tr>
<td>gdb-7.6</td>
<td>out</td>
<td>/nix/store/...-gdb-7.6</td>
</tr>
<tr>
<td>geiser-0.4</td>
<td>out</td>
<td>/nix/store/...-geiser-0.4</td>
</tr>
<tr>
<td>glibc-2.17</td>
<td>out</td>
<td>/nix/store/...-glibc-2.17</td>
</tr>
<tr>
<td>guile-2.0.9</td>
<td>out</td>
<td>/nix/store/...-guile-2.0.9</td>
</tr>
</tbody>
</table>

...
transactional upgrades

$ guix package --upgrade
The following packages will be installed:
  emacs-24.3 out /nix/store/...-emacs-24.3
  gdb-7.6 out /nix/store/...-gdb-7.6
  geiser-0.4 out /nix/store/...-geiser-0.4
  glibc-2.17 out /nix/store/...-glibc-2.17
  guile-2.0.9 out /nix/store/...-guile-2.0.9
...

$ emacs --version ; guile --version
GNU Emacs 24.3.1
guile (GNU Guile) 2.0.9
transactional upgrades

$ guix package --upgrade
The following packages will be installed:
  emacs-24.3    out  /nix/store/...-emacs-24.3
  gdb-7.6       out  /nix/store/...-gdb-7.6
  geiser-0.4    out  /nix/store/...-geiser-0.4
  glibc-2.17    out  /nix/store/...-glibc-2.17
  guile-2.0.9   out  /nix/store/...-guile-2.0.9
  ...

(interrupted right in the middle)

$ emacs --version ; guile --version
GNU Emacs
guile (GNU Guile)
transactional upgrades

$ guix package --upgrade
The following packages will be installed:
    emacs-24.3  out  /nix/store/...-emacs-24.3
    gdb-7.6    out  /nix/store/...-gdb-7.6
    geiser-0.4 out  /nix/store/...-geiser-0.4
    glibc-2.17 out  /nix/store/...-glibc-2.17
    guile-2.0.9 out  /nix/store/...-guile-2.0.9
...
(interrupted right in the middle)

$ emacs --version ; guile --version
GNU Emacs 23.2
guile (GNU Guile) 1.8.8
transactional upgrades

$ guix package --upgrade
The following packages will be installed:
  emacs-24.3  out  /nix/store/...-emacs-24.3
  gdb-7.6    out  /nix/store/...-gdb-7.6
  geiser-0.4 out  /nix/store/...-geiser-0.4
  glibc-2.17 out  /nix/store/...-glibc-2.17
  guile-2.0.9 out  /nix/store/...-guile-2.0.9
...
(interrupted right in the middle)

$ emacs --version ; guile --version
GNU Emacs 23.2
guile (GNU Guile) 1.8.8
$PATH

/nix/.../profiles

current

42

/nix/store

pp56i0a01si5...-user-env

bin

icecat

ssh

l9w6773m1msy...-openssh-4.6p1

bin

ssh

rpdqxnilb0cg...-icecat-3.5.4

bin

icecat

aqn3wyggq9jzk...-openssh-5.2p1

bin

ssh

guix package --upgrade=openssh
guix package --upgrade=openssh
guix package --upgrade=openssh
guix package --upgrade=openssh
$PATH
/nix/.../profiles
  current
  43

/nix/store
  rpdqxnib0cg...-icecat-3.5.4
    bin
    icecat
  aqn3wygq9jzk...-openssh-5.2p1
    bin
    ssh
  i3d9vh6d8ip1...-user-env
    bin
    ssh
    icecat

guix gc
rollback

$ emacs --version
GNU Emacs 24.2

$ guix package --upgrade=emacs
The following packages will be installed:
  emacs-24.3.1 out /nix/store/...-emacs-24.3.1
...

$ emacs --version
Segmentation Fault

$ guix package --roll-back
switching from generation 43 to 42

$ emacs --version
GNU Emacs 24.2
Hackable.
<project xmlns="http://guix.gnu.org/POM/0.0.1"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://guix.gnu.org/POM/0.0.1
  http://guix.gnu.org/xsd/guix-0.0.1.xsd">
  <modelVersion>0.0.1</modelVersion>

  <!-- The Basics -->
  <groupId>...</groupId>
  <artifactId>...</artifactId>
  <version>...</version>
  <packaging>...</packaging>
  <dependencies>...</dependencies>
  <parent>...</parent>
  <dependencyManagement>...</dependencyManagement>
  <modules>...</modules>
  <properties>...</properties>

  <!-- Build Settings -->
  <build>...</build>
  <reporting>...</reporting>

  <!-- More Project Information -->
  <name>...</name>
  <description>...</description>
{
    "name": "http-server",
    "preferGlobal": true,
    "version": "0.3.0",
    "description": "a simple zero-configuration command-line http server",
    "bin": {
        "http-server": "./bin/http-server"
    },
    "scripts": {
        "start": "node ./bin/http-server",
        "test": "vows --spec --isolate"
    },
    "main": "./lib/http-server",
    "dependencies": {
        "colors": "*",
        "flatiron": "0.1.x",
        "optimist": "0.2.x"
    },
    "license": "MIT",
    "engines": {
        "node": ">=0.6"
    }
}
LISP is over half a century old and it still has this perfect, timeless aura about it.
The truth is that Lisp is not the right language for any particular problem. Rather, Lisp encourages one to attack a new problem by implementing new languages tailored to that problem.

– Abelson & Sussman, 1987
(define hello

  (package
    (name "hello")
    (version "2.8")
    (source (origin
      (method url-fetch)
      (uri (string-append
                 "mirror://gnu/.../hello-" version
                 ".tar.gz"))
      (sha256 (base32 "0wqd...dz6"))))

  (build-system gnu-build-system)
  (synopsis "Hello, GNU world: An example GNU package")
  (description "Produce a friendly greeting.")
  (home-page "http://www.gnu.org/software/hello/")
  (license gpl3+)))
build processes
chroot, separate UIDs

Guile
(guix packages)
(guix store)

build daemon
build processes
chroot, separate UIDs

build daemon

Guile
(guix packages)
(guix store)

RPCs
(use-modules (guix packages) (guix store)
  (gnu packages base))

(define store
  (open-connection))

(package? hello)
=> #t

(define drv (package-derivation store hello))
drv
=> "/nix/store/xyz...-hello-2.8.drv"

(build-derivations (list drv))
... daemon builds/downloads package on our behalf...
=> "/nix/store/pqr...-hello-2.8"
copy fields from hello except for version and source

(package (inherit hello)
 (version "2.7")
 (source
  (origin
   (method url-fetch)
   (uri "mirror://gnu/hello/hello-2.7.tar.gz")
   (sha256
    (base32 "7dqw3..."))))
(define (static-package p)
  ;; Return a statically-linked variant of P.
  (package (inherit p)
    (arguments
     '(#:configure-flags '("--disable-shared"
                                "LDFLAGS=-static")
      ,(package-arguments p))))))
(define foo (package ...))
Workflow

(define foo (package ...))

test

guix build foo
/nix/store/...-foo-1.0

user
(define foo (package ...))

user

git push
git.sv.gnu.org

guix build foo /nix/store/...-foo-1.0
test
(define foo (package ...))

user

hydra.gnu.org

build farm

pull

git.sv.gnu.org

pull

git push

guix build foo /nix/store/...-foo-1.0

test

workflow
(define foo (package ...))

test

guix build foo

/git/store/...-foo-1.0

git push

user

get binary

hydra.gnu.org

build farm

pull

pull

get binary

git.sv.gnu.org
(define foo (package ...) )

guix build foo
/nix/store/...-foo-1.0

git push

git.sv.gnu.org
(define foo (package ...))

guix build foo /nix/store/...-foo-1.0

no single point of trust
git push

no "maintainer uploads"
git.sv.gnu.org

user

workflow
boot time!
(define my-config
  (operating-system
    (host-name "gnubox")
    (timezone "Europe/Paris")
    (locale "en_US.UTF-8")
    (users (list (user-account
                   (name "ludo")
                   (uid 1000) (gid 100)
                   (comment "Hello, this is me!"))
                   (home-directory "/home/ludo"))
    (packages (list coreutils bash grep sed
               findutils inetuutils
guile-2.0
dmd guix
procps psmisc
zile less))))
(define my-config
  (operating-system
    (host-name "gnubox")
    (timezone "Europe/Paris")
    (locale "en_US.UTF-8")
    (initrd (qemu-initrd))
    (users (list (user-account
                   (name "ludo")
                   (uid 1000) (gid 100)
                   (comment "Hello, this is me!")
                   (home-directory "/home/ludo"))))
  (packages (list coreutils bash grep sed
              findutils inetsutils
              guile-2.0
              dmd guix
              procps psmisc
              zile less))))
(define my-config
  (operating-system
    (host-name "gnubox")
    (timezone "Europe/Paris")
    (locale "en_US.UTF-8")
    (initrd (expression->initrd ...)))
  (users (list (user-account
                (name "ludo")
                (uid 1000) (gid 100)
                (comment "Hello, this is me!")
                (home-directory "/home/ludo"))))
  (packages (list coreutils bash grep sed
              findutils inetutils
              guile-2.0
              dmd guix
              procps psmisc
              zile less)))
(expression->initrd
  '(begin
      (mkdir "/proc")
      (mount "none" "/proc" "proc")

    ;; Load Linux kernel modules.
    (let ((slurp (lambda (module)
                    (call-with-input-file
                      (string-append "/modules/" module)
                      get-bytevector-all)))))
      (for-each (compose load-linux-module slurp)
                (list "md4.ko" "ecb.ko" "cifs.ko")))

    ;; Turn eth0 up.
    (let ((sock (socket AF_INET SOCK_STREAM 0)))
      (set-network-interface-flags sock "eth0" IFF_UP))

    ;; At last, the warm and friendly REPL.
    (start-repl)))

boot to Guile!
(define my-config
  (operating-system
    (host-name "gnubox")
    (timezone "Europe/Paris")
    (locale "en_US.UTF-8")
    (users (list (user-account
                     (name "ludo")
                     (uid 1000) (gid 100)
                     (comment "Hello, this is me!")
                     (home-directory "/home/ludo")))))
  (packages (list coreutils bash grep sed ...)...))
(define my-config
  (operating-system
    (host-name "gnubox")
    (timezone "Europe/Paris")
    (locale "en_US.UTF-8")
  )
  (services
    (list (mingetty-service "tty1"
      #:motd (text-file "motd" "This is tty One.")
    )
    (mingetty-service "tty2")
    (syslog-service)
    (nscd-service))
  )
  (users (list (user-account
      (name "ludo")
      (uid 1000) (gid 100)
      (comment "Hello, this is me!")
      (home-directory "/home/ludo")))
  )
  (packages (list coreutils bash grep sed ...)))
# deco status dmd
Started: (term-tty1 term-tty2 nsqd syslog)
Stopped: ()

# deco stop nscd
Service nscd has been stopped
GNU dmd in a nutshell

- born in **2003**, revived in 2013 :-)  
- dependency-based service manager
GNU dmd in a nutshell

- born in **2003**, revived in 2013 :-)
- dependency-based service manager
- `dmd` is PID 1, `deco` is a client
GNU dmd in a nutshell

- born in **2003**, revived in 2013 :-)  
- dependency-based service manager  
- `dmd` is PID 1, `deco` is a client  
- written in **Guile** Scheme  
- dynamic, extensible, etc.
Liberating.
GNU
run free run GNU
The “Corresponding Source” for a work in object code form means all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities.
The “Corresponding Source” for a work in object code form means all the source code needed to generate, install, and to modify the work, including executable code for an executable work, and scripts to control those activities. Guix users get the Corresponding Source.
build-time dependencies of GNU Hello
can we close the loop?
$ guix build bootstrap-tarballs
/nix/store/...-bootstrap-tarballs-0
$ guix build bootstrap-tarballs

/porting to new arches:

$ guix build bootstrap-tarballs \
   --target=mips64el-linux-gnuabi64
Does this binary correspond to that source?
$ guix build guile
$ guix build guile
/nix/store/ h2g4sc09h4...-guile-2.0.9

hash of **all** the dependencies
$ guix build guile
/nix/store/ h2g4sc09h4... -guile-2.0.9

$ guix gc --references /nix/store/...-guile-2.0.9
/nix/store/4jl83jgzaac...-glibc-2.17
/nix/store/iplay43cg58...-libunistring-0.9.3
/nix/store/47p47v92cj9...-libffi-3.0.9
/nix/store/drkwck2j965...-gmp-5.0.5
...

$ guix build guile
/nix/store/ h2g4sc09h4... -guile-2.0.9

$ guix gc --references /nix/store/...-guile-2.0.9
/nix/store/4jl83jgzaac...-glibc-2.17
/nix/store/iplay43cg58...-libunistring-0.9.3
/nix/store/47p47v92cj9...-libffi-3.0.9
/nix/store/drkwck2j9...-gmp-5.0.5
...

(nearly) bit-identical for everyone
controlled build environment

1. one directory **per installed package**
2. **immutable** installation directories
3. undeclared dependencies **invisible** to the build process
4. **isolated build**: chroot, container, etc.
do not trust a single binary provider
do not trust a single binary provider

Deterministic Builds: Integrity through Decentralization

– Mike Perry
Lively!
Shipping is a feature.
A really important feature.

– Joel Spolsky
timeline

- July 2012 — GHM, Düsseldorf
- Nov. 2012 — dubbed GNU
- Jan. 2013 — 0.1
- Feb. 2013 — Boot-to-Guile
- May 2013 — 0.2
- June 2013 — European Lisp Symposium
- July 2013 — 0.3, cross-compilation, debug info, etc.
- 27 Sep. 2013 — 0.4, with VM image
- Dec. 2013 — GNU dmd 0.1
- Dec. 2013 — 0.5, system config, mips64, 600 packages
status

- full-featured package manager
- self-contained distro, 600+ packages, 3 platforms
- binaries built & served at http://hydra.gnu.org
- tooling: auto-update, sync descriptions with GNU, etc.
- l10n: 4 languages!
Commits per Month
In a Nutshell, GNU Guix...

... has had 6,989 commits made by 15 contributors representing 123,238 lines of code

... is mostly written in Scheme with a very well-commented source code

... has a young, but established codebase maintained by a large development team with stable Y-O-Y commits

... took an estimated 32 years of effort (COCOMO model) starting with its first commit in April, 2012 ending with its most recent commit 5 days ago
thanks for the code, bug reports, and ideas!

- Eric Bavier, John Darrington, Eelco Dolstra & the Nix crew, Andreas Enge, Guy Grant, Nikita Karetnikov, Aljosha Papsch, Cyril Roelandt, Alex Sassmannshausen, Sree Harsha Totakura, David Thompson, Mark H. Weaver
- Lluís Batlle i Rossell, Felipe Castro, Daniel Clark, Alexandru Cojocaru, Aleix Conchillo Flaqué, Rafael Ferreira, Christian Grothoff, Jeffrin Jose, Kete, Matthew Lien, Niels Möller, Yutaka Niibe, Cyrill Schenkel, Jason Self, Alen Skondro, Matthias Wachs, Zerwas
the road to 1.0

1. simple **installer** ISO image (real soon)
the road to 1.0

1. simple **installer** ISO image (real soon)
2. **infrastructure**: get a real build farm
the road to 1.0

1. simple **installer** ISO image (real soon)
2. **infrastructure**: get a real build farm
3. packages, packages, **packages**!
install Guix atop your current distro
use it, report bugs, add packages
help with the infrastructure + admin
share your ideas!

your help needed!
 credits


➤ “Lisp is over half a century old”, http://xkcd.com/297/


➤ commit stats & project summary, http://www.ohloh.net/p/gnuguix