GNU Guix:
Unifying provisioning, deployment, and package management

Ludovic Courtès

FOSDEM, 1 February 2020
“The Linux distribution as we know it is **coming to an end**, and is being replaced by a new concept of containerized, multi-instance, multi-user applications [...]”

— Daniel Riek (2020)

https://fosdem.org/2020/schedule/event/riek_kubernetes/
Slackware | Debian | Red Hat
modules | Spack | EasyBuild | VirtualEnv
Ansible | Puppet | Propellor
Flatpak | snap | Docker | Vagrant
Are distros doomed?
“Debian and other distributions are going to be that thing you run docker on, little more.”

— Jos Poortvliet, ownCloud developer (2016)

http://lwn.net/Articles/670566/
It’s also that thing you run inside Docker!
Containers lack transparency

strawberry?

whale oil?

courtesy of Ricardo Wurmus
guix **install** gcc-toolchain openmpi hwloc

eval 'guix package --search-paths=prefix'

guix package **--roll-back**

guix **install** **--profile]**=./experiment \\
gcc-toolchain@5.5 hwloc@1
guix package --manifest=my-packages.scm

(specifications->manifest
  '("gcc-toolchain" "emacs"
   "guile" "emacs-geiser")
)
bob@laptop$ guix package --manifest=my-packages.scm

bob@laptop$ guix describe
guix cabba9e
  repository URL: https://git.sv.gnu.org/git/guix.git
  commit: cabba9e15900d20927c1f69c6c87d7d2a62040fe
bob@laptop$ guix package --manifest=my-packages.scm
bob@laptop$ guix describe
  guix cabba9e
  repository URL: https://git.sv.gnu.org/git/guix.git
  commit: cabba9e15900d20927c1f69c6c87d7d2a62040fe

alice@supercomp$ guix pull --commit=cabba9e
alice@supercomp$ guix package --manifest=my-packages.scm
travel in space and time!
guix **time-machine** --commit=cabba9e -- \install hello
guix `environment` --ad-hoc \ 
python python-numpy python-scipy \ 
-- python3
guix `environment` --ad-hoc --container \ python python-numpy python-scipy \ -- python3
$ guix pack \\
  python python-numpy
...
/gnu/store/...-pack.tar.gz
$ guix pack --relocatable \\n    python python-numpy
...
/gnu/store/...-pack.tar.gz
$ guix pack --format=docker \
    python python-numpy
...
/gnu/store/...-docker-image.tar.gz
LOL
package  environments  containers  systems
<bob> this is how Guix System works: you tell it what you want, and it puts all the pieces in place for you

<alice> yeah you just need to speak its language

<civodul> such a fine language, though :-)

(seen on #guix)
(operating-system
  (host-name "guixbox")
  (timezone "Europe/Brussels")
  (locale "fr_BE.utf8")
  (bootloader (bootloader-configuration
    (bootloader grub-efi-bootloader)
    (target "/boot/efi")))
  (file-systems (append (list (file-system
      (device (file-system-label "my-root"))
      (mount-point "/")
      (type "ext4")))))
  %base-file-systems))
(users (append (list (user-account
    (name "charlie")
    (group "users")
    (home-directory "/home/charlie")))
    %base-user-accounts))
(services (append (list (service dhcp-client-service-type)
    (service openssh-service-type))
    %base-services)))
$ guix system vm config.scm
...

$ guix system docker-image config.scm
...

$ guix system container config.scm
...

$ guix system reconfigure config.scm
...
(define (os-for-machine n)
    ;; Return an OS for machine number N.
    (operating-system
        (host-name (string-append "machine" 
                            (number->string n)))
        ...
    ))

    ;; Return a list of machines.
    (map (lambda (n)
        (machine
            (operating-system (os-for-machine n))
            (environment managed-host-environment-type)
            (configuration (machine-ssh-configuration
                                (host-name (ip-for-machine n))))))))
    (list 1 2 3 4 5))
(define (os-for-machine n)
  ;; Return an OS for machine number N.
  (operating-system
   (host-name (string-append "machine" (number->string n)))
   ...
  ))

;; Return a list of machines.
(map (lambda (n) (machine
  (operating-system (os-for-machine n))
  (environment managed-host-environment-type)
  (configuration (machine-ssh-configuration
    (host-name (ip-for-machine n))))
  )
  (list 1 2 3 4 5))

New! guix deploy machines.scm
(define (os-for-machine n)
  ;; Return an OS for machine number N.
  (operating-system
    (host-name (string-append "machine"
                      (number->string n)))
    ...
  ))

;; Return a list of machines.
(map (lambda (n)
  (machine
    (operating-system (os-for-machine n))
    (environment managed-host-environment-type)
    (configuration (machine-ssh-configuration
                    (host-name (ip-for-machine n))))))
  (list 1 2 3 4 5))
(define (os-for-machine n)
  ;; Return an OS for machine number N.
  (operating-system
   (host-name (string-append "machine" (number->string n)))
   ...
  )
)

;; Return a list of machines.
(map (lambda (n)
  (machine
   (operating-system (os-for-machine n))
   (environment digital-ocean-environment-type)
   (configuration (digital-ocean-configuration
                   (region "nyc3")
                   ...
                  ))
  )
   (list 1 2 3 4 5))
It’s all about source code.
(define audacity
  (package
    (name "audacity")
    (home-page "https://github.com/audacity/audacity")
    (source (origin
              (method git-fetch)
              (uri (git-reference
                    (url home-page)
                    (commit "2f30ff07a")
                    (recursive? #t)))
              (sha256
                (base32
                  "106rf402cvfdhc2yf..."))))
...))
(define audacity
  (package
    (name "audacity")
    (home-page "https://github.com/audacity/audacity")
    (source (origin
      (method git-fetch)
      (uri (git-reference
          (url home-page)
          (commit "2f30ff07a")
          (recursive? #t)))
      (sha256
        (base32
          "106rf402cvfdhc2yf..."))))
...))
emacs = f(gtk+, gcc, make, coreutils)

where $f = .configure \&\& make \&\& make install$
emacs = f(gtk+, gcc, make, coreutils)

gtk+ = g(glib, gcc, make, coreutils)
emacs = f(gtk+, gcc, make, coreutils)

gtk+ = g(glib, gcc, make, coreutils)

gcc = h(make, coreutils, gcc₀)

...
$ guix build hello

isolated build: chroot, separate name spaces, etc.
$ guix build hello

/gnu/store/ h2g4sf72... -hello-2.10

hash of all the dependencies
$ guix build hello
/gnu/store/ h2g4sf72... -hello-2.10

$ guix gc --references /gnu/store/...-hello-2.10
/gnu/store/...-glibc-2.29
/gnu/store/...-gcc-7.4.0-lib
/gnu/store/...-hello-2.10
$ guix build hello
/gnu/store/ h2g4sf72... -hello-2.10

$ guix gc --references /gnu/store/...-hello-2.10
/gnu/store/...-glibc-2.29
/gnu/store/...-gcc-7.4.0-lib
/gnu/store/...-hello-2.10

(nearly) bit-identical for everyone
$ guix challenge --substitute-urls="https://ci.guix.gnu.org https://example.org"

/gnu/store/. . . -openssl-1.0.2d contents differ:
  local hash: 0725122...
  http://ci.guix.gnu.org/. . . -openssl-1.0.2d: 0725122...
  http://example.org/. . . -openssl-1.0.2d: 1zy4fma...

/gnu/store/. . . -git-2.5.0 contents differ:
  local hash: 00p3bmr...
  http://ci.guix.gnu.org/. . . -git-2.5.0: 069nb85...
  http://example.org/. . . -git-2.5.0: 0mdqa9w...

/gnu/store/. . . -pius-2.1.1 contents differ:
  local hash: 0k4v3m9...
  http://ci.guix.gnu.org/. . . -pius-2.1.1: 0k4v3m9...
  http://example.org/. . . -pius-2.1.1: 1cy25x1...
Reflections on Trusting Trust

To what extent should one trust a statement that a program is free of Trojan horses? Perhaps it is more important to trust the people who wrote the software.

KEN THOMPSON
250 MiB of binary blobs
250 MiB → 130 MiB of binary blobs

Rust entirely built from source!

\[ f(config\_scm) = \]
$f(\text{config.scm}) = f^{-1}$?
$ guix system describe
file name: /var/guix/profiles/system-126-link
canonical file name: /gnu/store/...-system
label: GNU with Linux-Libre 5.4.15
bootloader: grub-efi
root device: label: "root"
channels:
guix:
    repository URL: https://git.savannah.gnu.org/...
    commit: 93f4511eb0c9b33f5083c2a04f4148e0a494059c
configuration file: /gnu/store/...-configuration.scm
Wrap-up.
Not included in this talk :-) 

- **embedded usage**
  - Go to K.3.201, Sun. 11:00AM!

- **Guile & programming language technology**
  - Go to AW1.125, Sun. 11:30AM!

- **Guix-HPC: high-performance computing**
  - Go to UB.132, Sun. 12:30PM!
Join us now, share the parens!

▶ install it!
▶ use it!
▶ hack it!
▶ join for Outreachy or GSoC!
Reproducible deployment is the logical next step.