Declaratively Yours
Composing system abstractions with GNU Guix

Ludovic Courtès
FOSDEM, 7 February 2021
Desired state and ‘idempotency’

Most Ansible modules check whether the desired final state has already been achieved, and exit without performing any actions if that state has been achieved, so that repeating the task does not change the final state. Modules that behave this way are often called ‘idempotent.’ Whether you run a playbook once, or multiple times, the outcome should be the same. However, not all playbooks and not all modules behave this way. If you are unsure, test your playbooks in a sandbox environment before running them multiple times in production.
```
{ fetchurl, stdenv }:

stdenv . mkDerivation {
  name = "hello-2.3";
  src = fetchurl {
    url = mirror://gnu/hello/hello-2.3.tar.bz2;
    sha256 = "0c7vijq8y68...";
  };

  meta = {
    description = "Produce a friendly greeting";
    homepage = http://www.gnu.org/software/hello/;
    license = "GPLv3+";
  };
}
```
Bill of rights:

1. What You Declare is What You Get
2. Declarations Bring Structure
The package structure.
Where are GCC, libc, etc.?
What about the compiler’s compiler, etc.?
120 nodes

(too big)

guix graph --type=bag coreutils
(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..."))))
  ...
)

guix lint -c archival audacity
$ guix **install** gcc-toolchain openmpi hwloc
...
**hint**: Consider setting the necessary environment variables by running:

```
GUIX_PROFILE="${HOME}/.guix-profile"
. "$GUIX_PROFILE/etc/profile"
```

Alternatively, see ‘guix package --search-paths’.
(define gcc
  (package
    ;; ...
    (native-search-paths
      (list (search-path-specification
              (variable "C_INCLUDE_PATH")
              (files '("include")))
      (search-path-specification
       (variable "CPLUS_INCLUDE_PATH")
       (files '("include/c++" "include")))
      (search-path-specification
       (variable "LIBRARY_PATH")
       (files '("lib" "lib64")))))
guix pack hwloc \
   --with-source=.hwloc-2.1rc1.tar.gz

guix install emacs-next \
   --with-branch=emacs-next=master

guix build guix \
   --with-latest=guile-json
(define (make-glibc-utf8-locales locales)
  (package
    (name "glibc-utf8-locales")
    ...)))

(define glibc-utf8-locales
  (make-glibc-utf8-locales (list "ca_ES.utf8" ...)))
(define emacs
  (package
    (name "emacs")
    ...))

(define emacs-no-x
  (package/inherit emacs
    (name "emacs-no-x")
    ...))

(define emacs-xwidgets
  (package/inherit emacs
    (name "emacs-xwidgets")
    ...))
What if we could declare package parameters?
(define emacs
  (package
    (name "emacs")
    ...))

(define emacs-no-x
  (package/inherit emacs
    (name "emacs-no-x")
    ...))

(define emacs-xwidgets
  (package/inherit emacs
    (name "emacs-xwidgets")
    ...))

What if we could declare package parameters?
WIP!
Package collections.
guix package --manifest=my-packages.scm

(specifications->manifest
  '("emacs" "emacs-geiser"
   "guile" "guile-picture-language"))
guix `install` emacs

`guix install` guile guile-picture-language

`guix install` emacs-geiser
guix install emacs

guix install guile guile-picture-language

guix package --export-manifest

guix install emacs-geiser
Systems.
(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)))
(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 docker-image config.scm
guix system container config.scm
guix system reconfigure config.scm
(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)))
https://notabug.org/civodul/guix-explorer
Wrap-up.