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

pip | Cabal | Cargo | CONDA | Gradle

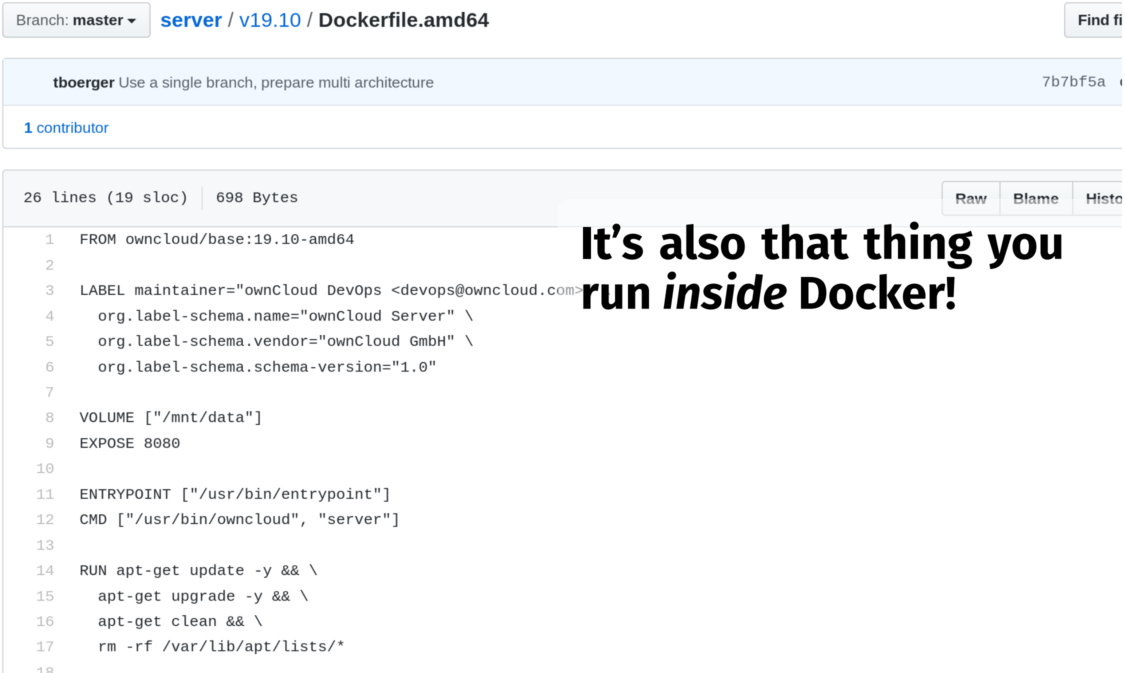
Flatpak | snap | Docker | Vagrant

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/



Branch: master server / v19.10 / Dockerfile.amd64

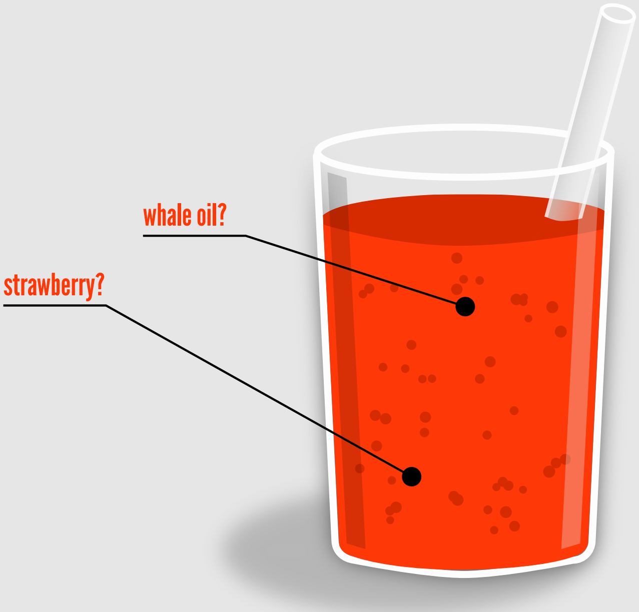
tboerger Use a single branch, prepare multi architecture 7b7bf5a

1 contributor

26 lines (19 sloc) | 698 Bytes

```
FROM owncloud/base:19.10-amd64

LABEL maintainer="ownCloud DevOps <devops@owncloud.com>
  org.label-schema.name="ownCloud Server" \
  org.label-schema.vendor="ownCloud GmbH" \
  org.label-schema.schema-version="1.0"

VOLUME ["/mnt/data"]
EXPOSE 8080

ENTRYPOINT ["/usr/bin/entrypoint"]
CMD ["/usr/bin/owncloud", "server"]

RUN apt-get update -y && \
  apt-get upgrade -y && \
  apt-get clean && \
  rm -rf /var/lib/apt/lists/*
```

It’s also that thing you run *inside* Docker!

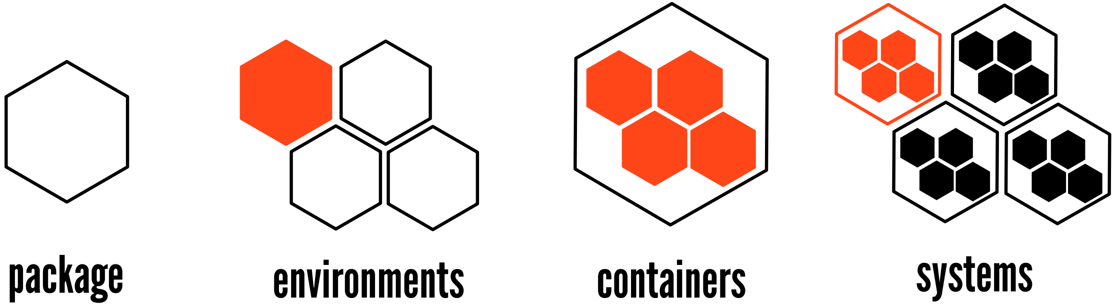


Containers

lack transparency

courtesy of Ricardo Wurmus

Guix


package
environments
containers
systems

```
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
```

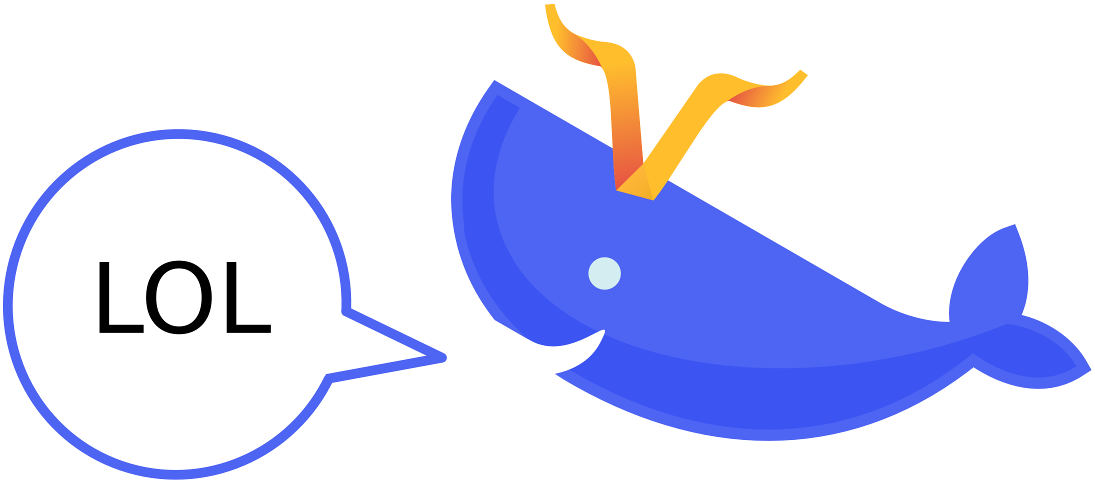
```
guix environment --ad-hoc                        \
      python python-numpy python-scipy \
      -- python3
```

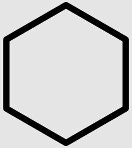
```
guix environment --ad-hoc --container \
      python python-numpy python-scipy \
      -- python3
```

```
$ guix pack \
      python python-numpy
…
/gnu/store/…-pack.tar.gz
```

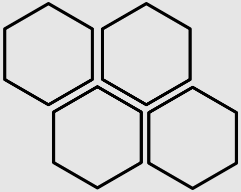
```
$ guix pack --relocatable \
      python python-numpy
…
/gnu/store/…-pack.tar.gz
```

```
$ guix pack --format=docker \
      python python-numpy
…
/gnu/store/…-docker-image.tar.gz
```

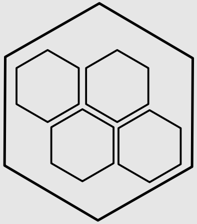

LOL



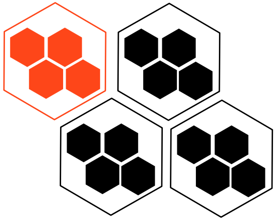
package



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)

```scheme
(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))
```

New!

```
(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))
```

```
guix deploy machines.scm
```

New!

```scheme
(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))
```

```scheme
(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…"))))
    …))
```

Software Heritage

https://www.softwareheritage.org/2019/04/18/software-heritage-and-gnu-guix-join-forces-to-enable-long-term-reproducibility/

Reproducible
Builds

$$\texttt{emacs} = f(\texttt{gtk+}, \texttt{gcc}, \texttt{make}, \texttt{coreutils})$$

where f = `./configure && make && make install`

$$\texttt{emacs} = f(\texttt{gtk+}, \texttt{gcc}, \texttt{make}, \texttt{coreutils})$$

$$\texttt{gtk+} = g(\texttt{glib}, \texttt{gcc}, \texttt{make}, \texttt{coreutils})$$

$$\texttt{emacs} = f(\texttt{gtk+}, \texttt{gcc}, \texttt{make}, \texttt{coreutils})$$

$$\texttt{gtk+} = g(\texttt{glib}, \texttt{gcc}, \texttt{make}, \texttt{coreutils})$$

$$\texttt{gcc} = h(\texttt{make}, \texttt{coreutils}, \texttt{gcc}_0)$$

$$\ldots$$

```
$ 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

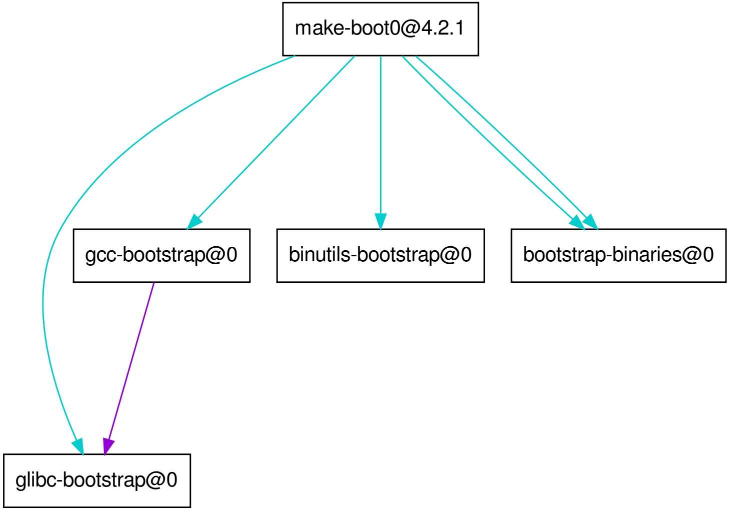
```
$ guix challenge --substitute-urls="https://ci.guix.gnu.org https://example.org
/gnu/store/. . . -openssl-1.0.2d contents differ:
  local hash: 0725l22. . .
  http://ci.guix.gnu.org/. . . -openssl-1.0.2d: 0725l22. . .
  http://example.org/. . . -openssl-1.0.2d: 1zy4fma. . .
/gnu/store/. . . -git-2.5.0 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. . .
```

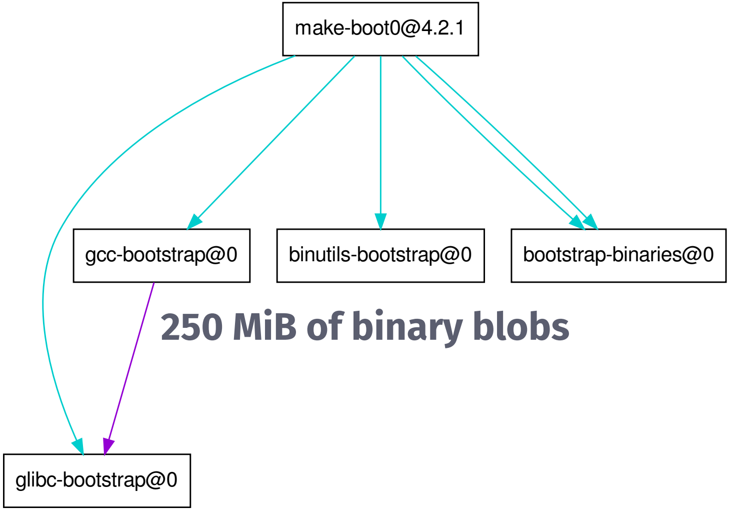
Reflections on Trusting Trust

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.

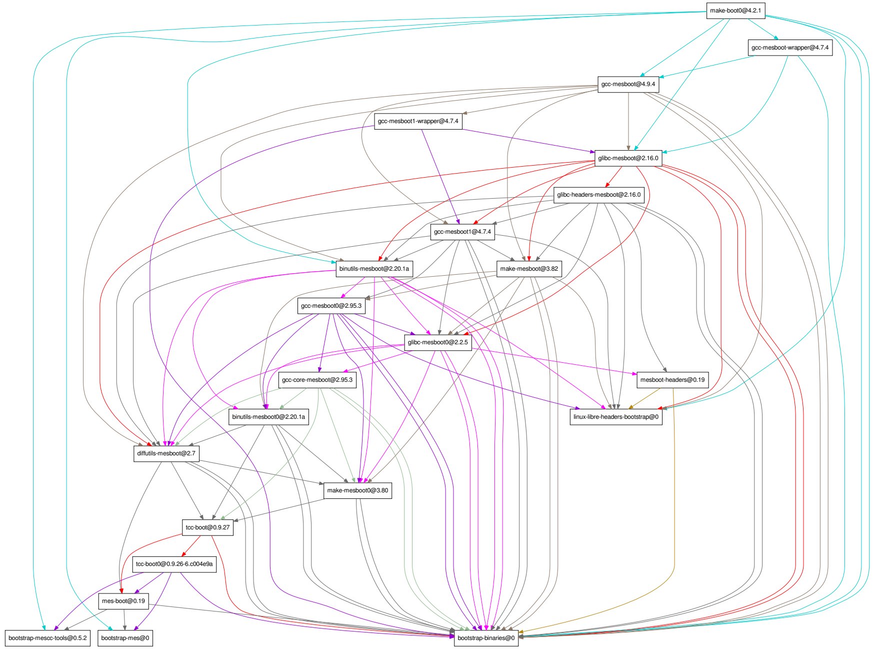
KEN THOMPSON

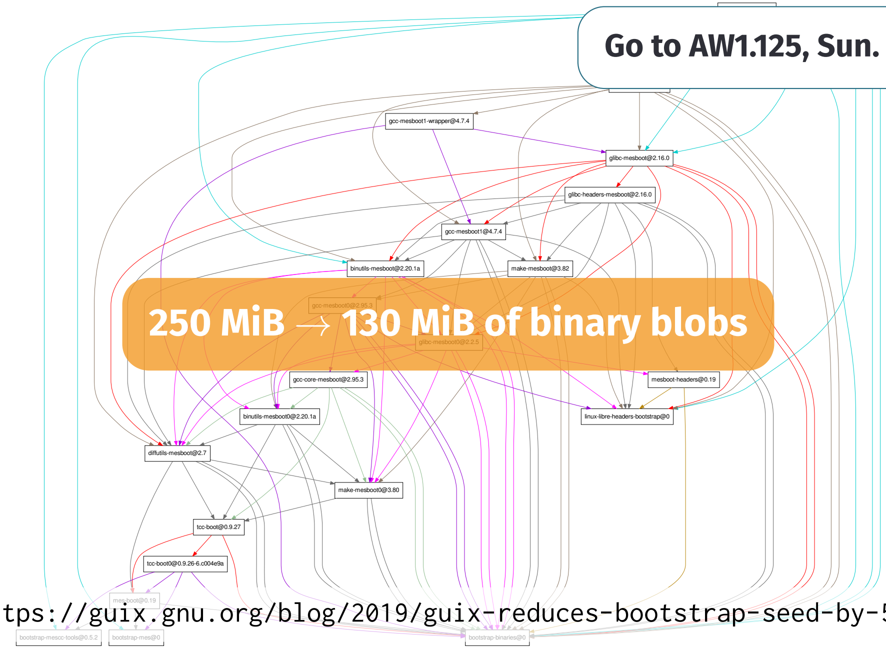
Bootstrappable Builds


make-boot0@4.2.1
gcc-bootstrap@0
binutils-bootstrap@0
bootstrap-binaries@0
glibc-bootstrap@0



250 MiB of binary blobs


make-boot0@4.2.1
gcc-mesboot-wrapper@4.7.4
gcc-mesboot@4.9.4
gcc-mesboot1-wrapper@4.7.4
glibc-mesboot@2.16.0
glibc-headers-mesboot@2.16.0
gcc-mesboot1@4.7.4
binutils-mesboot@2.20.1a
make-mesboot@3.82
gcc-mesboot0@2.95.3
glibc-mesboot0@2.2.5
gcc-core-mesboot@2.95.3
mesboot-headers@0.19
binutils-mesboot0@2.20.1a
linux-libre-headers-bootstrap@0
diffutils-mesboot@2.7
make-mesboot0@3.80
tcc-boot@0.9.27
tcc-boot0@0.9.26-6.c004e9a
mes-boot@0.19
bootstrap-mescc-tools@0.5.2
bootstrap-mes@0
bootstrap-binaries@0

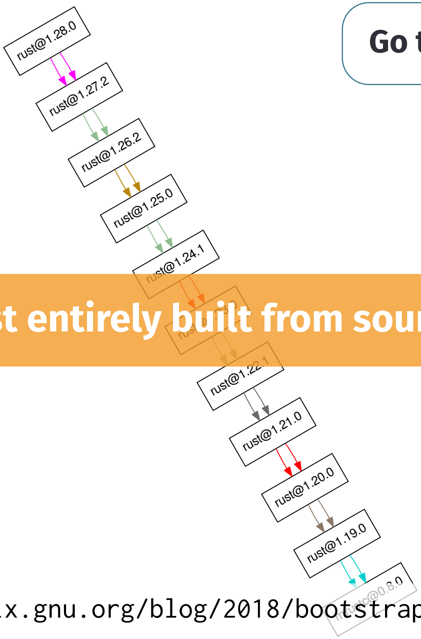


Go to AW1.125, Sun. 11:50AM

250 MiB → 130 MiB of binary blobs

https://guix.gnu.org/blog/2019/guix-reduces-bootstrap-seed-by-50/

Go to K.3.401, Sun. 10:00AM



Rust entirely built from source!

https://guix.gnu.org/blog/2018/bootstrapping-rust/

$f(\texttt{config.scm}) =$ ⚙

$$f(\texttt{config.scm}) = $$

$$f^{-1}\ ?$$

```
$ 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
```

New!

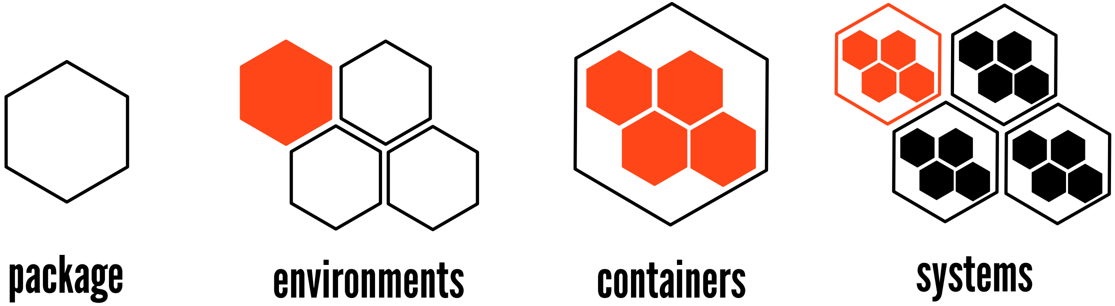
Wrap-up.

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!


package
environments
containers
systems

git → Guix

Reproducible deployment
is the logical next step.

Guix

ludo@gnu.org

https://guix.gnu.org/

Copyright © 2010, 2012–2020 Ludovic Courtès `ludo@gnu.org`.

GNU Guix logo, CC-BY-SA 4.0, `https://gnu.org/s/guix/graphics`
Reproducible Builds logo under CC-BY 3.0, `https://uracreative.github.io/reproducible-builds-styleguide/visuals/`.
Bootstrappable Builds logo by Ricardo Wurmus, `https://bootstrappable.org`.
Docker whale image by Ricardo Wurmus.
Smoothie image © 2019 Ricardo Wurmus, CC-BY-SA 4.0.
Hand-drawn arrows by Freepik from flaticon.com.

Picture of the sun under CC-BY-SA 3.0, `https://commons.wikimedia.org/wiki/File:%22Sun%22.JPG`.
Cloud picture 1 under CC-BY-SA 2.0, `https://commons.wikimedia.org/wiki/File:Cloud_(5018750171).jpg`.
Cloud picture 2 under CC-BY-SA 3.0, `https://commons.wikimedia.org/wiki/File:Cumulunimbus_IMG_5537.JPG`.
Cloud picture 3 under CC-BY-SA 4.0, `https://commons.wikimedia.org/wiki/File:2018-05-18_18_27_24_Low_stratiform_clouds_(base_near_3,000_feet_AGL)_with_wavy,_bumpy_base_viewed_from_Mercer_County_Route_622_(North_Olden_Avenue)_in_Ewing_Township,_Mercer_County,_New_Jersey.jpg`.
Thunder picture under CC-BY-SA 4.0, `https://commons.wikimedia.org/wiki/File:004_2018_05_14_Extremes_Wetter.jpg`.

Copyright of other images included in this document is held by their respective owners.

This work is licensed under the **Creative Commons Attribution-Share Alike 3.0** License. To view a copy of this license, visit `http://creativecommons.org/licenses/by-sa/3.0/` or send a letter to Creative Commons, 171 Second Street, Suite 300, San Francisco, California, 94105, USA.

At your option, you may instead copy, distribute and/or modify this document under the terms of the **GNU Free Documentation License, Version 1.3 or any later version** published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is available at `http://www.gnu.org/licenses/gfdl.html`.

The source of this document is available from `http://git.sv.gnu.org/cgit/guix/maintenance.git`.