

Packaging and deploying with GNU Guix and GuixSD

Andreas Enge

LFANT project-team
INRIA Bordeaux-Sud-Ouest
`andreas.enge@inria.fr`
`http://www.math.u-bordeaux.fr/~aenge`

Sage Days 77, Cernay, 5 April 2016

Upgrades are difficult

[4.4.3. Make sure you have sufficient space for the upgrade](#)

[4.4.4. Minimal system upgrade](#)

[4.4.5. Upgrading the kernel and udev](#)

[4.4.6. Upgrading the system](#)

[4.5. Possible issues during upgrade](#)

[4.5.1. cryptoloop support not included in the squeeze Linux kernel](#)

[4.5.2. Expected removals](#)

[4.5.3. Errors running aptitude or apt-get](#)

[4.5.4. Conflicts or Pre-Depends loops](#)

[4.5.5. File conflicts](#)

[4.5.6. Configuration changes](#)

[4.5.7. Change of session to console](#)

[4.5.8. Special care for specific packages](#)

[4.6. Upgrading your kernel and related packages](#)

[4.6.1. Installing the kernel metapackage](#)

[4.6.2. Device enumeration reordering](#)

[4.6.3. Boot timing issues \(waiting for root device\)](#)

[4.7. Preparing for the next release](#)

[5. From Squeeze to Wheezy](#)

<https://www.debian.org/releases/squeeze/amd64/release-notes/ch-upgrading.en.html>

Upgrades are dangerous

Distribution Upgrade of all the files:



WARNING

Following the upgrade instructions found in the [release notes](#) is the best way to ensure that your system upgrades from one major Debian release to another (e.g. from lenny to squeeze) without breakage!

These instructions will tell you to do a `dist-upgrade` (instead of `upgrade`) in the case of `apt-get` or `full-upgrade` (instead of `safe-upgrade` in the case of `aptitude`) at least once. So you would have to type something like

```
# aptitude full-upgrade
```

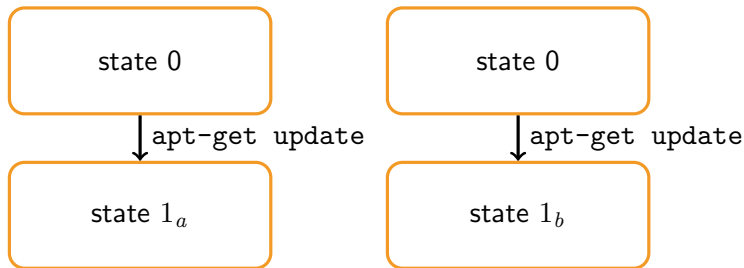
<https://wiki.debian.org/DebianPackageManagement>

Stateful system management is unpredictable

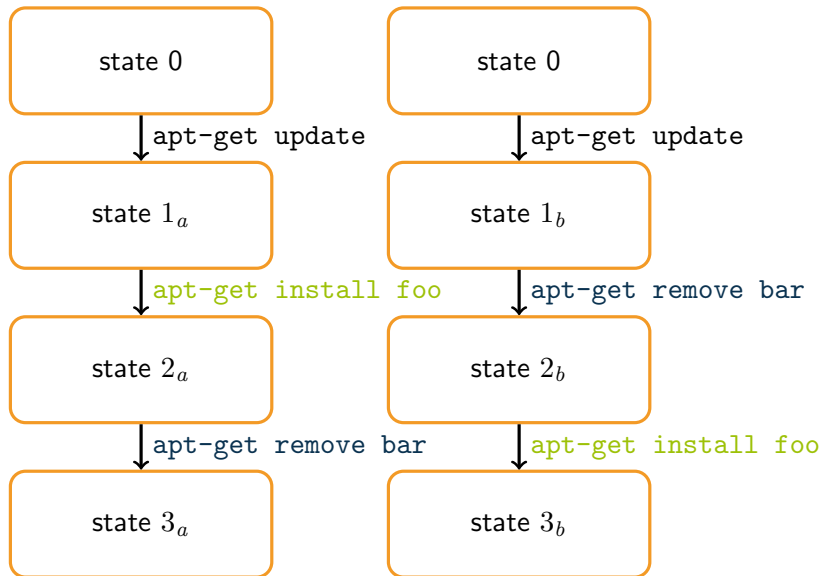
state 0

state 0

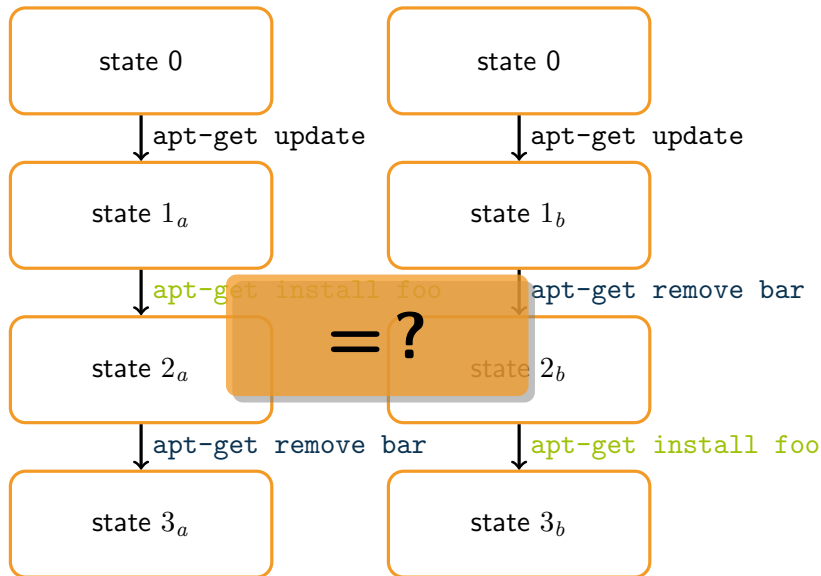
Stateful system management is unpredictable



Stateful system management is unpredictable



Stateful system management is unpredictable



Too many cooks spoil the broth

Application-level package managers [\[edit \]](#)

- [Anaconda](#) - a package manager for [Python](#)
- [Assembly](#) - a partially [compiled](#) code library for use in [Common Language Infrastructure](#) (CLI) deployment, versioning and security.
- [Bower](#) - a package manager for the web.
- [Cabal](#) - a programming library and package manager for [Haskell](#)
- [CocoaPods](#) - Dependency Manager for [Objective-C](#) and [RubyMotion](#) projects
- [Composer](#) - Dependency Manager for [PHP](#)
- [CPAN](#) - a programming library and package manager for [Perl](#)
- [CRAN](#) - a programming library and package manager for [R](#)
- [CTAN](#) - a package manager for [TeX](#)
- [EasyInstall](#) - a package manager for [Python](#) and the [PyPI](#) programming library which is part of the [Setuptools](#) packaging system
- [Gradle](#) - a build system and package manager for [Groovy](#) and other JVM languages
- [Ivy](#) - a package manager for [Java](#), integrated into the [Ant](#) build tool, also used by [sbt](#)
- [LuaRocks](#) - a programming library and package manager for [Lua](#)
- [Maven](#) - a package manager and build tool for [Java](#)
- [npm](#) - a programming library and package manager for [Node.js](#)
- [NuGet](#) - a package manager for the [.NET Framework](#) and C++.
- [PAR::Repository](#) and [Perl package manager](#) - binary package managers for [Perl](#)
- [PEAR](#) - a programming library for [PHP](#)
- [pip](#) - a package manager for [Python](#) and the [PyPI](#) programming library.
- [Quicklisp](#) - a package manager and repository for [Common Lisp](#)
- [RubyGems](#) - a package manager and repository for [Ruby](#)
- [sbt](#) - a build tool for [Scala](#), uses [Ivy](#) for dependency management
- [leinigen](#) - a project automation tool for [Clojure](#)

https://en.wikipedia.org/wiki/List_of_software_package_management_systems

Over 30% of Official Images in Docker Hub Contain High Priority Security Vulnerabilities

[Docker Hub](#) is a central repository for Docker developers to pull and push container images. We performed a detailed study on Docker Hub images to understand how vulnerable they are to security threats. Surprisingly, we found that more than 30% of images in [official repositories](#) are highly susceptible to a variety of security attacks (e.g., Shellshock, Heartbleed, Poodle, etc.). For general images – images pushed by docker users, but not explicitly verified by any authority – this number jumps up to ~40% with a sampling error bound of 3%.





Packaging and deploying with GNU Guix and GuixSD

- 1 GNU Guix: Principles and package management
- 2 GNU GuixSD: Declarative system deployments
- 3 State of the nation

`pari-gp` = $f_1(\text{pari-2.7.5.tar.gz}, \text{readline}, \text{gcc-5}, \text{make})$

Functional package management

```
pari-gp  =   $f_1$ (pari-2.7.5.tar.gz, readline, gcc-5, make)
readline =   $f_2$ (readline-6.3.tar.gz, ncurses, gcc-5, make)
ncurses  =   $f_3$ (ncurses-6.0.tar.gz, gcc-5, make)
```

Functional package management

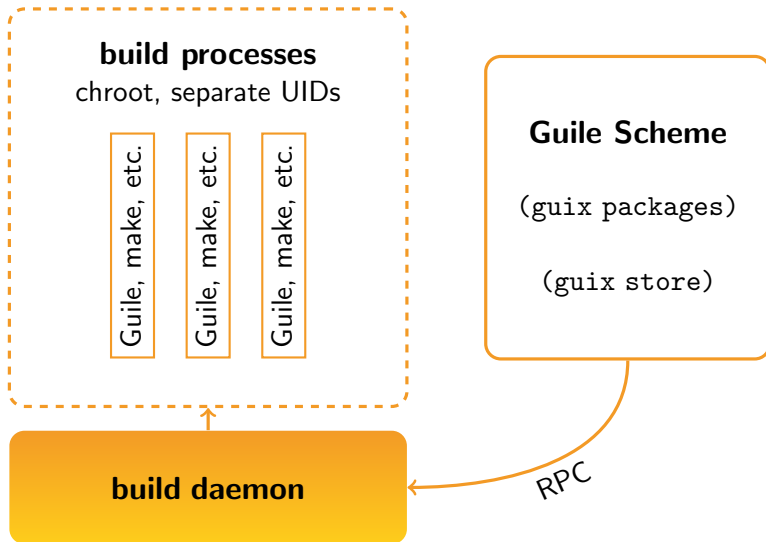
```
pari-gp  =  f1(pari-2.7.5.tar.gz, readline, gcc-5, make)
readline =  f2(readline-6.3.tar.gz, ncurses, gcc-5, make)
ncurses  =  f3(ncurses-6.0.tar.gz, gcc-5, make)
gcc-5    =  f4(gcc-5.3.0.tar.gz, gcc-4.9, make)
```

Functional package management

```
pari-gp  = f1(pari-2.7.5.tar.gz, readline, gcc-5, make)
readline = f2(readline-6.3.tar.gz, ncurses, gcc-5, make)
ncurses  = f3(ncurses-6.0.tar.gz, gcc-5, make)
gcc-5    = f4(gcc-5.3.0.tar.gz, gcc-4.9, make)
```

Full **directed acyclic graph** of dependencies is captured,
with a few **bootstrap binaries** at the root.

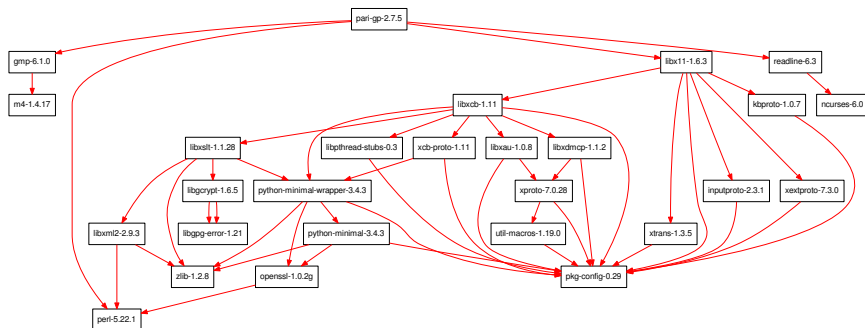
The store



Package recipes

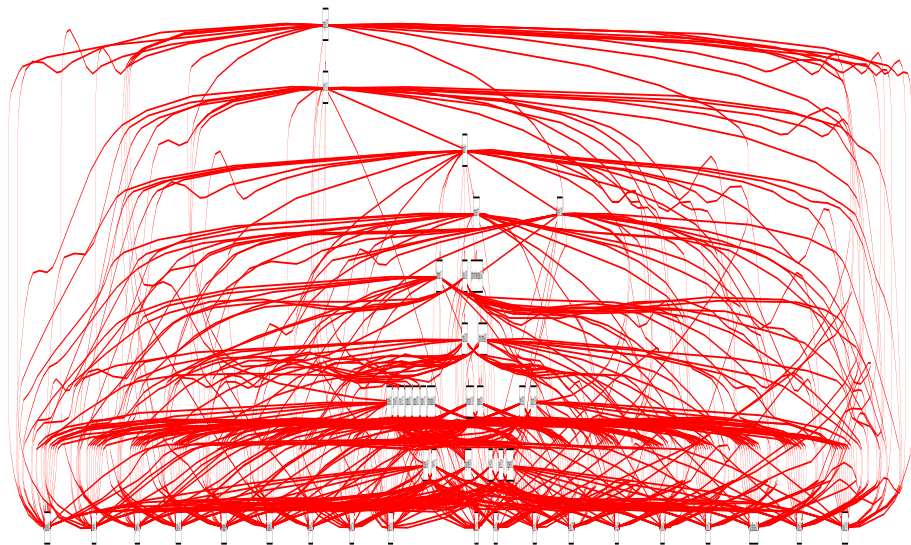
```
(define-public pari-gp
  (package
    (name "pari-gp")
    (version "2.7.5")
    (source (origin
      (method url-fetch)
      (uri "http://pari.math.u-bordeaux.fr/pub/pari/unix/pari-2.7.5.tar.gz")
      (sha256 (base32 "0c8l83a0gj73r9hndsrzkyvwxvnnm4pxkkzbg6jm95m80nzwh11"))))
    (build-system gnu-build-system)
    (inputs `(("gmp" ,gmp)
              ("libx11" ,libx11)
              ("perl" ,perl)
              ("readline" ,readline)))
    (arguments
      '(:make-flags '("gp")
        #:test-target "dobench"
        #:phases (modify-phases %standard-phases
          (replace 'configure
            (lambda* (#:key outputs #:allow-other-keys)
              (system* "./Configure"
                (string-append "--prefix=" (assoc-ref outputs "out"))))))))
    (license license:gpl2+)
    (home-page "http://pari.math.u-bordeaux.fr/")))
```

guix graph -type=packages pari-gp



And gcc, make and so on? Implicit inputs!

```
guix graph -type=bag-emerged pari-gp
```



Demo: Packages and profiles

- `guix build pari-gp`

`/gnu/store/qbb9f5s6xzbzac9f260g16b68j7w1wqb-pari-gp-2.7.5`

Hash of all inputs!

Demo: Packages and profiles

- `guix build pari-gp`
`/gnu/store/qbb9f5s6xzbzac9f260g16b68j7w1wqb-pari-gp-2.7.5`
Hash of all inputs!
- `guix gc --references /gnu/store/-...-pari-gp-2.7.5`

Demo: Packages and profiles

- `guix build pari-gp`
`/gnu/store/qbb9f5s6xzbzac9f260g16b68j7w1wqb-pari-gp-2.7.5`
Hash of all inputs!
- `guix gc --references /gnu/store/-...-pari-gp-2.7.5`
- `guix package -i pari-gp`
Look at `$HOME/.guix-profile`; run `gp`.

Demo: Packages and profiles

- `guix build pari-gp`
`/gnu/store/qbb9f5s6xzbzac9f260g16b68j7w1wqb-pari-gp-2.7.5`
Hash of all inputs!
- `guix gc --references /gnu/store/-...-pari-gp-2.7.5`
- `guix package -i pari-gp`
Look at `$HOME/.guix-profile`; run `gp`.
- `./pre-inst-env guix edit pari-gp`
`./pre-inst-env guix build pari-gp`
`guix gc --references ...`
`./pre-inst-env guix package -u pari-gp`

Demo: Packages and profiles

- `guix build pari-gp`
`/gnu/store/qbb9f5s6xzbzac9f260g16b68j7w1wqb-pari-gp-2.7.5`
Hash of all inputs!
- `guix gc --references /gnu/store/-...-pari-gp-2.7.5`
- `guix package -i pari-gp`
Look at `$HOME/.guix-profile`; run `gp`.
- `./pre-inst-env guix edit pari-gp`
`./pre-inst-env guix build pari-gp`
`guix gc --references ...`
`./pre-inst-env guix package -u pari-gp`
- `guix package -l`

Demo: Packages and profiles

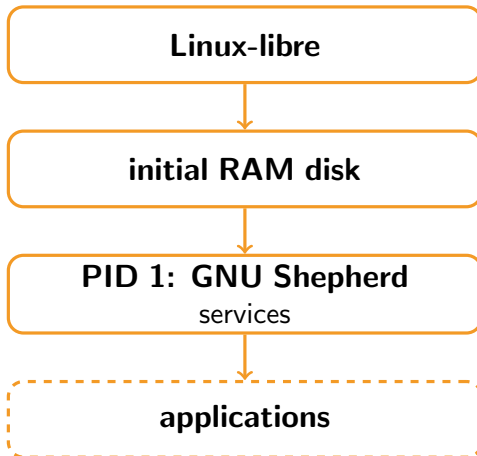
- `guix build pari-gp`
`/gnu/store/qbb9f5s6xzbgac9f260g16b68j7w1wqb-pari-gp-2.7.5`
Hash of all inputs!
- `guix gc --references /gnu/store/-...-pari-gp-2.7.5`
- `guix package -i pari-gp`
Look at `$HOME/.guix-profile`; run `gp`.
- `./pre-inst-env guix edit pari-gp`
`./pre-inst-env guix build pari-gp`
`guix gc --references ...`
`./pre-inst-env guix package -u pari-gp`
- `guix package -l`
- `guix package --roll-back`

- Hackable
 - ▶ based on Guile Scheme
- Dependable
 - ▶ immutable packages in store
 - ▶ transactional upgrades and downgrades
- Liberating
 - ▶ user centric: every user has an individual profile
 - ▶ transparent (and reproducible?) binary and source deployments
 - ▶ easy roll-back
 - ▶ different versions can co-exist
 - ★ in one user profile (?)
 - ★ for different users
 - ★ for one user in different profiles

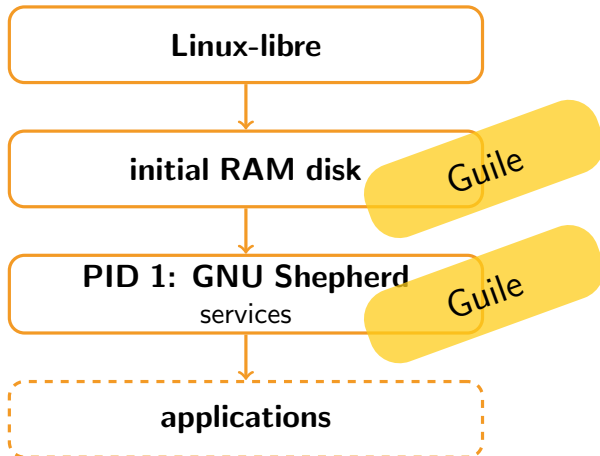
Packaging and deploying with GNU Guix and GuixSD

- 1 GNU Guix: Principles and package management
- 2 GNU GuixSD: Declarative system deployments
- 3 State of the nation

System layers



System layers



Demo: Always change a running system!

Have a look at

`/run`

`/run/current-system`

`/run/booted-system`

`/run/current-system/`

Demo: Do as you please!

- Look at `config-desktop.scm`.

Demo: Do as you please!

- Look at `config-desktop.scm`.
- Change!
 - ▶ change locales
 - ▶ add pari-gp package to global list
 - ▶ auto-login the user enge

Demo: Do as you please!

- Look at `config-desktop.scm`.
- Change!
 - ▶ change locales
 - ▶ add `pari-gp` package to global list
 - ▶ auto-login the user `enge`
- `guix system vm config-pari.scm`

Demo: Do as you please!

- Look at `config-desktop.scm`.
- Change!
 - ▶ change locales
 - ▶ add `pari-gp` package to global list
 - ▶ auto-login the user `enge`
- `guix system vm config-pari.scm`
- `guix system reconfigure config-pari.scm`

Packaging and deploying with GNU Guix and GuixSD

- 1 GNU Guix: Principles and package management
- 2 GNU GuixSD: Declarative system deployments
- 3 State of the nation

Timeline

- 2012-11: GNU project by Ludovic Courtès
- 2013-01: release 0.1
- 2014-07: installable operating system
- 2015-01: ARMv7 port
- 2016-01: successful fundraiser for new build farm
- 2016-02: creation of Guix Europe, non-profit organisation to support GNU Guix
- 2016-03: release 0.10.0

Status and activities

- 3200 packages
- 4 platforms
 - ▶ GuixSD: x86_64, i686
 - ▶ Guix as package manager: mips64el, armhf
- \approx 25 contributors per month
- \approx 400 commits per month
- lots of friendly people on the mailing list and on IRC

- 3200 packages
- 4 platforms
 - ▶ GuixSD: x86_64, i686
 - ▶ Guix as package manager: mips64el, armhf
- \approx 25 contributors per month
- \approx 400 commits per month
- lots of friendly people on the mailing list and on IRC

Try it out!

- <https://gnu.org/software/guix/>
- It is free
 - ▶ as in **libre**
 - ▶ of **cost**
 - ▶ of **risk**



Copyright © 2016 Andreas Enge andreas.enge@inria.fr
Some slides are © 2010, 2012–2016 Ludovic Courtès ludo@gnu.org

GNU GuixSD logo © 2015 Luis Felipe López Acevedo [<felipe.lopez@opmbx.org>](mailto:felipe.lopez@opmbx.org)
CC-BY-SA 4.0, <http://gnu.org/s/guix/graphics>

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](https://creativecommons.org/licenses/by-sa/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](https://www.gnu.org/licenses/gfdl.html) 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/cgi/guix/maintenance.git>.