

FIXME

Wed, November 6, 2019

# Présentation de Guix

PirBoazo

# Gnu Guix SD

- Agenda
  - 1)Introduction
  - 2)Guix vs Nixos
  - 3)Installation
  - 4)Concepts
  - 5)Utilisation courante
  - 6)Liens & référence

# GNU Guix

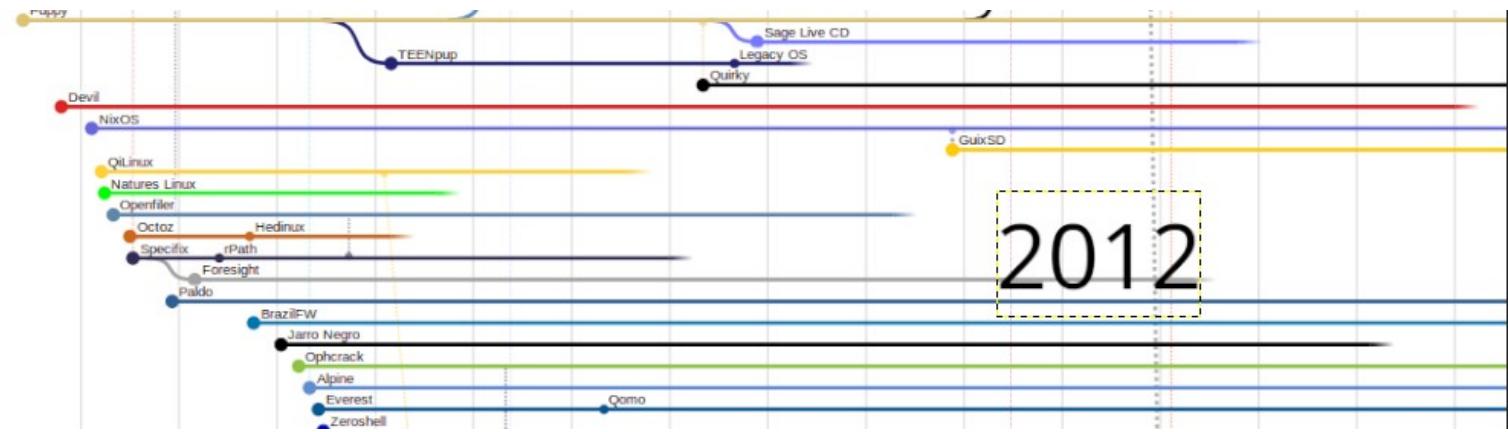
GNU Guix est :

un gestionnaire de paquets transactionnels  
une distribution avancee du systeme GNU qui respecte la  
liberte des utilisateurs.

un projet jeune

un fork de Nixos

2004



## Introduction

Guix peut etre utilise sur n'importe quel systeme executant le noyau Linux, ou il peut être utilisé comme une distribution de systeme d'exploitation autonome pour plusieurs architectures

i686,

x86\_64,

ARMv7,

Aarch64

La distribution fournit les paquets, ~11000, cœur de GNU .

## Introduction

En plus des fonctionnalités de gestion des paquets standard, Guix prend en charge :

1. les mises à niveau et les annulations transactionnelles,
2. la gestion des paquets non privilégiés,
3. les profils par utilisateur
4. le nettoyage des espaces.

Utilisé en tant que distribution GNU/Linux autonome, Guix offre une approche déclarative.

# Introduction

Guix System utilise un mécanisme de configuration où tous les aspects de la configuration globale du système sont déclarés en un seul endroit. (`/etc/config.scm`)

Cela permet les mises à jour transactionnelles du système ce qui rend possible le fait de revenir en arrière.

Et rend facile la réplication de la même configuration sur plusieurs machines différentes

## Fonctionnalité

- |  |   |
|--|---|
| • Utiliser le système de configuration | Personnaliser votre système GNU.                              |
| • Référence de système d'exploitation  | Détail sur la déclaration de système d'exploitation.          |
| • Systèmes de fichiers                 | Configurer les montages de systèmes de fichiers.              |
| • Périphériques mappés                 | Gestion des périphériques de bloc.                            |
| • Comptes utilisateurs                 | Spécifier des comptes utilisateurs.                           |
| • Disposition du clavier               | La manière dont le système interprète les touches du clavier. |
| • Régionalisation                      | Paramétrier la langue et les conventions culturelles.         |
| • Services                             | Spécifier les services du système.                            |
| • Programmes setuid                    | Programmes tournant avec les priviléges root.                 |
| • Certificats X.509                    | Authentifier les serveurs HTTPS.                              |
| • Name Service Switch                  | Configurer le « name service switch » de la libc.             |
| • Disque de RAM initial                | Démarrage de Linux-Libre.                                     |
| • Configuration du chargeur d'amorçage | Configurer le chargeur d'amorçage.                            |
| • Invoquer guix system                 | Instantier une configuration du système.                      |
| • Invoking guix deploy                 | Deploying a system configuration to a remote host.            |
| • Lancer Guix dans une VM              | Comment lancer Guix dans une machine virtuelle.               |
| • Définir des services                 | Ajouter de nouvelles définitions de services.                 |

## Introduction

Guix est, au départ , **une distribution basée sur les sources** avec des substituts, et à ce titre, la compilation de paquets à partir de leur code source fait partie des installations et mises a jour regulières de paquets.

En effet Guix prend en charge le **déploiement source/binaire** de manière transparente, ce qui signifie qu'il peut soit construire des choses localement, soit télécharger des éléments pré-construits depuis un serveur, ou les deux.

Ces éléments **préfabriques appelés substituts** - ce sont des substituts aux résultats de construction locaux.

Dans de nombreux cas, le téléchargement d'un substitut est beaucoup plus rapide que la construction locale.

Les substituts peuvent être tout ce qui résulte d'une construction de dérivation (voir [Derivations](#))

## Guix vs Nixos

## Les différences

- Les packages Libre(~11000) vs libre-Nonlibre (~40000)
- Le versionning de la distribution Commit vs version Bi-annuelle
- La gestion des services shepherd vs systemd
- La documentation multi-langue vs anglais
- La gestion du support mail-list vs discourse
- Le langage utilisé “guile Scheme” vs “Nix Expression Language”

- L'organisation système des répertoires */var/guix* vs */nix/var/nix*

```
[pboizot@nixos-1909:/nix/var/nix]$ ls -la
total 40
drwxr-xr-x 10 root root 4096 Nov  3 16:58 .
drwxr-xr-x  4 root root 4096 Sep 29  2018 ..
drwxr-xr-x  2 root root 4096 Sep 29  2018 channel-cache
drwxr-xr-x  2 root root 4096 Nov  3 16:58 daemon-socket
drwxr-xr-x  2 root root 4096 Nov  4 05:36 db
-rw-----  1 root root    0 Sep 29  2018 gc.lock
drwxr-xr-x  5 root root 4096 Nov  3 16:58 gcroots
drwxr-xr-x  2 root root 4096 Sep 29  2018 manifests
drwxr-xr-x  3 root root 4096 Nov  1 14:42 profiles
drwxr-xr-x  2 root root 4096 Nov  4 05:36 temproots
drwxr-xr-x  2 root root 4096 Sep 30  2018 userpool
```

```
[pboizot@nixos-1909:/nix/var/nix]$ █
```

```
/var/guix
root@guix-virt /var/guix# ls -la
total 36
drwxr-xr-x  9 root root 4096 Jun 12 16:53 .
drwxr-xr-x 10 root root 4096 Nov  4 08:19 ..
drwxr-xr-x  2 root root 4096 Nov  4 08:19 daemon-socket/
drwxr-xr-x  2 root root 4096 Nov  4 09:30 db/
-rw-----  1 root root    0 Jun 12 16:51 gc.lock
drwxr-xr-x  3 root root 4096 Aug 27 18:46 gcroots/
drwxr-xr-x  3 root root 4096 Aug 27 18:46 profiles/
drwxr-xr-x  3 root root 4096 Jun 12 16:51 substitute/
drwxr-xr-x  2 root root 4096 Nov  4 09:30 temproots/
drwxr-xr-x  2 root root 4096 Jun 12 16:53 userpool/
root@guix-virt /var/guix# █
```

# Installation

Installation d'une VM a partir de l'iso telechargeable : [

[https://guix.gnu.Org/download/Khttps://guix.gnu.org/download/ ici\)](https://guix.gnu.Org/download/Khttps://guix.gnu.org/download/)



**GNU Guix System 1.0.1**  
USB/DVD ISO installer of the standalone Guix System.  
Download options:  
[x86\\_64](#) [i686](#)  
Signatures: [x86\\_64](#) [i686](#)  
[Installation instructions.](#)



**GNU Guix 1.0.1 QEMU Image**  
QCOW2 virtual machine (VM) image.  
Download options:  
[x86\\_64](#)  
Signatures: [x86\\_64](#)  
[Installation instructions.](#)



**GNU Guix 1.0.1 Binary**  
Self-contained tarball providing binaries for Guix and its dependencies, to be installed on top of your Linux-based system.  
Download options:  
[x86\\_64](#) [i686](#) [armhf](#)  
[aarch64](#)  
Signatures: [x86\\_64](#) [i686](#) [armhf](#)  
[aarch64](#)  
[Installation instructions.](#)



**GNU Guix 1.0.1 Source**  
Source code distribution.  
Download options:  
[tarball](#)  
Signatures: [tarball](#)  
[Installation instructions.](#)

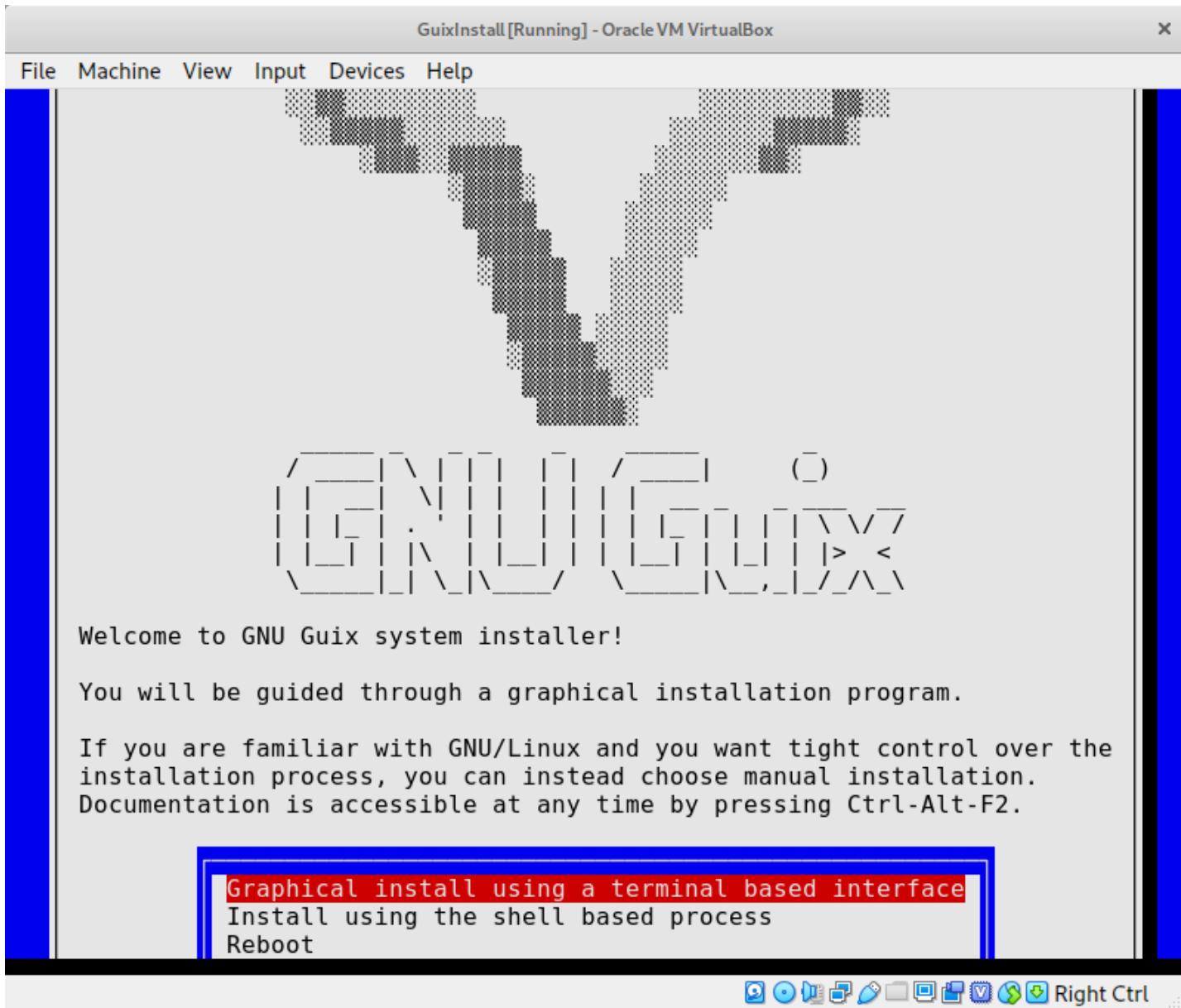
Source code and binaries for the Guix System distribution ISO image as well as GNU Guix can be found on the GNU servers at  
<https://ftp.gnu.org/gnu/guix/>. Older releases can still be found on [alpha.gnu.org](http://alpha.gnu.org).

# Installation

Elle est en mode caractere.  
Durant cette installation on choisi

- la langue , le clavier
- le formatage du disque
- le nom de serveur
- Le window manager
- Les utilisateurs
- Des services de reseau

# Installation



# Installation

## -1 Partitioning method

Please select a partitioning method.



Exit

P e e R i g h t Ctrl

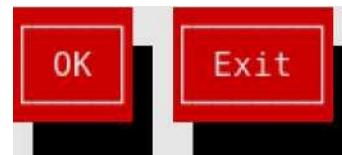
# Installation

## Manual partitioning

You can change a disk's partition table by selecting it and pressing ENTER. You can also edit a partition by selecting it and pressing ENTER, or remove it by pressing DELETE. To create a new partition, select a free space area and press ENTER.

At least one partition must have its mounting point set to '/'.

```
,TA VBOX HARDDISK (scsi) /dev/sda 53.7GB c
-
- 1 2G97kB biosgrub
- 2 33.GGB ext4 /
- 3 3999MB linux-swap(vl) swap
- 4 16.7GB ext4 /home
```



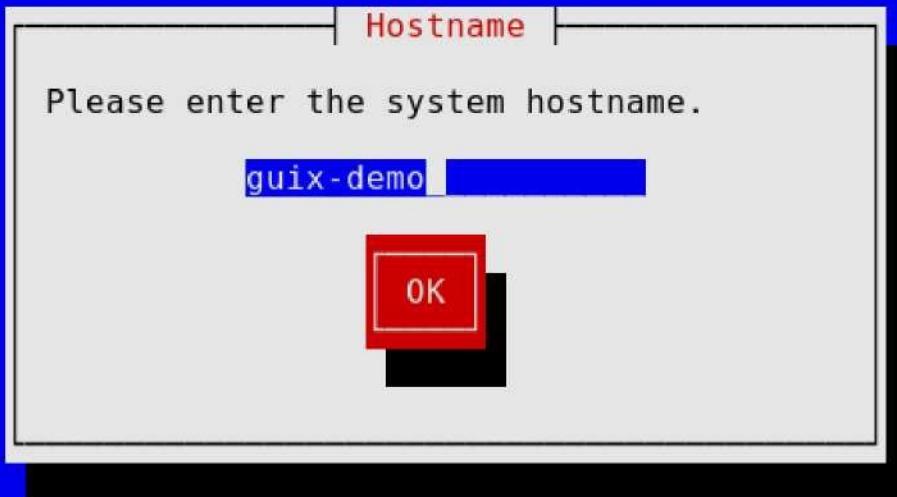
Pee®® Right Ctrl

# Installation

GuixInstall[Running] - OracleVM VirtualBox

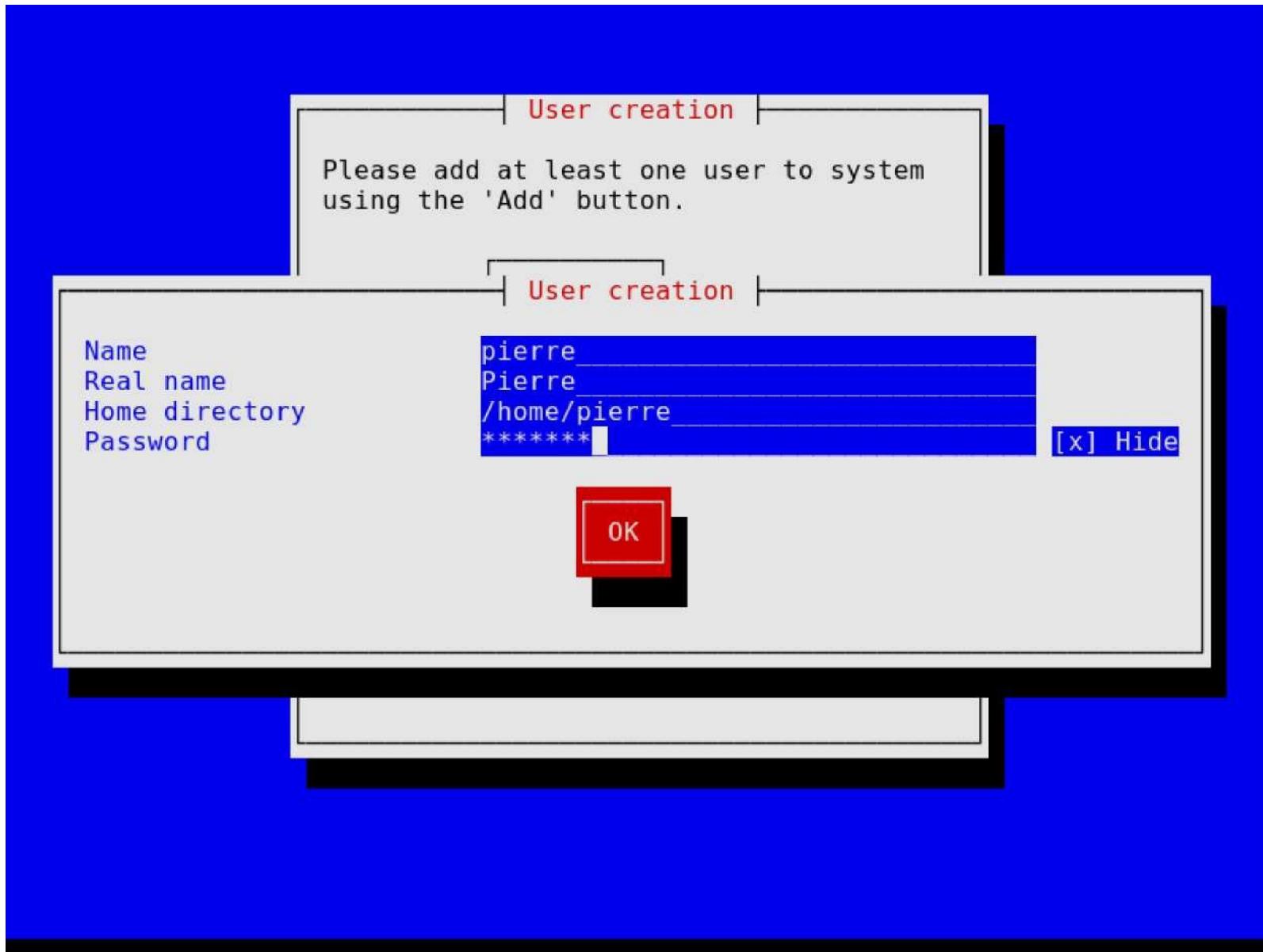
X

File Machine View Input Devices Help



Right Ctrl

# Installation



# Installation



# Installation

Le résultat un linux dont la configuration est dans un fichier qu'il faut modifier pour ajouter des utilisateurs mettre ses outils habituels

Exemples :

Ajout d'un groupe

```
(groups (cons (user-group(name "postgres" )) %base-groups))
```

Ajout d'un utilisateur

```
(user-account
```

```
    (name "demo")
```

```
    (comment "Demo user")
```

```
    (group "users")
```

```
    (home-directory "/home/demo")
```

```
    (supplementary-groups
```

```
        ("wheel" "netdev" "audio" "video"))
```

```
)
```

# Installation

Ajout de paquets, déclaration

```
(packages
  (append
    (list ( specification->package "NSS-CERTS" )
          ( specification->package "vim")
          ( specification->package "which")
          ( specification->package "terminator")
          ( specification->package "i3-wm")
          ( specification->package "i3status")
          ( specification->package "dmenu")
          ( specification->package "feh")
          ( specification->package "conky"))
    )
  %base-packages))
```

## Concept

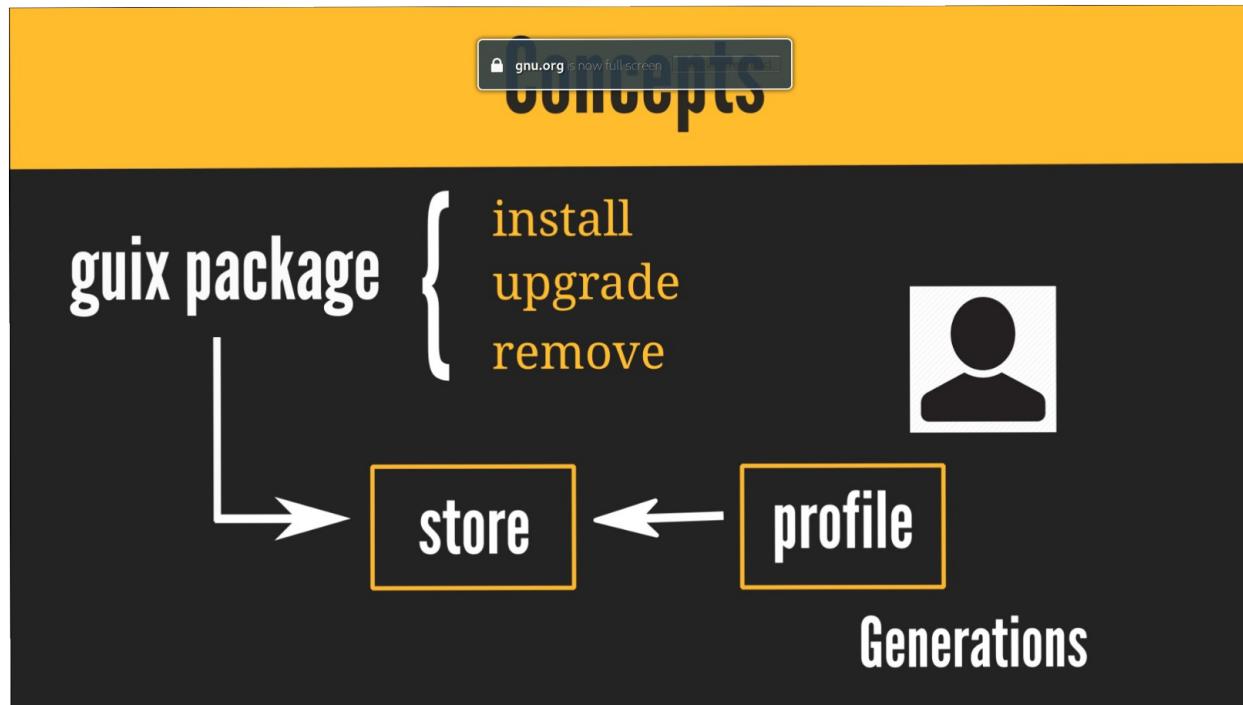
Avant d'aborder les commandes d'installation etc....

Parlons des concepts fondateurs de guix (nixos)

Store : */gnu/store*

Profile. */run/current-system/profile/bin*

Notion de génération ( system / package / profile )



## Concept

**Store** : */gnu/store*

```
devuser@guix-virt /gnu/store$ ls -la *git-mini*
-r--r--r-- 2 root root 1306 Jan 1 1970 0fgnh0s28h17lpzb9zsas8mwa4gsnr5i-
git-minimal-2.22.0drv
-r--r--r-- 2 root root 1450 Jan 1 1970 3jig1bvr1r15g9xl35cg84xlal415mvb-git-
minimal-2.22.1drv
-r--r--r-- 2 root root 10924 Jan 1 1970 40w2yirnnrwqijip11ra92linm6qdq3y-git-
minimal-2.23.0-guile-builder
-r--r--r-- 2 root root 10971 Jan 1 1970 6hh4c66834s58rpj57p33v4qdr8lalxb-git-
minimal-2.23.0-guile-builder
-r--r--r-- 2 root root 3035 Jan 1 1970 6p9nkw7y5zn0hhk4dplm2b7rzxzj3984-
git-minimal-2.22.0drv
```

Puisque Guix vous permet de conserver plusieurs générations de configurations de votre système (y compris tout l'historique de vos paquets), il peut être plus exigeant sur l'utilisation du disque que les autres systèmes d'exploitation.

## Concept

Profile. /run/current-system/profile/bin

ls -la

```
lrwxrwxrwx 2 root root 66 Jan 1 1970 yes ->
/gnu/store/9kzrrccpzl6i1sfwb0drb00gi2gwk0x0-coreutils-8.31/bin/yes
lrwxrwxrwx 3 root root 62 Jan 1 1970 zcat ->
/gnu/store/py3k9zla9fj3z7430v4crqj5pyrsd3qj-gzip-1.10/bin/zcat
lrwxrwxrwx 3 root root 62 Jan 1 1970 zcmp ->
/gnu/store/py3k9zla9fj3z7430v4crqj5pyrsd3qj-gzip-1.10/bin/zcmp
lrwxrwxrwx 3 root root 63 Jan 1 1970 zdiff ->
```

## Concept

Notion de génération ( system )

guix system list-generations

Generation 13 Aug 27 2019 18:42:02

file name: /var/guix/profiles/system-13-link

canonical file name: /gnu/store/mhaj2kg4xgbv3gzgq62pxfnfrsqqi1fb-system

label: GNU with Linux-Libre 5.2.10

bootloader: grub

root device: UUID: 2cab8a75-2dbf-4d03-afb0-58377abeee0e

kernel:

/gnu/store/ms7naxaxgqf7f7zhyc5j4xkx0wm0gg24-linux-libre-5.2.10/bzImage

Generation 14 Nov 04 2019 10:22:01

file name: /var/guix/profiles/system-14-link

canonical file name: /gnu/store/zmric2s1cnj738vgwzmag6vbadqjlkgd-system

label: GNU with Linux-Libre 5.3.8

bootloader: grub

root device: UUID: 2cab8a75-2dbf-4d03-afb0-58377abeee0e

kernel:

/gnu/store/9l8z8k5ya6g9vwzn1jcba9qdlfhf6xhd-linux-libre-5.3.8/bzImage

## Concept

Notion de génération ( system / package / profile )

Package d'un utilisateur:

```
guix package --list-generations
```

Generation 1 Nov 04 2019 11:09:44

```
  eolie  0.9.63  out /gnu/store/jqcwkj5rsnky5ip75ibwi7p12paagb4v-eolie-  
0.9.63
```

Generation 2 Nov 04 2019 11:19:54

```
  + icecat 68.2.0-guix0-preview3 out
```

```
/gnu/store/i5ymbcddvhqkzwhcij6g5a6b636idqp6-icecat-68.2.0-guix0-preview3
```

Generation 3 Nov 04 2019 11:42:28 (current)

```
  + git    2.23.0  out /gnu/store/1k0hn1qfy4phzdydrwsdr6vr7rvbcw3v-git-  
2.23.0
```

## Concept

Notion de génération ( profile )

/home/devuser/.config/guix/current/bin:

/home/devuser/.guix-profile/bin

.guix-profile -> /var/guix/profiles/per-user/devuser/guix-profile

ls /var/guix/profiles/per-user/devuser/guix\*

**/var/guix/profiles/per-user/devuser/guix-profile:**

**bin/ etc/ lib/ libexec/ manifest share/**

**/var/guix/profiles/per-user/devuser/guix-profile-1-link:**

**bin etc/ lib/ libexec manifest share/**

**/var/guix/profiles/per-user/devuser/guix-profile-2-link:**

**bin/ etc/ lib/ libexec manifest share/**

**/var/guix/profiles/per-user/devuser/guix-profile-3-link:**

**bin/ etc/ lib/ libexec/ manifest share/**

# Utilisation Courante

- Où sont les binaires?
  - which which
  - /run/current-system/profile/bin/which
- A quoi a accès un utilisateur ?
  - echo \$PATH
  - /run/setuid-programs:/home/demo/.config/guix/current/bin:/home/demo/.guix-profile/bin:/home/demo/.guix-profile/sbin:/run/current-system/profile/bin:/run/current-system/profile/sbin
- Que faire si l'application n'est pas accessible pour l'utilisateur?
  - L'installer dans l'environnement de l'utilisateur
- Comment mettre à jour le système ?
  - Ajouter des utilisateurs
  - Update des Packages

## Utilisation courante

# Mettre à jour le systeme.

En tant que root passer ces deux commandes.

```
guix pull  
guix system reconfigure /etc/config.scm
```

Nota : si on veut mettre à jour le systeme pour une version particulière  
Guix pull --commit=71adb4228ab43ded5dd3bfcbdc3a368280d51513

Les paquets seront mise à jour.

La mise à jour d'une autre machine pour un même état est donc possible.  
guix pull --commit=numero

Recopie du fichier config.scm puis exécution de la commande.

## Utilisation courante

# Ajouter des utilisateurs ou des paquets

En tant que root

Editer /etc/config.scm

Puis lancer

guix system reconfigure /etc/config.scm

#### Managing the Operating System

```
guix system search regexp
  search for services matching regexp
guix system reconfigure file
  reconfigure the OS according to the configuration in file
guix system list-generations [pattern]
  list OS generations matching pattern—e.g., 1m for one month
guix system roll-back
  roll back to the previous system generation
guix system delete-generations pattern
  delete generations matching pattern
guix system build file
  build the OS declared in file
```

#### Building and Running Containers

```
guix system container file
  produce a script that runs the OS declared in file in a container
guix system docker-image file
  build a Docker image of the OS declared in file
```

#### Building Virtual Machines

```
guix system vm file
  produce a script that runs the OS declared in file in a VM
guix system vm-image file
  produce a QCOW2 image of the OS in file
```

#### Building Operating System Images

```
guix system disk-image file
  create a raw disk image for the OS declared in file
guix system disk-image --file-system-type=iso9660 file
  create an ISO CD/DVD image for the OS declared in file
```

#### Inspecting an Operating System

```
guix system extension-on-graph file
  show the graph of services extensions for the OS in file
guix system shepherd-d-graph file
  show the dependency graph of Shepherd services for file
```

#### Declaring an Operating System

guix system takes a configuration file that declares the *complete* configuration of an operating system, along these lines:

```
(use-modules (gnu)
  (use-service-modules networking ssh)
  (use-package-modules certs screen))

(operating-system
  (host-gnus)
  (timezone "Europe/Berlin")
  (locale "en_US.UTF-8")
  (keyboard-layout (keyboard-layout "us" "altgr-intl"))

  (bootloader
    (bootloader-configuration
      (bootloader grub-efi-bootloader)
      (device (file-system (mount-point "/"))
              (type "ext4")))
    %base-bootloader)

  (file-systems (cons (file-system
    (device (file-system-label "my-root"))
    (mount-point "/")
    (type "ext4"))
    %base-file-systems))

  (users (cons (user-account
    (name "charlie")
    (comment "Charlie Smith")
    (group "users")
    (supplementary-groups ("wheel" "audio" "video")))
    %base-user-accounts))

  ; Globally installed packages.
  (packages (append (list screen nss-certs)
    %base-packages)))

  ; System services: add sshd and DHCP to the base services.
  (services (append (list (service dhcp-client-service-type)
    (service openssh-service-type
      (openssh-configuration
        (port-number 2222)))
    %base-services))))
```

Copyright © 2018, 2019 Ludovic Courtès <ludo@gnu.org>.  
Permission is granted to 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 <https://gnu.org/licenses/gfdl.html>.  
The source of this document is available from  
<https://git.savannah.gnu.org/cgit/guix/manualsrcs.git>.



## **Utilisation courante**

La commande guix pour l'installation, la mise à jour et la suppression d'un package

```
guix install NomduPaquet  
guix upgrade NomduPaquet  
guix remove NomduPaquet
```

Ecoutez deux courtes video sur le sujet : [Video1](#) & [Video2](#)

## **Utilisation courante**

Il est de plus possible d'installer des paquets dans  
**un profil**

Exemple :

```
guix install postgresql@10.8 -p ~/app/postgresql/version/10
```

On devra pour utiliser le profil le positionner.

Vous trouverez une [explication détaillée](#) sur le site de guix

## **un environnement**

guix environement –pure emacs which etc...

Vous avez dit utilisation courante ? .....

Trois video sur le sujet creation de paquets

How to set up a development environment for GNU Guix ?

How to create a package recipe for not yet packaged software ?

How to submit a package for inclusion in the GNU Guix distribution ?

# Utilisation courante

```
/gnu/store/n1c9jiv2njnvdzf58v71fvzq0hkgjvz1-bash-5.0.7/bin/bash  
/gnu/store/n1c9jiv2njnvdzf58v71fvzq0hkgjvz1-bash-5.0.7/bin/bash 72x19  
devuser@guix-virt ~$ guix environment --pure guix --ad-hoc coreutils findutils which  
devuser@guix-virt ~ [env]$ echo $PATH  
/gnu/store/cxia42rbhwzbzmmwmbihxwhxwmvi43qbc-profile/bin:/gnu/store/cxia42rbhwzbzmmwmbihxwhxwmvi43qbc-profile/sbin  
devuser@guix-virt ~ [env]$ which vim  
which: no vim in (/gnu/store/cxia42rbhwzbzmmwmbihxwhxwmvi43qbc-profile/bin:  
/gnu/store/cxia42rbhwzbzmmwmbihxwhxwmvi43qbc-profile/sbin)  
devuser@guix-virt ~ [env]$ which guile  
/gnu/store/cxia42rbhwzbzmmwmbihxwhxwmvi43qbc-profile/bin/guile  
devuser@guix-virt ~ [env]$ █
```

```
/gnu/store/n1c9jiv2njnvdzf58v71fvzq0hkgjvz1-bash-5.0.7/bin/bash 72x19  
devuser@guix-virt ~$ echo $PATH  
/run/setuid-programs:/home/devuser/.config/guix/current/bin:/home/devuser/.guix-profile/bin:/run/current-system/profile/bin:/run/current-system/profile/sbin  
devuser@guix-virt ~$ which vim  
/run/current-system/profile/bin/vim  
devuser@guix-virt ~$ which guile  
/run/current-system/profile/bin/guile  
devuser@guix-virt ~$ █
```

Invoquer guix environnement (Manuel de référence de GNU Guix) - GNU IceCat

Invoquer guix environnement | x Guix Profiles in Practice | x Linux Commando: How to | +

https://guix.gnu.org/manual/fr/html\_node/Invoquer.html

Suivant: [Invoquer guix pack](#), Monter: [Développement](#) [[Table des matières](#)][[Index](#)]

## 5.1 Invoquer guix environnement

Le but de `guix environment` est d'assister les hackers dans la création d'environnements de développement reproductibles sans polluer leur profil de paquets. L'outil `guix environment` prend un ou plusieurs paquets, construit leurs entrées et crée un environnement shell pour pouvoir les utiliser.

La syntaxe générale est :

```
guix environment options paquet...
```

L'exemple suivant crée un nouveau shell préparé pour le développement de GNU Guile :

```
guix environment guile
```

Si les dépendances requises ne sont pas déjà construites, `guix environment` les construit automatiquement. L'environnement du nouveau shell est une version améliorée de l'environnement dans lequel `guix environment` a été lancé. Il contient les chemins de recherche nécessaires à la construction du paquet donné en plus des variables d'environnement existantes. Pour créer un environnement « pur », dans lequel les variables d'environnement de départ ont été nettoyées, utilisez l'option `--pure`<sup>10</sup>.

`guix environment` définit la variable `GUIX_ENVIRONMENT` dans le shell qu'il crée ; sa valeur est le nom de fichier du profil de cet environnement. Cela permet aux utilisateurs, disons, de définir un prompt spécifique pour les environnements de développement dans leur `.bashrc` (voir [Bash Startup Files](#) dans [The GNU Bash Reference Manual](#)) :

```
if [ -n "$GUIX_ENVIRONMENT" ]  
then  
    export PS1="\u@\h \w [dev]\$ "  
fi
```

... ou de naviguer dans le profil :

```
$ ls "$GUIX_ENVIRONMENT/bin"
```

De surcroît, plus d'un paquet peut être spécifié, auquel cas l'union des entrées des paquets donnés est utilisée. Par exemple, la commande ci-dessous crée un shell où toutes les dépendances de Guile et Emacs sont disponibles :

## Utilisation courante

# manifests & Channel

Les outils pour construire des Profils reproductibles

Pour reproduire un profil bit à bit, nous avons besoin de deux informations :

un manifest,  
une spécification de channel Guix.

## Un manifest

```
(specifications->manifest  
'("emacs" "python@3.7" "git"))
```

## Le channel.

```
guix describe --format=channels >spec-channel.scm  
cat spec-channel.scm  
(list (channel  
      (name 'guix)  
      (url "https://git.savannah.gnu.org/git/guix.git")  
      (commit  
        "e5c447907c0649ea3f662e1c27a25d0113ec921f")))
```

## Utilisation courante

```
mkdir test  
GUIX_EXTRA_PROFILES=$HOME/test  
mkdir -p "$GUIX_EXTRA_PROFILES"/my-project  
guix package --manifest=manifest1.scm –  
profile="$GUIX_EXTRA_PROFILES"/my-project/test
```

```
installing new manifest from 'manifest1.scm' with 3 entries  
substitute: updating substitutes from 'https://ci.guix.gnu.org'... 100.0%  
substitute: updating substitutes from 'https://ci.guix.gnu.org'... 100.0%  
The following derivation will be built:  
/gnu/store/3fkk8raj1j93dwdjl4i084p40c6bk07-profile.drv  
11.9 MB will be downloaded:  
building /gnu/store/3fkk8raj1j93dwdjl4i084p40c6bk07-profile.drv...  
....  
3 packages in profile
```

```
hint: Consider setting the necessary environment variables by running:  
  GUIX_PROFILE="/home/devuser/test/my-project/test"  
  . "$GUIX_PROFILE/etc/profile"  
Alternately, see `guix package --search-paths -p  
"/home/devuser/test/my-project/test"'.  
.....
```

## Utilisation courante

### shepherd & herd

```
root@guix-virt ~# shepherd --help
```

```
shepherd [OPTIONS...]
```

This is a service manager for Unix and GNU.

-s, --socket=FILE	get commands from socket FILE or from stdin (-)
-c, --config=FILE	read configuration from FILE
--pid[=FILE]	when ready, write PID to FILE or stdout
-l, --logfile=FILE	log actions in FILE
-I, --insecure	don't ensure that the setup is secure
-S, --silent	don't do output to stdout
--quiet	synonym for --silent
-p, --persistency[=FILE]	use FILE to load and store state
--help	display this help and exit
--usage	display short usage message and exit
--version	display version information and exit

Mandatory or optional arguments to long options are also mandatory or optional to the corresponding short options.

Report bugs to: [bug-guix@gnu.org](mailto:bug-guix@gnu.org) .

GNU Shepherd general home page: <<http://www.gnu.org/software/shepherd/>>

General help using GNU software: <<http://www.gnu.org/gethelp/>>

## Utilisation courante

### shepherd & herd

```
root@guix-virt ~# herd --help
herd [OPTIONS...] ACTION SERVICE [ARG...]
Apply ACTION (start, stop, status, etc.) on \
SERVICE with the ARGs.
```

- s, --socket=FILE      send commands to FILE
- help                  display this help and exit
- usage                display short usage message and exit
- version              display version information and exit

Mandatory or optional arguments to long options are also mandatory or optional to the corresponding short options.

Report bugs to: [bug-guix@gnu.org](mailto:bug-guix@gnu.org) .

GNU Shepherd general home page: <<http://www.gnu.org/software/shepherd/>>

General help using GNU software: <<http://www.gnu.org/gethelp/>>

# Où trouver de l'information

## Le site officiel

Le site guix- <https://www.gnu.org/software/guix/>  
le blog - <https://www.gnu.org/software/guix/blog/>

## La documentation

En Frangais -

[https://www.gnu.org/software/guix/manual/fr/html\\_node](https://www.gnu.org/software/guix/manual/fr/html_node)

En Anglais -

[https://www.gnu.org/software/guix/manual/en/html\\_node/](https://www.gnu.org/software/guix/manual/en/html_node/)

cette documentation existe aussi en allemand, espagnol chinois et russe.

guix referencecard - <https://www.gnu.org/software/guix/guix-refcard.pdf>

[GNU Guix Cookbook](#)

# Articles ou video

Abordant l'organisation de guix

[Guix : un outil pour les remplacer tous](#)

[EveryDay use of Guix](#)

[Gestion de paquets sure et flexible avec Gnu guix](#)

[Guix System, la 2eme distribution Gnu/Linux fonctionnelle](#)

[Guix Profiles in Practice](#)

[Guix-advance](#)