Installation binaire de Guix sur Debian

- Objet : installer le package manager Guix sur Debian
- Niveau requis :
  débutant, avisé
- Commentaires : *https://git.savannah.gnu.org/cgit/guix.git/plain/etc/guix-install.sh*
- Débutant, à savoir : Utiliser GNU/Linux en ligne de commande, tout commence là !.

Introduction

Il s'agit de récupérer un script shell d'installation, de le rendre exécutable et de le lancer. Celà installera Guix aux côtés des autres gestionnaires de paquetage sans les altérer de quelque façon que ce soit.

Sources :

https://www.gnu.org/software/guix/manual/fr/html_node/Installation-binaire.html#Installation-binaire

https://git.savannah.gnu.org/cgit/guix.git/plain/etc/guix-install.sh

J'ai copié-collé le script dans un nouveau fichier à l'aide de vim, vous pouvez le copier-coller d'ici si vous avez la flemme de le télécharger :

```
$ su
```

```
# vim guix-install.sh
```

guix-install.sh

```
#!/bin/sh
# GNU Guix --- Functional package management for GNU
# Copyright © 2017 sharlatan <sharlatanus@gmail.com>
# Copyright © 2018 Ricardo Wurmus <rekado@elephly.net>
# Copyright © 2018 Efraim Flashner <efraim@flashner.co.il>
#
# This file is part of GNU Guix.
#
# GNU Guix is free software; you can redistribute it and/or modify it
# under the terms of the GNU General Public License as published by
# the Free Software Foundation; either version 3 of the License, or (at
# your option) any later version.
#
# GNU Guix is distributed in the hope that it will be useful, but
# WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
# GNU General Public License for more details.
#
# You should have received a copy of the GNU General Public License
```

```
# along with GNU Guix.  If not, see <http://www.gnu.org/licenses/>.

# We require Bash but for portability we'd rather not use /bin/bash or
# /usr/bin/env in the shebang, hence this hack.
if [ "x$BASH_VERSION" = "x" ]
then
    exec bash "$0" "$@"
fi

set -e

[ "$UID" -eq 0 ] || { echo "This script must be run as root."; exit 1;
}

REQUIRE=(
    "dirname"
    "readlink"
    "wget"
    "gpg"
    "grep"
    "which"
    "sed"
    "sort"
    "getent"
    "mktemp"
    "rm"
    "chmod"
    "uname"
    "groupadd"
    "tail"
    "tr"
)

PAS=$'[ \033[32;1mPASS\033[0m ] '
ERR=$'[ \033[31;1mFAIL\033[0m ] '
INF="[ INFO ] "

DEBUG=0
GNU_URL="https://ftp.gnu.org/gnu/guix/"
OPENPGP_SIGNING_KEY_ID="3CE464558A84FDC69DB40CFB090B11993D9AEBB5"

# This script needs to know where root's home directory is.  However,
we
# cannot simply use the HOME environment variable, since there is no
guarantee
# that it points to root's home directory.
ROOT_HOME="$(echo ~root)"

# ----------------------------------------------------------------------
---------
```

```
#+UTILITIES

_err()
{ # All errors go to stderr.
    printf "[%s]: %s\n" "$(date +%s.%3N)" "$1"
}

_msg()
{ # Default message to stdout.
    printf "[%s]: %s\n" "$(date +%s.%3N)" "$1"
}

_debug()
{
    if [ "${DEBUG}" = '1' ]; then
        printf "[%s]: %s\n" "$(date +%s.%3N)" "$1"
    fi
}


chk_require()
{ # Check that every required command is available.
    declare -a cmds
    declare -a warn

    cmds=(${1})

    _debug "--- [ $FUNCNAME ] ---"

    for c in ${cmds[@]}; do
        command -v "$c" &>/dev/null || warn+=("$c")
    done

    [ "${#warn}" -ne 0 ] &&
        { _err "${ERR}Missing commands: ${warn[*]}.";
          return 1; }

    _msg "${PAS}verification of required commands completed"

    gpg --list-keys ${OPENPGP_SIGNING_KEY_ID} >/dev/null 2>&1 || (
        _err "${ERR}Missing OpenPGP public key.  Fetch it with this
command:"
        echo "  gpg --keyserver pool.sks-keyservers.net --recv-keys
${OPENPGP_SIGNING_KEY_ID}"
        exit 1
    )
}

chk_term()
{ # Check for ANSI terminal for color printing.
    local ansi_term
```

```
    if [ -t 2 ]; then
        if [ "${TERM+set}" = 'set' ]; then
            case "$TERM" in
                xterm*|rxvt*|urxvt*|linux*|vt*|eterm*|screen*)
                    ansi_term=true
                    ;;
                *)
                    ansi_term=false
                    ERR="[ FAIL ] "
                    PAS="[ PASS ] "
                    ;;
            esac
        fi
    fi
}

chk_init_sys()
{ # Return init system type name.
    if [[ $(/sbin/init --version 2>/dev/null) =~ upstart ]]; then
        _msg "${INF}init system is: upstart"
        INIT_SYS="upstart"
        return 0
    elif [[ $(systemctl) =~ -\.mount ]]; then
        _msg "${INF}init system is: systemd"
        INIT_SYS="systemd"
        return 0
    elif [[ -f /etc/init.d/cron && ! -h /etc/init.d/cron ]]; then
        _msg "${INF}init system is: sysv-init"
        INIT_SYS="sysv-init"
        return 0
    else
        INIT_SYS="NA"
        _err "${ERR}Init system could not be detected."
    fi
}

chk_sys_arch()
{ # Check for operating system and architecture type.
    local os
    local arch

    os="$(uname -s)"
    arch="$(uname -m)"

    case "$arch" in
        i386 | i486 | i686 | i786 | x86)
            local arch=i686
            ;;
        x86_64 | x86-64 | x64 | amd64)
```

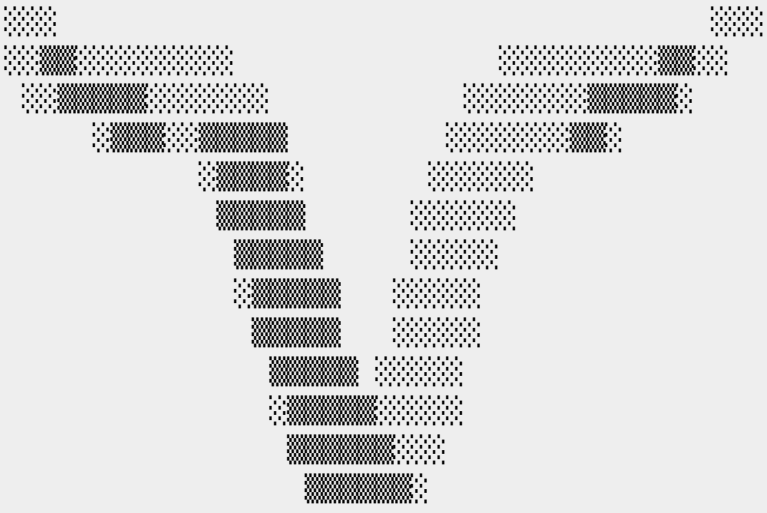
```
            local arch=x86_64
            ;;
        aarch64)
            local arch=aarch64
            ;;
    armv7l)
        local arch=armhf
        ;;
        *)
            _err "${ERR}Unsupported CPU type: ${arch}"
            exit 1
    esac

    case "$os" in
        Linux | linux)
            local os=linux
            ;;
        *)
            _err "${ERR}Your operation system (${os}) is not
supported."
            exit 1
    esac

    ARCH_OS="${arch}-${os}"
}

# ------------------------------------------------------------------------
---------
#+MAIN

guix_get_bin_list()
{ # Scan GNU archive and save list of binaries
    local gnu_url="$1"
    local -a bin_ver_ls
    local latest_ver
    local default_ver

    _debug "--- [ $FUNCNAME ] ---"

    # Filter only version and architecture
    bin_ver_ls=("$(wget -qO- "$gnu_url" \
        | sed -n -e 's/.*guix-binary-\([0-9.]*\)\..*.tar.xz.*/\1/p' \
        | sort -Vu)")

    latest_ver="$(echo "$bin_ver_ls" \
                      | grep -oP "([0-9]{1,2}\.){2}[0-9]{1,2}" \
                      | tail -n1)"

    default_ver="guix-binary-${latest_ver}.${ARCH_OS}"

    if [[ "${#bin_ver_ls}" -ne "0" ]]; then
```

```
        _msg "${PAS}Release for your system: ${default_ver}"
    else
        _err "${ERR}Could not obtain list of Guix releases."
        exit 1
    fi

    # Use default to download according to the list and local ARCH_OS.
    BIN_VER="$default_ver"
}

guix_get_bin()
{ # Download and verify binary package.
    local url="$1"
    local bin_ver="$2"
    local dl_path="$3"

    _debug "--- [ $FUNCNAME ] ---"

    _msg "${INF}Downloading Guix release archive"

    wget --help | grep -q '\--show-progress' && \
        _PROGRESS_OPT="-q --show-progress" || _PROGRESS_OPT=""
    wget $_PROGRESS_OPT -P "$dl_path" "${url}/${bin_ver}.tar.xz"
"${url}/${bin_ver}.tar.xz.sig"

    if [[ "$?" -eq 0 ]]; then
       _msg "${PAS}download completed."
    else
        _err "${ERR}could not download ${url}/${bin_ver}.tar.xz."
        exit 1
    fi

    pushd $dl_path >/dev/null
    gpg --verify "${bin_ver}.tar.xz.sig" >/dev/null 2>&1
    if [[ "$?" -eq 0 ]]; then
        _msg "${PAS}Signature is valid."
        popd >/dev/null
    else
        _err "${ERR}could not verify the signature."
        exit 1
    fi
}

sys_create_store()
{ # Unpack and install /gnu/store and /var/guix
    local pkg="$1"
    local tmp_path="$2"

    _debug "--- [ $FUNCNAME ] ---"
```

```
    cd "$tmp_path"
    tar --warning=no-timestamp \
        --extract \
        --file "$pkg" &&
    _msg "${PAS}unpacked archive"

    if [[ -e "/var/guix" || -e "/gnu" ]]; then
        _err "${ERR}A previous Guix installation was found.  Refusing
to overwrite."
        exit 1
    else
        _msg "${INF}Installing /var/guix and /gnu..."
        mv "${tmp_path}/var/guix" /var/
        mv "${tmp_path}/gnu" /
    fi

    _msg "${INF}Linking the root user's profile"
    mkdir -p "${ROOT_HOME}/.config/guix"
    ln -sf /var/guix/profiles/per-user/root/current-guix \
       "${ROOT_HOME}/.config/guix/current"

    GUIX_PROFILE="${ROOT_HOME}/.config/guix/current"
    source "${GUIX_PROFILE}/etc/profile"
    _msg "${PAS}activated root profile at
${ROOT_HOME}/.config/guix/current"
}

sys_create_build_user()
{ # Create the group and user accounts for build users.

    _debug "--- [ $FUNCNAME ] ---"

    if [ $(getent group guixbuild) ]; then
        _msg "${INF}group guixbuild exists"
    else
        groupadd --system guixbuild
        _msg "${PAS}group <guixbuild> created"
    fi

    for i in $(seq -w 1 10); do
        if id "guixbuilder${i}" &>/dev/null; then
            _msg "${INF}user is already in the system, reset"
            usermod -g guixbuild -G guixbuild           \
                    -d /var/empty -s "$(which nologin)" \
                    -c "Guix build user $i"             \
                    "guixbuilder${i}";
        else
            useradd -g guixbuild -G guixbuild           \
                    -d /var/empty -s "$(which nologin)" \
                    -c "Guix build user $i" --system    \
                    "guixbuilder${i}";
```

```
            _msg "${PAS}user added <guixbuilder${i}>"
        fi
    done
}

sys_enable_guix_daemon()
{ # Run the daemon, and set it to automatically start on boot.

    local info_path
    local local_bin
    local var_guix

    _debug "--- [ $FUNCNAME ] ---"

    info_path="/usr/local/share/info"
    local_bin="/usr/local/bin"
    var_guix="/var/guix/profiles/per-user/root/current-guix"

    case "$INIT_SYS" in
        upstart)
            { initctl reload-configuration;
              cp
"${ROOT_HOME}/.config/guix/current/lib/upstart/system/guix-daemon.conf"
\
                /etc/init/ &&
                 start guix-daemon; } &&
               _msg "${PAS}enabled Guix daemon via upstart"
            ;;
        systemd)
            { cp
"${ROOT_HOME}/.config/guix/current/lib/systemd/system/guix-
daemon.service" \
                /etc/systemd/system/;
              chmod 664 /etc/systemd/system/guix-daemon.service;
              systemctl daemon-reload &&
                  systemctl start guix-daemon &&
                  systemctl enable guix-daemon; } &&
                _msg "${PAS}enabled Guix daemon via systemd"
            ;;
        NA|*)
            _msg "${ERR}unsupported init system; run the daemon
manually:"
            echo "  ${ROOT_HOME}/.config/guix/current/bin/guix-daemon -
-build-users-group=guixbuild"
            ;;
    esac

    _msg "${INF}making the guix command available to other users"

    [ -e "$local_bin" ] || mkdir -p "$local_bin"
```

```
    ln -sf "${var_guix}/bin/guix"  "$local_bin"

    [ -e "$info_path" ] || mkdir -p "$info_path"
    for i in ${var_guix}/share/info/*; do
        ln -sf "$i" "$info_path"
    done
}

sys_authorize_build_farms()
{ # authorize the public keys of the two build farms
    while true; do
        read -p "Permit downloading pre-built package binaries from the
project's build farms? (yes/no) " yn
        case $yn in
            [Yy]*) guix archive --authorize <
"${ROOT_HOME}/.config/guix/current/share/guix/hydra.gnu.org.pub" &&
                          _msg "${PAS}Authorized public key for
hydra.gnu.org";
                   guix archive --authorize <
"${ROOT_HOME}/.config/guix/current/share/guix/ci.guix.gnu.org.pub" &&
                       _msg "${PAS}Authorized public key for
ci.guix.gnu.org";
                   break;;
            [Nn]*) _msg "${INF}Skipped authorizing build farm public
keys"
                   break;;
            *) _msg "Please answer yes or no.";
        esac
    done
}

welcome()
{
    cat<<"EOF"
```



```
     _____ _   _ _     _    _____       _
    / ____| \ | | |   | |  / ____|     (_)
   | |  __|  \| | |   | | | |  __ _   _ ___  __
   | | |_ | . ' | |   | | | | |_ | | | | \ \/ /
```

```
   | |__| | |\  | |__| | | |__| | |_| | |>  <
    \_____|_| \_|\____/   \_____|\__,_|_/_/\_\

This script installs GNU Guix on your system

https://www.gnu.org/software/guix/
EOF
    echo -n "Press return to continue..."
    read -r  ANSWER
}

main()
{
    local tmp_path
    welcome

    _msg "Starting installation ($(date))"

    chk_term
    chk_require "${REQUIRE[*]}"
    chk_init_sys
    chk_sys_arch

    _msg "${INF}system is ${ARCH_OS}"

    tmp_path="$(mktemp -t -d guix.XXX)"

    guix_get_bin_list "${GNU_URL}"
    guix_get_bin "${GNU_URL}" "${BIN_VER}" "$tmp_path"

    sys_create_store "${BIN_VER}.tar.xz" "${tmp_path}"
    sys_create_build_user
    sys_enable_guix_daemon
    sys_authorize_build_farms

    _msg "${INF}cleaning up ${tmp_path}"
    rm -r "${tmp_path}"

    _msg "${PAS}Guix has successfully been installed!"
    _msg "${INF}Run 'info guix' to read the manual."
 }

main "$@"
```

Ensuite j'ai rendu le fichier exécutable :

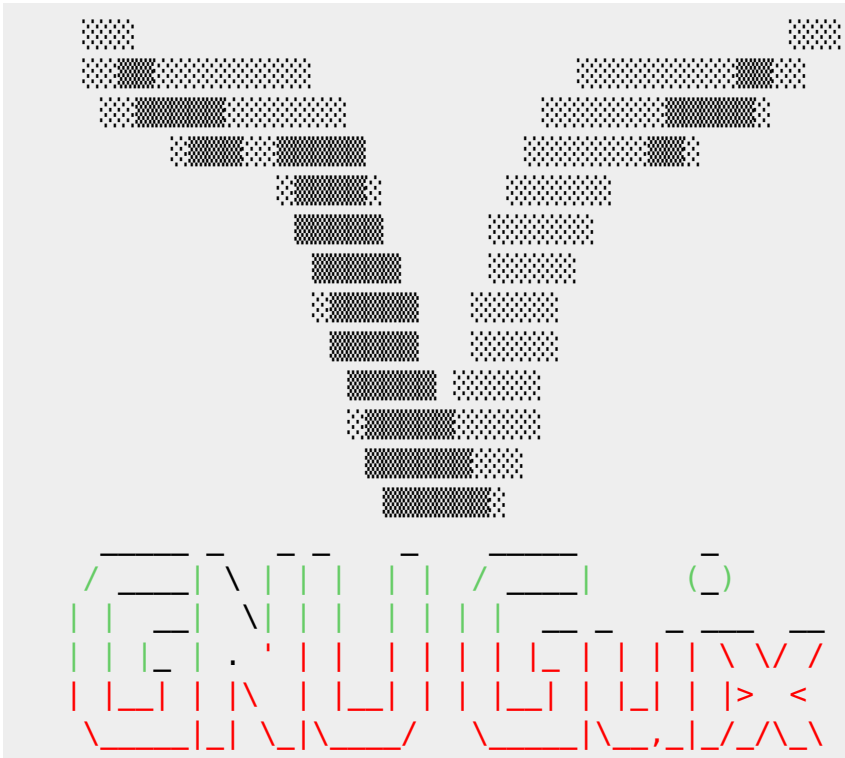
```
# chmod +x guix-install.sh
```

Installation

j'ai lancé :

```
# ./guix-install.sh
```

nom.sh



```
 _____ _   _ _     _    _____       _
/ ____| \ | | |   | |  / ____|     (_)
| |  __|  \| | |   | | | |  __ _   _ ___  __
| | |_ | . ' | |   | | | | |_ | | | | \ \/ /
| |__| | |\  | |__| | | |__| | |_| | |>  <
 \_____|_| \_|\____/   \_____|\__,_|_/_/\_\

This script installs GNU Guix on your system

https://www.gnu.org/software/guix/
Press return to continue...
[1560068216.600]: Starting installation (dimanche 9 juin 2019, 10:16:56
(UTC+0200))
[1560068216.603]: [ PASS ] verification of required commands completed
[1560068216.754]: [ INFO ] init system is: systemd
[1560068216.759]: [ INFO ] system is x86_64-linux
[1560068218.725]: [ PASS ] Release for your system: guix-
binary-1.0.1.x86_64-linux
[1560068218.728]: [ INFO ] Downloading Guix release archive
guix-binary-1.0.1.x 100%[===================>]  59,66M  52,7KB/s    ds
16m 52s
guix-binary-1.0.1.x 100%[===================>]     833  --.-KB/s    ds
0s
[1560069232.509]: [ PASS ] download completed.
[1560069233.457]: [ PASS ] Signature is valid.
[1560069244.913]: [ PASS ] unpacked archive
[1560069244.915]: [ INFO ] Installing /var/guix and /gnu...
[1560069244.921]: [ INFO ] Linking the root user's profile
```

```
[1560069244.954]: [ PASS ] activated root profile at
/root/.config/guix/current
[1560069245.261]: [ PASS ] group <guixbuild> created
[1560069245.965]: [ PASS ] user added <guixbuilder01>
[1560069246.461]: [ PASS ] user added <guixbuilder02>
[1560069246.877]: [ PASS ] user added <guixbuilder03>
[1560069247.382]: [ PASS ] user added <guixbuilder04>
[1560069247.787]: [ PASS ] user added <guixbuilder05>
[1560069248.226]: [ PASS ] user added <guixbuilder06>
[1560069248.643]: [ PASS ] user added <guixbuilder07>
[1560069249.114]: [ PASS ] user added <guixbuilder08>
[1560069249.531]: [ PASS ] user added <guixbuilder09>
[1560069249.958]: [ PASS ] user added <guixbuilder10>
Created symlink /etc/systemd/system/multi-user.target.wants/guix-
daemon.service → /etc/systemd/system/guix-daemon.service.
[1560069250.602]: [ PASS ] enabled Guix daemon via systemd
[1560069250.604]: [ INFO ] making the guix command available to other
users
Permit downloading pre-built package binaries from the project's build
farms? (yes/no) y
[1560069259.347]: [ PASS ] Authorized public key for hydra.gnu.org
[1560069259.592]: [ PASS ] Authorized public key for ci.guix.gnu.org
[1560069259.595]: [ INFO ] cleaning up /tmp/guix.z0V
[1560069259.615]: [ PASS ] Guix has successfully been installed!
[1560069259.617]: [ INFO ] Run 'info guix' to read the manual.

root@debian:~#
```

Voilà, il ne reste plus qu'à commencer à explorer ce nouveau package manager installable sur toute distribution GNU/Linux.

Utilisation

```
# apt install info
```

```
# info guix
```

Pour vérifier que Guix a bien été installé, j'ai lancé :

```
# guix install hello
.....................................................................
......
Guix a installé les paquets nécessaires aux dépendances : guile-2.2.4,
texinfo-6.5, pkg-config-0.29.2, perl-5.28.0,  module-import-compiled,
mkfontdir-1.0.7, module-import, mkfontscale-1.2.1, libunistring-0.9.10,
freetype-2.9.1, libpng-1.6.37, libpng-1.6.34, libltdl-2.4.6, libgc-7.6.6,
```

```
libfontenc-1.1.4, coreutils-8.30, libffi-3.2.1, libcap-2.25, libatomic-
ops-7.6.6,
gzip-1.9, guile-gdbm-ffi-20120209.fa1d5b6, guile-2.0.14, gmp-6.1.2,
gdbm-1.18,
acl-2.2.52, config.scm et attr-2.4.47.
.....................................................................
......
1 paquet dans le profile
Il pourrait être nécessaire de définir les variables d'environnement
suivantes :

export PATH="/root/.guix-profile/bin${PATH:+:}$PATH"
```

```
root@debian:~# export PATH="/root/.guix-profile/bin${PATH:+:}$PATH"
```

Test d'installation d'un logiciel depuis le compte user :

```
$ guix install quassel
.....................................................................
......
Installation d'un nombre considérable de dépendances, 406,0 Mo seront
téléchargés
.....................................................................
......
1 paquet dans le profile
Il pourrait être nécessaire de définir les variables d'environnement
suivantes :
   export PATH="/home/hubert/.guix-profile/bin${PATH:+:}$PATH"
```

```
hubert@debian:~$ export PATH="/home/hubert/.guix-profile/bin${PATH:+:}$PATH"
```

```
hubert@debian:~$ quassel
```

quassel se lance sans problème

Tout a l'air de bien se passer pour l'instant, je prévois des mises à jour de cette page au fil des avancées...

Hum hum...

```
hubert@debian:~$ quassel

Command 'quassel' not found, but can be installed with:

apt install quassel
Please ask your administrator.
```

Bon alors là ?...

Un petit tour sur l'irc #guix m'apprend qu'il faut que je rajoute dans **le ~/.bashrc du compte user** la ligne suivante :

nom.sh

```
export PATH=$PATH:$HOME/.guix-profile/bin
```

Et ça y est, maintenant quassel se lance sans probème. Il m'a suffi, avec gedit, de copier-coller la ligne tout-en-haut du fichier et de l'enregistrer. Le problème qu'il y avait, c'était que la modification apportée initialement ne concernait que le ~/.bashrc du compte root.

From:
http://debian-facile.org/ - **Documentation - Wiki**

Permanent link:
http://debian-facile.org/utilisateurs:gonzoleri:tutos:installation-binaire-de-guix-sur-debian

Last update: **24/07/2019 05:23**