UTILISATION DE GUIX AU MÉSOCENTRE GLICID

GuixHPC

Atelier Guix en calcul intensif (HPC)

Jeudi 7 nov 2024

GLiCID

université angers · CENTRALE NANTES · Nantes Université · Le Mans Université

Yann Dupont <Yann.Dupont@univ-nantes.fr>

& GLiCID

since 2018

Merci de votre attention !

Des questions ? [1]

1. Oui, c'est une présentation rapide...

UN PEU DE DÉTAILS ET DE NUANCES

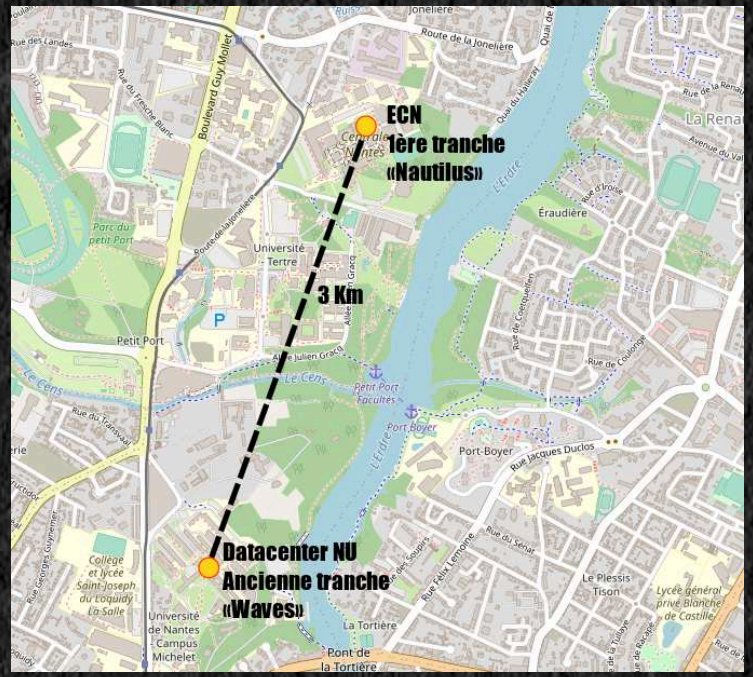
Principalement mise à jour d'éléments déjà présentés pendant :

- Café Guix 2023/2024
- Jcad 2023
- Workshop on Reproducible Software Environments for Research and HPC 2023

⚠ Désolé pour les redites

DACAS / GLICID : CALCULATEUR RÉGIONAL.

- 3 volets CPER DACAS :
  Datacentre (mi 2026),
  Réseau régional (06/2023)
  HPC (GLiCID), plusieurs clusters :
  - Nautilus (Tranche 1, AMD, 06/2023)
  - Phileas (Mesonet, Intel, été 2024)
  - Waves (existant, fin migration 2025)
  - Tranches 2 et 3 à venir (nouveau DC)



- À cheval sur 2 salles (jusqu'à mi 2026)
- Liaison unique 100 Gb/s depuis 11/2021, passe à 16x100 Gb/s à la fin du mois ! [1]

1. Raison du retard de migration de Waves : manque de bande passante pour le scratch SSD

GLICID TIME-MACHINE

Avant , Pays de la Loire

Centre	Utilisateurs actifs	CPU	GPU
CCIPL (UN)	155	5500	27
BiRD (Bio/UN)	142	450	9
ICI (ECN)	130	6300	18
MathStic (UA)	30	1000	10
INFRALAB (LMU)	50	1900	89

Guix était installé depuis 2018 au CCIPL sur le cluster Waves

Pourquoi ?

en 2024 les raisons restent identiques...

CCIPL : CLUSTER *HÉTÉROGÈNE*

376 nœuds, 7196 cœurs, 3 réseaux, **image unique**

Loire (ex-PBS)	Xeon Westmere	IB 40
Waves v1 (Slurm)	Broadwell, Skylake, GPU, Cascade Lake	OPA 100
Waves v2.a	Cascade Lake	RoCE 2x25/100
Waves v2.b	AMD Epyc 7282	RoCE 2x25/100

2024 : GLiCID : Nautilus (AMD), Phileas (Intel) en IB 100 + Waves : pas plus homogène.

UTILISATEURS & ÉQUIPE

+ de 300 utilisateurs :

- Majorité non experts, quelques profils «pointus».
- Tous types de sciences = tous types de programmes
  - «Classiques» opensource, fortement optimisés
  - Outils «maisons» ⟶ (outils de développement)
  - Binaires divers et … variés.
- ~ 1 ETP pour l'admin système, 0.5 pour les logiciels.

2024 : Fusion des équipes mais encore plus d'utilisateurs (> 500)

CALCULATEUR WAVES

Livré «clé en main» par l'intégrateur en 2016:

- Installation monolithique (2 serveurs / HA PCS)
- Déploiement par image (y compris frontale)
- Centos 7.2, Slurm 16.05, drivers OPA Intel
- Scratch BeeGFS
- Logiciels pré-installés (partage NFS)
  Ensemble de compilateurs GNU,Intel et PGI
  + librairies et logiciels scientifiques pré-compilés.

2024: Installation «clé en main» de Nautilus...

ET… ?

Clé en main = confort initial ! … de courte durée :-(

- Installation de nouveaux logiciels complexes
- Intégration des anciens nœuds
- Mise à jour annuelle délicate
  - Nœuds : Centos (7.2 ⟶ 7.3 … 7.9)
  - Serveurs : Inextricable
- Autonomie nécessaire (pas/plus d'intégrateur…)

⚠ Pas de tout repos !

💡 2024: Redhat 8.7 ⟶ 9 et sans trop de surprises… pas meilleur.

QUELQUES SOUCIS

- Upgrade glibc ⟶ Segfault de binaires générés par ICC (recompilation pas toujours possible)
- Incompatibilité de certains logiciels pré-installés

```
dupont-y@jaws ~]$ xmgrace
--> Broken or incomplete installation - read the FAQ!
```

- des logiciels de la racine (mal) packagés !

```
[dupont-y@jaws ~]$ emacs
emacs: error while loading shared libraries: libMagickWand.so.5: cannot open shared object file: No such file or directory
[dupont-y@jaws ~]$ ldapsearch
LDAP vendor version mismatch: library 20444, header 20440
```

SATISFAIRE L'UTILISATEUR

- Enrichir et redéployer l'image ? **NON !**

```
yum install gedit evince
Installation   2 Paquets (+37 Paquets en dépendance)
Is this ok [y/d/N]:
```

- Demande de versions et dépendances précises (Ocaml,R, Python, Julia)...
- Optimisations/tunings spécifiques :
  - Nœuds Intel Waves : AVX 512, Loire : SSE4.1
  - Nœuds AMD Waves : AVX2

ENVIRONMENT MODULES

Solution historique : compilations «à la main», mise à dispositions via modules.

Liens optionnels pour l'architecture créés au boot

Plusieurs versions concurrentes possibles

```
ls -al /opt/machine-dependant
lrwx … /opt/machine-dependant -> …/map/chezine003
```

⚠ Beaucoup de travail.

Dépendances très nombreuses.

Y compris à l'OS utilisé à la compilation.

💡 2024, liens multi-architecture non reconduits sur GLiCID

ALTERNATIVES

- Conteneurs : 
  - Problèmes de sécurité possibles (docker)
  - Confiance dans les images opaques récupérées (et les couches intermédiaires) (pas toujours à jour)
  - Accès aux GPU et réseaux OPA et IB pas toujours simple
  - Problèmes pour créer l'image (non root), recettes souvent non rejouables 6 mois + tard
- CONDA (ou ses avatars)
  - Utilisateur autonome , mais...
  - place occupée sur le /home...
  - Met le bazar (TM) dans les backups et les systèmes de fichiers distribués !!

Guix

- JCAD 2018, Ludovic présente Guix

Une façon de faire plus radicale ?

GUIX sur tous les nœuds ! ☺

GUIX SUR TOUS LES NŒUDS

Tous les nœuds et frontales depuis fin 2018 :

```
controller.10g:/var/guix    2793287680   1331191808   1462095872  48% /var/guix
controller.10g:/gnu         2793287680   1331191808   1462095872  48% /gnu
[root@cribbar012 ~]#
```

Source supplémentaire de logiciels,

Apporter de la reproductibilité (environnements…)

Faciliter la création de paquets «customisés»

2024 : capitalisation de l’expérience sur GLiCID

GUIX PRÉCHARGÉ POUR TOUS ?

⚠ pas totalement.

- Binaire guix [1] dans le path
- Le profil utilisateur n'est pas chargé par défaut sur les distributions RedHat (action volontaire)
  - Conflit possible avec des modules pré-chargés
    (certains utilisateurs en raffolent...)
- Autonomie utilisateurs ? ⟶ Formation nécessaire !
  - Des mécompréhensions peuvent diffuser parmi les utilisateurs

1. provient d'un utilisateur spécial : binaire mis à jour toutes les nuits et avec les canaux GLiCID et HPC...

INSTALLATION GLOBALE

- compte "guix-g@univ-nantes.fr" spécial
- guix pull toutes les nuits (depuis ce compte), avec tous les canaux (glicid...)
- Autres logiciels usuels dans ce profil
- link de ce profil vers /usr/local/bin (partagé sur tous les noeuds RedHat)

> ❗ Tous les utilisateurs accèdent à certains logiciels guix sans le savoir

STORE PARTAGÉ

- Partage NFS… pour l'instant [1]
- Montages valables sur distribution "foreign"
  Pas de Garbage Collect actuellement

```
guix-store.intra.glicid.fr:/var/log/guix   22G  654M   21G   4% /var/log/guix
guix-store.intra.glicid.fr:/var/guix       21G  550M   20G   3% /var/guix
guix-store.intra.glicid.fr:/gnu           1.1T  692G  361G  66% /gnu
```

- Compilation sur ce serveur dédié (GUIX_DAEMON_SOCKET="guix://guix-store.intra.glicid.fr")

1. Maquette Cephfs fonctionnelle, attente de passage en Ceph "reef ou squid", raison : robustesse du serveur de metadata

LE PROBLÈME DE GUIX PULL

```
[…]
Construction depuis ces canaux :
 guix     https://git.savannah.gnu.org/git/guix.git   dcca13e
 glicid   https://gitlab.univ-nantes.fr/glicid-public/guix-glicid.git acb78c3 1
 guix-sciencehttps://github.com/guix-science/guix-science.git  4c7ce20
```

1 : Des canaux locaux sont déclarés.

PROCESSUS COÛTEUX EN TEMPS

- Utilisateur : assimile à apt-get update
  - … fait beaucoup plus
  - … beaucoup plus long (surtout avec des canaux supplémentaires)
- Lenteur mal comprise
  - "Guix pull en début de TP… c'est 1H de perdue"

```
 lrwxrwxrwx 1 guix-g@univ-nantes.fr guix-g@univ-nantes.fr 21 Nov  4 22:07
 /var/guix/profiles/per-user/guix-g@univ-nantes.fr/current-guix  -> current-guix-258-link
[dupont-y@univ-nantes.fr@nautilus-devel-001 ~]$
```

- BUT: repousser le moment où un guix pull sera obligatoire pour l'utilisateur

PAQUETS SPÉCIFIQUES GLICID

Des paquetages spécifiques GLICID sont maintenus :

- gfortran > version standard (intéressant pour AMD EPYC)
- Adaptation à nos clusters
  - OpenMPI (support mises à jour libfabric, rdmacore, ucx... pour RoCE)
  - slurm (support sview, mysql, rest API)
  - softs antiques: xmgrace (avec motif)
  - librairies customisées
- Autres logiciels (podman, infrastructure)
  - et des services (slurm, munge, caddy, ldap) *Corrections de bugs
  - LVM2 (référence à systemld pendant l'activation footnote:5 [il restevisiblement des races conditions])
  - Ceph (patch pour apporter une compatibilité avec les nouvelels libbost)
- ... et des paquets obsoletes ou "cracra" :-)

https://gitlab.univ-nantes.fr/glicid-public/guix-glicid

DES PAQUETS DE RÉFÉRENCE

Les paquetages de Guix servent aussi de référence :

Lourd travail manuel de recompilations de GCC11 et OpenMPI (libfabric, rdamcore, ucx) en modules 😨

Idem benchmarks :

```
(define-public glicid-specific-intel-mpi-benchmarks
(package
(inherit glicid-intel-mpi-benchmarks)
(name "glicid-specific-intel-mpi-benchmarks")
(inputs `(("openmpi", glicid-specific-openmpi-gcc-11)
,@(package-inputs glicid-intel-mpi-benchmarks)))
)
)
```

GLICID ET ARRIVÉE DE NAUTILUS

- Déploiement "clé en main" de Nautilus : 2nd semestre 2023
  - À l'École Centrale de Nantes
  - À la place de l'ancien cluster "Liger" d'ICI
  - **NE REMPLACE PAS LIGER** (Calculateur régional)
- Waves (v1) continue sa vie et migrera plus tard
- plusieurs semaines d'arrêt : Il faut gérer les anciens utilisateur de Liger !

> Pour gérer la transition, Un cluster "Alpha", virtuel et pur Guix , a été déployé.

- Peu/Pas de logiciels disponibles et optimisés pour le nouveau cluster au démarrage
  - Systeme RedHat 8, Waves(v1) et Liger étaient RedHat 7
- Guix était disponible dès le jour 1

BILAN CÔTÉ UTILISATEUR

- Bien adapté au HPC
  - Façon élégante de sortir de «l'enfer des dépendances»
  - Création et suivi par l'équipe de paquets spécifiques au cluster
  - Autonomie des utilisateurs pour gérer leurs logiciels (dépendances et versions)
- Utilisation croissante

Guix est loin de n'être qu'un gestionnaire de paquets !

TAUX D'UTILISATION

```
bash-5.1# find /var/guix/profiles/per-user -name 'guix-profile-*' | grep -v guix-profile-1-link | awk -F '/' '{ print $6 }' | uniq -c | sort -n | wc
    110     220    3453
```

- 608 users au 6/11/2024
- 138 utilisateurs ont lancé guix install
- 110 ont fait plus de 1 guix install
- 6 utilisateurs ont dépassés 100 profiles
- 35 ont plus de 20 profiles

POURQUOI GÉRER L'INFRASTRUCTURE À PART ?

CPER : 3 tranches d'investissements

- 3 tranches = 3 marchés
  - Compatibilité des matériels et des solutions logicielles non assurée
- 1ère tranche contrainte par le temps
  - Cluster autonome environnementé, destiné à s'insérer dans la nouvelle infrastructure

> ⚠ Dès 2022 : choix de neutralité, d'indépendance et de contrôle :
> Nouvelle infrastructure gérée totalement en interne

NOUVELLE INFRASTRUCTURE À DÉMARRER

- Distribuée, mutualisée, **redondée**
- Déploiement en parallèle avec Waves de :
  - Réseau (*Fabric, Open Networking*)
  - Stockage Ceph (NVME **ET** volumétrique (plusieurs Po))
  - Services
    - Gestion de la haute disponibilité
    - Gestion d'identité
    - Slurm côté Waves (plusieurs clusters séparés fonctionnant de concert)
    - [...]
- Simple à redéployer en cas de soucis
  - Faible adhérence aux solutions déployées par les constructeurs

Choix infrastructure virtuelle KVM (→ PROXMOX) ... et Guix

GUIX, LE COUTEAU SUISSE DU PLOMBIER NUMÉRIQUE

- guix --help
  - guix pull
  - guix package
  - guix time-machine
    [...]
  - guix system
    [...]
    - guix system image («construit une image du Système Guix»)
    - docker-image
      [...]

CONTENU DU CANAL GLICID

```
$ guix pull
[…]
Construction depuis ces canaux :
 guix    https://git.savannah.gnu.org/git/guix.git   dcca13e
 glicid  https://gitlab.univ-nantes.fr/glicid-public/guix-glicid.git acb78c3 1
```

1 Canal «GLiCID» utilisé en ajout du canal officiel

- Ajoute paquets logiciels et services :
  - Inexistants (service slurmctld, packages scientifiques)
  - Versions plus récentes ou incompatibles (libfabric pour RoCE)
  - Dérivations spécifiques adaptées à GLiCID (slurm avec openpmix v3 & openmpi-glicid)
  - Suivent l'évolution de la branche principale (qemu-with-rbd)
  - Utilisation directement possible dans nos VM

```
$ guix package -A | grep glicid
slurm-glicid      22.05.9          out              glicid/packages/parallel.scm:159:2
[…]
qemu-with-rbd     8.1.0            out,static,doc   glicid/packages/virtualization.scm:17:2
```

GUIX SYSTEM IMAGE

```
$ guix system image --help
Utilisation : guix system [OPTION ...] ACTION [ARG ...] [FICHIER]
Compiler le système d'exploitation déclaré dans FICHER en suivant ACTION.
Certaines ACTIONS prennent en charge des ARGUMENTS supplémentaires.
[...]
Les valeurs possibles pour ACTION sont :
  image         construit une image du Système Guix
[...]
```

- Fichier à fournir : **programme**[1] guile, renvoie un objet du type attendu par l'action.
- guix system image : attend objet «operating-system»

1. OUI, un **programme**, pas une simple définition

GUILE, SCHEME

```
(fonction arg1 arg2 (fonction-qui-renvoie-arg3 arg1-de-celle-là))
```

- Tout se programme en Guile (Langage fonctionnel, dialecte de Scheme, famille du Lisp) : définitions de paquets, services et systèmes
- Connaissance pas strictement nécessaire au début (copier/coller de définitions)
- Déclarer des systèmes complexes ⟶ écriture + complexe ⟶ connaissance plus approfondie de Guile
- Documentation officielle, cookbook, communauté : listes, GuixHPC, CaféGuix

⚠ Syntaxe riche en «(» et «)», on aime ou pas ...

OPERATING-SYSTEM : DOCUMENTATION

12.2 Référence de `operating-system`

Cette section résume toutes les options disponibles dans les déclarations `operating-system` (voir Utiliser le système de configuration).

Type de données : `operating-system`

C'est le type de données représentant une configuration d'un système d'exploitation. On veut dire par là toute la configuration globale du système, mais pas la configuration par utilisateur (voir Utiliser le système de configuration).

`kernel` (par défaut : `linux-libre`)

L'objet du paquet du système d'exploitation à utiliser[28].

`hurd` (par défaut : `#f`)

L'objet du paquet du hurd à être lancé par le noyau. Lorsque ce champ est défini, produire un système d'exploitation GNU/Hurd. Dans ce cas, `kernel` doit également être défini sur le paquet `gnumach` — le micro-noyau sur lequel tourne le Hurd.

> **Attention :** Cette fonction est expérimentale et seulement prise en charge pour les images de disques.

`kernel-loadable-modules` (par défaut : '())

Une liste d'objets (généralement des paquets) pour collecter les modules de noyau chargeables depuis - par exemple `(liste ddcci-driver-linux)`.

`server-arguments` (par défaut : `%default-kernel-arguments`)

Liste de chaînes ou de gexps représentant des arguments supplémentaires à passer sur la ligne de commande du noyau — p. ex. `("console=ttyS0")`.

`bootloader`

L'objet de configuration du chargeur d'amorçage. Voir Configuration du chargeur d'amorçage.

- Record guix, nombreux champs, mais nombreuses valeurs par défaut

DÉFINITION MINIMALE DE VM

```
(use-modules (gnu))

(operating-system
 (host-name "mini-1") 1
 (bootloader (bootloader-configuration 1
          (bootloader grub-bootloader)
          (targets '("/dev/sda"))))
 (file-systems (cons (file-system 1
                (device (file-system-label "my-root"))
                (mount-point "/")
                (type "ext4")) %base-file-systems))
 (kernel-arguments (list "console=tty0 console=ttyS0,115200"))) 2
```

1. 3 champs strictement nécessaires, tous les autres sont par défaut
2. Optionnel, pour lancement qemu en mode texte

```
$ guix system image simple-1.scm -r virtsimple1.img
$ qemu-system-x86_64 -enable-kvm -nographic -m 4G virtsimple1.img

GRUB loading.
[    0.000000] Linux version 6.4.16-gnu (guix@guix) (gcc (GCC) 11.3.0, GNU ld (GNU Binutils) 2.38) #1 SMP PREEMPT_DYNAMIC 1
[    0.000000] Command line: BOOT_IMAGE=/gnu/store/qhynq8jfskirrn7fj5965ajmrs7zfshc-linux-libre-6.4.16/bzImage root=38af4c98-6715-004c-d4b2-17f038af4c90
[    0.000000] KERNEL supported cpus:
[    0.000000]   Intel GenuineIntel
[...]
This is the GNU system.  Welcome.
mini-1 login: root
This is the GNU operating system, welcome!
root@mini-1 ~#
```

🎉 ÇA FONCTIONNE 🎉

- **Génération** d'un système d'exploitatiton complet
  - Définition de moins de 15 lignes, sans rien de plus
  - Sans télécharger un média d'installation
  - À jour (par exemple kernel 6.4.16) [1]
  - Pas besoin de customiser après coup (ex: ansible)
- La définition est minimale, mais la VM ne l'est pas
  - Des packages de base ne sont pas nécessaires dans le cadre de VM.

1. au moment où ces slides ont été rédigés...

«YAKA FAUKON»

- Ôter le superflu, ajouter des packages & services : ajouter et modifier les entrées des listes %base-packages et %base-services
- Factoriser les fonctions et configurations entre OS : créer des templates pour écriture simple module guile (glicid template v3)
- Système plus évolué pour lancer les VM :
  - libvirt/virt-manager
  - Proxmox

TEMPLATE GLICID

```
(define-public %ccipl-net-v4-cluster "10.141.0.0/16") 1
(define-public %glicid-net-gateway "10.50.255.254")
(define-public %glicid-net-gateway "10.141.255.252")
(define-public %glicid-dmznet-gateway "xx.yy.zz.1")
(define-public %glicid-base-services 2
        (append (list
                glicid-default-ssh-services
                glicid-default-ntp-services
[...]
(define-public %glicid-one-disk-vm-os 3
 (operating-system
[...]
  (packages %glicid-base-packages)
  (services %glicid-base-services)))
```

1. Nombreuses définitions de réseaux, gateway, name servers...
2. Différentes listes de services (également des listes de paquetages), configs spécifiques
3. Des définitions d'operating system paramétrées et prêtes à l'(inherit)

VM «DEBUG» COMPLÈTE GLICID

```
(use-modules (glicid template v3) (gnu services networking))

(define test001-ip (list (network-address (device "eth0") (value "10.50.103.201/16"))))

(define custom-net 1
 (service static-networking-service-type
       (list (static-networking (addresses test001-ip)
                        (routes  %glicid-testnet-default-routes)
                        (name-servers %glicid-testnet-name-servers)))))


(define %base-os %glicid-one-disk-debug-os) 2

(define %inherited-services (operating-system-user-services %base-os))

(operating-system
 (inherit %base-os)
 (host-name "test001")
 (services (append (list custom-net) %inherited-services))) 1
```

1. Service static-networking propre à chaque instance de VM
2. Variante «debug» du template GLiCID : la plus riche en options

INCLUS:

- Réglages réseau, DNS, NTP, SSH+clés des admin, Syslog, Agents Zabbix et Qemu, Canaux Guix...
- Variante debug ajoute : réglages NSS LDAP, NSCD, configs par GIT, éditeurs et nombreux outils de debug...

CONSTRUCTION VM

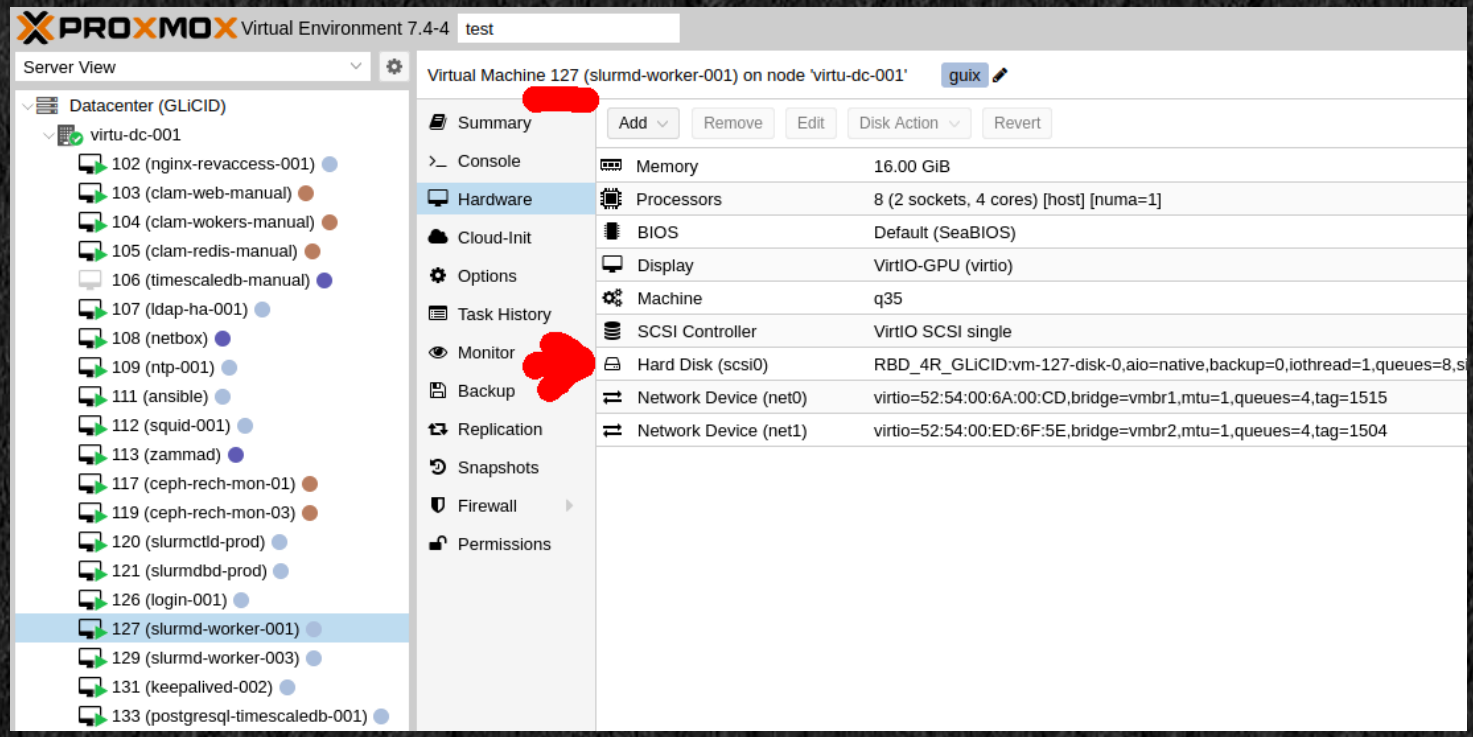
```
for-slides/test001$ ./build.sh
RBD_4R_GLiCID/VMroot_for-slides-test001_202309242151 2

substitute: updating substitutes from 'https://guix-substitutes.glicid.fr'... 100.0%
substitute: updating substitutes from 'https://ci.guix.gnu.org'... 100.0%
substitute: updating substitutes from 'https://bordeaux.guix.gnu.org'... 100.0%
0.4 MB will be downloaded
[…]
The following derivations will be built:
/gnu/store/36qi998hw15s02ipa7czlcs2iry46cfn-disk-image.drv
/gnu/store/ara6gi64cdm2hd8jvb4gg7mxy4p4ziwj-genimage.cfg.drv
/gnu/store/qc5a0bvfdhz2nqhy48j6qvxfhlzrb6rq-partition.img.drv
[…]
building /gnu/store/ara6gi64cdm2hd8jvb4gg7mxy4p4ziwj-genimage.cfg.drv...
building /gnu/store/36qi998hw15s02ipa7czlcs2iry46cfn-disk-image.drv...
/gnu/store/b34n9k0hcwaanacgf2g1zp2vnxjj4yim-disk-image 1
dd to ceph, please wait, will take time 2

real  1m5.389s 3
```

1. Image créée dans le store GUIX local
2. Copiée dans le pool Ceph RBD_4R_GLiCID (nom horodaté)
3. Clone de l'image (VMroot_for-slides-test001_candidate)

Un script permet de renommer ce candidat au nom utilisé par KVM ou proxmox (vm-233-disk-0).

Images utilisables partout où CEPH est disponible

DÉPLOYER SUR PROXMOX



- VM Guix
- VM Debian
- VM Centos

- Disques identifiés ainsi : vm-xxx-disk-yyy, ici vm-127-disk-0
- 0 script de déploiement… juste un changement d'image

ET C'EST PARTI !

(Certains services sont en haute disponibilité)

- répartiteurs de charge croisés (keepalived) (4 Intra, 4 Pub)
- DNS
- LDAP
- bastions SSH
- serveurs NFS «bloc» et en cluster
- miroirs, proxies, reverses proxies, serveurs WWW
- serveur zabbix
- BD : mysql, postgresql + timescaledb
- slurmctld, slurmdbd
- machines login et devel (client slurm)
- pseudo nœuds de calculs virtuels (client slurm)

06/11/2024 : 94 VM déployées ainsi

Dans certains cas, création de nouveaux services (ldap, slurmctld...)

MACHINES IMMUABLES

- La configuration est **généralement** embarquée dans /gnu/store

```
…4mrz9wlf-chrony-4.3/sbin/chronyd -d -f /gnu/store/n1…1f5bq7c7wc-chrony.conf
…4gy9y9bx-rsyslog-8.2212.0/sbin/rsyslogd -n -f /gnu/store/qx…7x-rsyslog.conf
```

- /gnu/store est en lecture-seule
- Machines immuables **ET** volatiles : mettre à jour = regénérer la machine !
- Déploiement facile et reproductible en quelques secondes
  - Génération ⟶ VM OFF ⟶ changement image ⟶ VM ON
  - VM sans état ; certaines détiennent de la donnée persistante.
    - Utilisation des templates %glicid-two-disks… : volumes LVM2 persistants

Snapshot du volume persistant pour déploiement de nouvelles versions

MACHINES À CONFIG GIT

- Certains configurations sont dynamiques
  - White-list des proxies
  - Configuration des services keepalived
- Regénérer des VM est une perte de temps
- Nous fabriquons des VM qui "piochent" leur config dans gitlab (manuellement ou toutes les 5 minutes)
  - Grace a des fonctions d'enrichissement d'operating system

```
(define parameters-file (canonicalize-path "../common/localconf/parameters"))  ;; need this to have absolute path


(define base-os (make-glicid-clusternode-keepalived-os "/etc/keepalived/keepalived-onlyglicid.txt" parameters-file)) 1
(define inherited-services (operating-system-user-services base-os))
(define custom-motd (extra-special-file "/etc/motd" %generated-local-motd))

(operating-system
 (inherit base-os)
 (host-name %generated-local-host-name)
 (services
  (append (list custom-local-static-networking-service-type custom-motd) inherited-services)))
```

1 Ici on construit l'OS avec des paramètres (dont les dépots+token GITLAB)

EXEMPLE MACHINES KEEPALIVED (2)

```
(define base-os (make-osvariant-gitconfig (make-osvariant-ipvs %glicid-one-disk-vm-os))) 1

(define (make-keepalived-service-type local-config-file)
 (service keepalived-service-type
        (keepalived-configuration (config-file local-config-file)
                                  (keepalived keepalived-glicid))))
```

1. Idée : partir d'OS simplifié à l'extrême et enrichi par appels de fonctions d'enrichissement successives

EXEMPLE MACHINES KEEPALIVED (3)

```
(define (make-glicid-clusternode-services config-file update-conf-path) 2
 (append (list (make-keepalived-service-type config-file)
               (extra-special-file "/etc/localconf/parameters" (local-file update-conf-path))
                clusternode-iptables-service)
               (operating-system-user-services base-os)))

(define clusternode-keepalived-packages (append (list ipvsadm tcpdump conntrack-tools) (operating-system-packages base-os)))


(define-public (make-glicid-clusternode-keepalived-os root-config-file update-conf-path) 1
  (make-osvariant-nodaemonsocket (operating-system (inherit base-os)
              (packages clusternode-keepalived-packages)
              (host-name "undefined-keepalived-nodes-default-name")
              (services (make-glicid-clusternode-services root-config-file update-conf-path)))))
```

1. Encore une fonction qui transforme un OS
2. Et ici une fonction qui transforme des services

RÉACTIVITÉ ET CONTRÔLE

Début septembre 2023 :

NIST

NVD MENU

Information Technology Laboratory

NATIONAL VULNERABILITY DATABASE

NIST NATIONAL VULNERABILITY DATABASE NVD

VULNERABILITIES

CVE-2023-41915 Detail

Description

OpenPMIx PMIx before 4.2.6 and 5.0.x before 5.0.1 allows attackers to obtain ownership of arbitrary files via a race condition during execution of library code with UID 0.

Severity

CVSS Version 3.x | CVSS Version 2.0

CVSS 3.x Severity and Metrics:

NIST: NVD **Base Score:** 8.1 HIGH **Vector:** CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:U/C:H/I:H/A:H

NVD Analysts use publicly available information to associate vector strings and CVSS scores. We also display any CVSS information provided within the CVE List from the CNA.

Note: NVD Analysts have published a CVSS score for this CVE based on publicly available information at the time of analysis. The CNA has not provided a score within the CVE List.

QUICK INFO

CVE Dictionary Entry:
CVE-2023-41915
NVD Published Date:
09/09/2023
NVD Last Modified:
09/13/2023
Source:
MITRE

MISE À JOUR DU PAQUET OPENPMIX

```
+(define-public openpmix-3.2.5
+  (package
+    (inherit openpmix-3.2.4) 1
+    (version "3.2.5") 2
+    (source (origin
+              (method url-fetch)
+              (uri (string-append
+                    "https://github.com/openpmix/openpmix/releases/download/v" version "/pmix-" version ".tar.bz2")) 2
+              (sha256
+               (base32
+                "13cc11wxf00w485h6pxjcpwziihaix1pj9rrd20cis1i4bi2hrfv")))))) 3

 (define-public openpmix-4.1.0
   (package
@@ -95,7 +111,7 @@

 (define-public openpmix-3
-  openpmix-3.2.4)
+  openpmix-3.2.5) 2
```

1. Hériter d'une version proche
2. Changer la version et la définir comme nouvelle version stable
3. Changer le checksum de la source du paquet.

Actualise : slurm-glicid, openmpi-glicid... et les nombreux paquets qui en dépendent

Les nouvelles VM déployées sont non vulnérables

GUIX VS ANSIBLE/PUPPET/CHEF/SALT

L'un n'empêche pas les autres…

- **Générer** un système complet et totalement configuré
- Écriture dans un langage de programmation, intelligence & liberté possibles
- guix deploy : remplit les besoins de déploiements massifs

vs

- Fichiers de configuration spécifiques au système utilisé
- **Modifier et reconfigurer** un système pré-installé (comment/par qui ?)

Aspect «immuable» perdu.

ARCHITECTURES EXOTIQUES

```
guix system --list-targets
The available targets are:

  - aarch64-linux-gnu [1]
  - arm-linux-gnueabihf
  - i586-pc-gnu
  - i686-linux-gnu
  - i686-w64-mingw32
  - mips64el-linux-gnu
  - powerpc-linux-gnu
  - powerpc64le-linux-gnu
  - riscv64-linux-gnu [1]
  - x86_64-linux-gnu
  - x86_64-w64-mingw32
```

[1] Intéressant pour la prospective

```
$ guix system image virtrv.scm --target=riscv64-linux-gnu
```

- ... Fonctionne directement ! (testé sur qemu et machines physiques ARM et Risc-V)
- Limitations pour certains paquets (non cross-compilables ou architecture non supportée)

BILAN : BÉNÉFICES

- Composition de systèmes d'exploitation
  - (inherit base-os) = troncs communs, risques d'erreurs ou d'omissions minimisés
  - Effort payant : capitalisation, peu de réécritures constatées en presque 3 ans, temps gagné *in fine*
- Contrôle et chaîne de confiance qui vont très loin
  - Boostrap minimal, sources contrôlées, binaires reproductibles
  - Contrôle fin de ce qui est installé, « durcissement » noyau possible
  - Logiciels spécifiques et dépendances intégrés de façon cohérente partout
- Constantes (réseaux, etc) faciles à modifier : redéploiements massifs aisés (GLiCID test/alpha/beta)
- Machines jetables redéployables à l'envie
  - Pas de configuration après coup
  - Reproductibilité système ! (time-machine est utilisable)
    - Retour arrière simple (attention : snapshots pour les volumes de VM avec état)
    - Les définitions de machines (et templates GUIX) sont dans GIT

BILAN : INCONVÉNIENTS

- Guile requiert un apprentissage certain
  - Appropriation différente selon les membres de l'équipe
  - « Tout le monde fait autrement »
- Si service ou package non porté :
  - S'y confronter : parfois compliqué
  - Y passer du temps : ressource rare…
    - Certains packages ou services demandent trop d'efforts : effort **GLOBAL** nécessaire
    - Solution de facilité : déployer du « tout fait » temporairement
- Ne préserve pas des bugs (mise à jour de paquetages…)
  - Attention à l'excès de confiance et au redéploiement sans vérification
- «Bus factor»
  - 3 membres de l'équipe génèrent régulièrement des packages et des VM.

MERCI DE VOTRE ATTENTION

Guix & GLiCID since 2018

Merci de votre attention !

Des questions ? [1]

1. Oui, c'est **vraiment** fini