Guix-GENERATED HPC CENTER INFRASTRUCTURE

GuixHPC

First Workshop on Reproducible Software Environments
Montpellier, 2023/11/09

GLiCID

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

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

DISCLAIMER: WE WON’T TALK ABOUT SCIENCE

… Plumbing[1] session.
(infrastructures[2].)

- Making this activity more …
- fun ? noble ? reliable ? reproducible ?

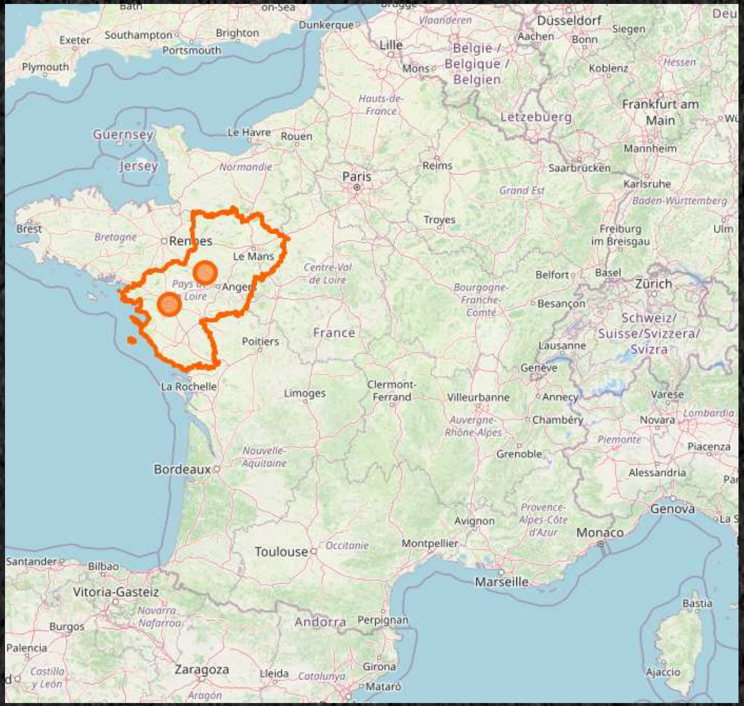
[1] I am infrastructures engineer / digital plumber at GLiCID.

[2] gaz plant : french way to speak of arcane systems .

GLICID : HPC "PAYS DE LA LOIRE"

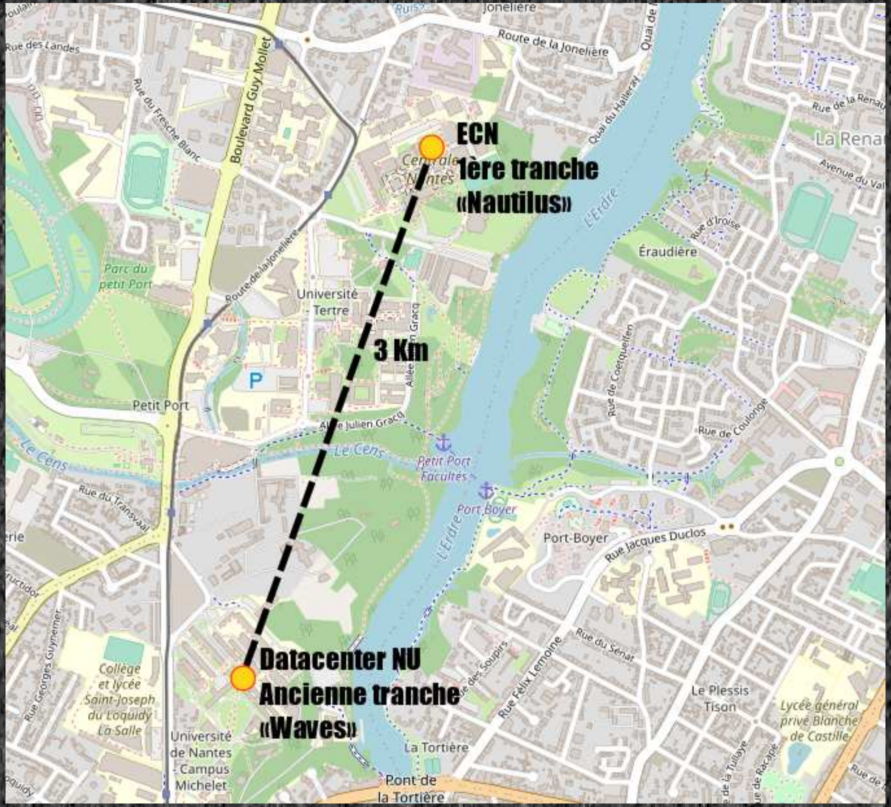
HPC center for >300 regional researchers
Hardware located in Nantes

- 3 clusters : (managed as one) (SLURM_CLUSTERS="all")
  - Waves (Historical) :
    6712 cores (all kind)
    50 GPU (all kind)
    OPA and RoCE, very heterogeneous
  - Nautilus (New) :
    5376 cores AMD Genoa
    16 GPU A100
    IB 100
  - Phileas (mesonet, to come !) :
    3072 cores Intel Sapphire rapids
    IB 100



DACAS / GLICID

- 3 parts CPER DACAS :
  Datacentre (2025 or 2026),
  Regional network (2023-06)
  HPC (GLiCID), different clusters :
  - Nautilus (1st Slice, 2023-06)
  - Old "Waves" will migrate end 2023
  - Slices 2 et 3 of HPC cluster to come (installation in new datacentre)



CPER is a French three part contract
Contrat Plan-État-Région
(means ... money allocation)

- Installation split between rooms (until 2025/2026)
- Single link 100 Gb/s since 11/2021

3 SLICES

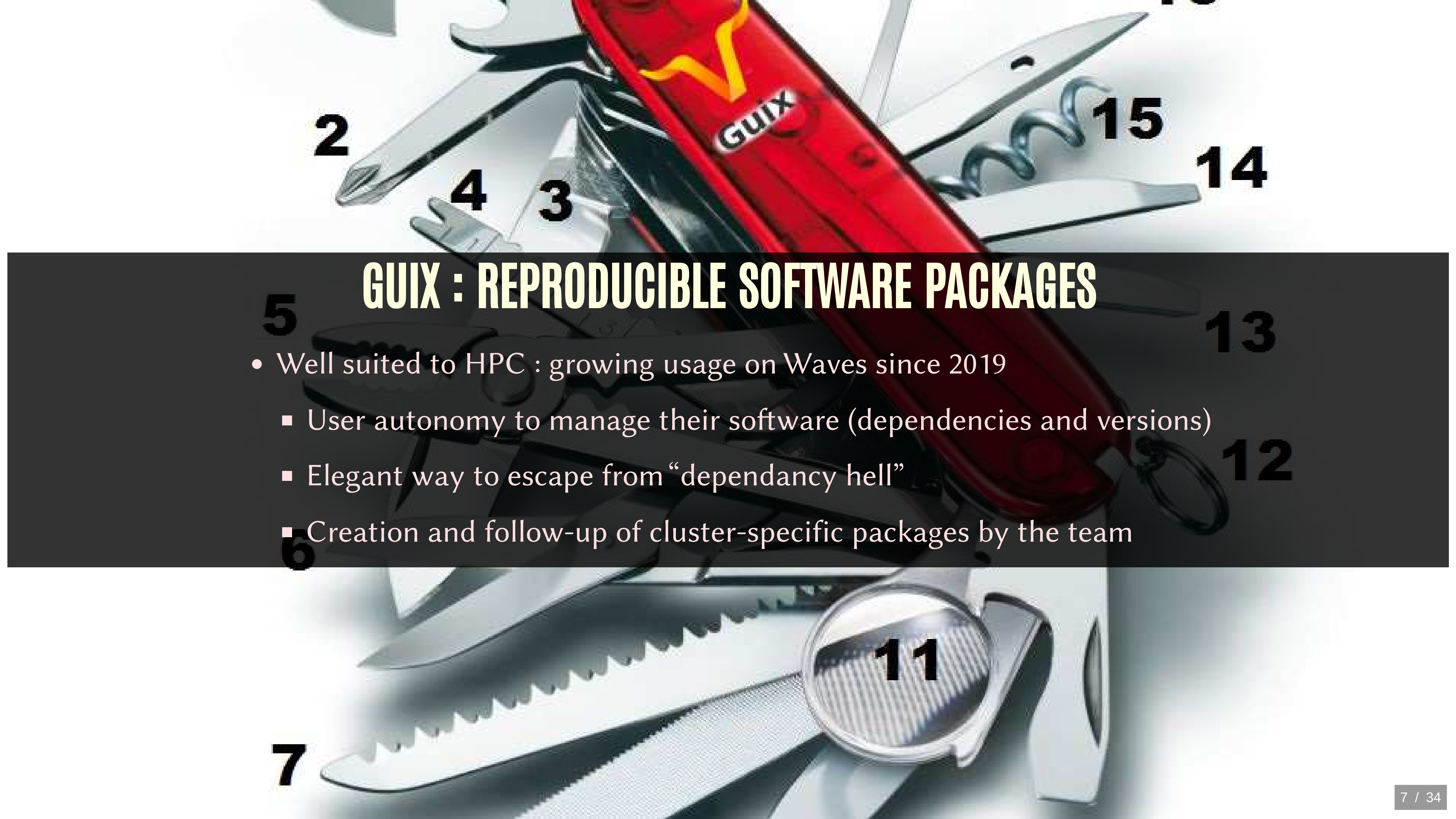
- 3 slices = 3 public procurements = differents solution providers
  - Compatibility of hardware and software solutions not guaranteed
- 1st slice constrained by time
  - Autonomous cluster (with proprietary software...)
    intended to be inserted as soon as possible into the new infrastructure

> ⚠ From 2022: choice of neutrality, independence and control:
>
> New infrastructure managed entirely in-house

NEW INFRASTRUCTURE TO START

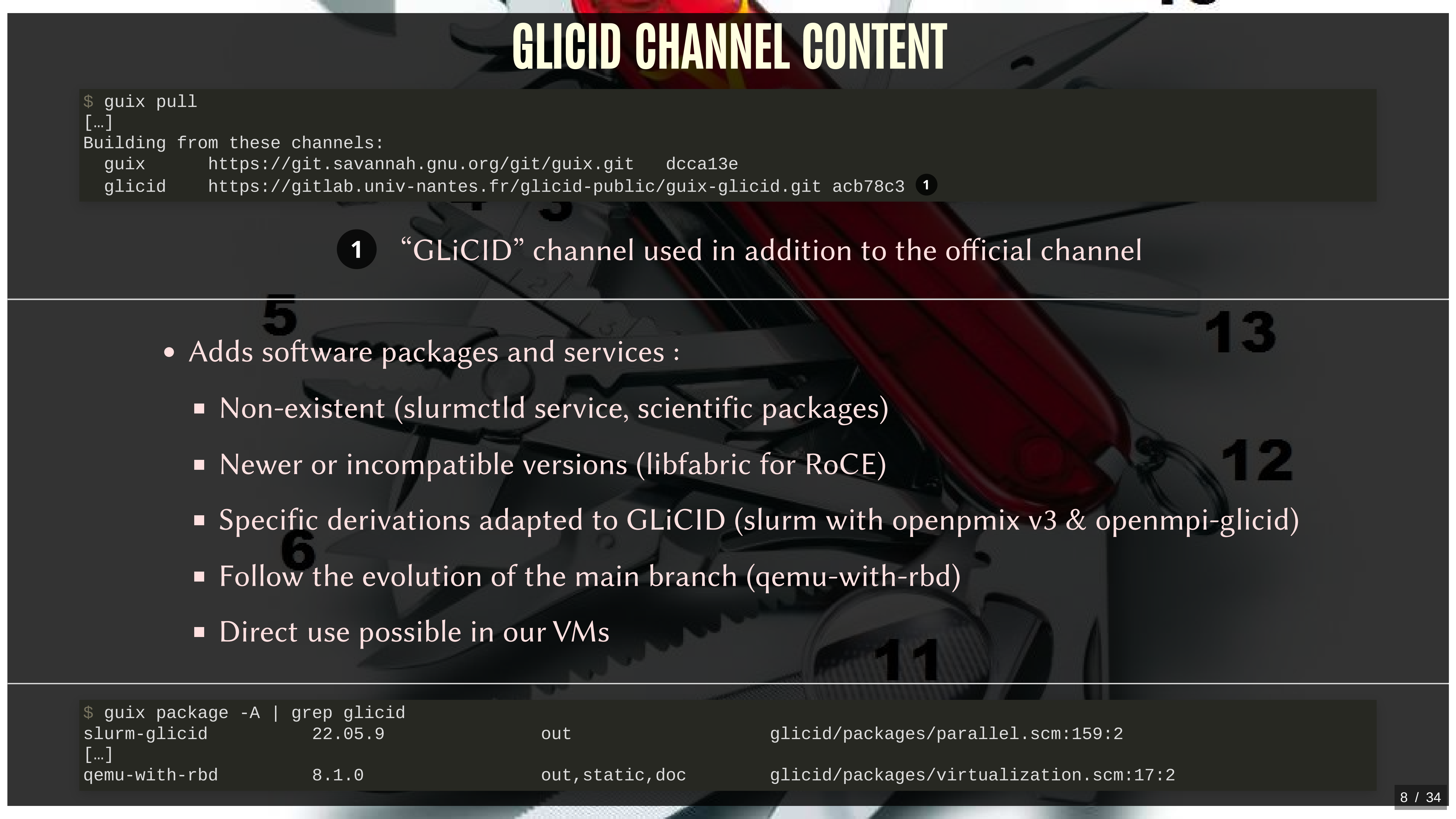
- Distributed, shared, **redundant**
- Parallel deployment with "Waves" of :
  - Network (*Fabric, Open Networking*)
  - Ceph storage (NVME **AND** volumetric (multiple PB))
  - Services
    - High availability management
    - Identity management
    - Slurm on the "Waves" side (several separate clusters working in concert)
    - [...]
- Simple to redeploy in case of problems
  - Low adherence to the solutions deployed by manufacturers

Choice : KVM virtual infrastructure (→ PROXMOX) and Guix



GUIX : REPRODUCIBLE SOFTWARE PACKAGES

- Well suited to HPC : growing usage on Waves since 2019
  - User autonomy to manage their software (dependencies and versions)
  - Elegant way to escape from “dependancy hell”
  - Creation and follow-up of cluster-specific packages by the team



GLICID CHANNEL CONTENT

```
$ guix pull
[…]
Building from these channels:
  guix      https://git.savannah.gnu.org/git/guix.git   dcca13e
  glicid    https://gitlab.univ-nantes.fr/glicid-public/guix-glicid.git acb78c3 ①
```

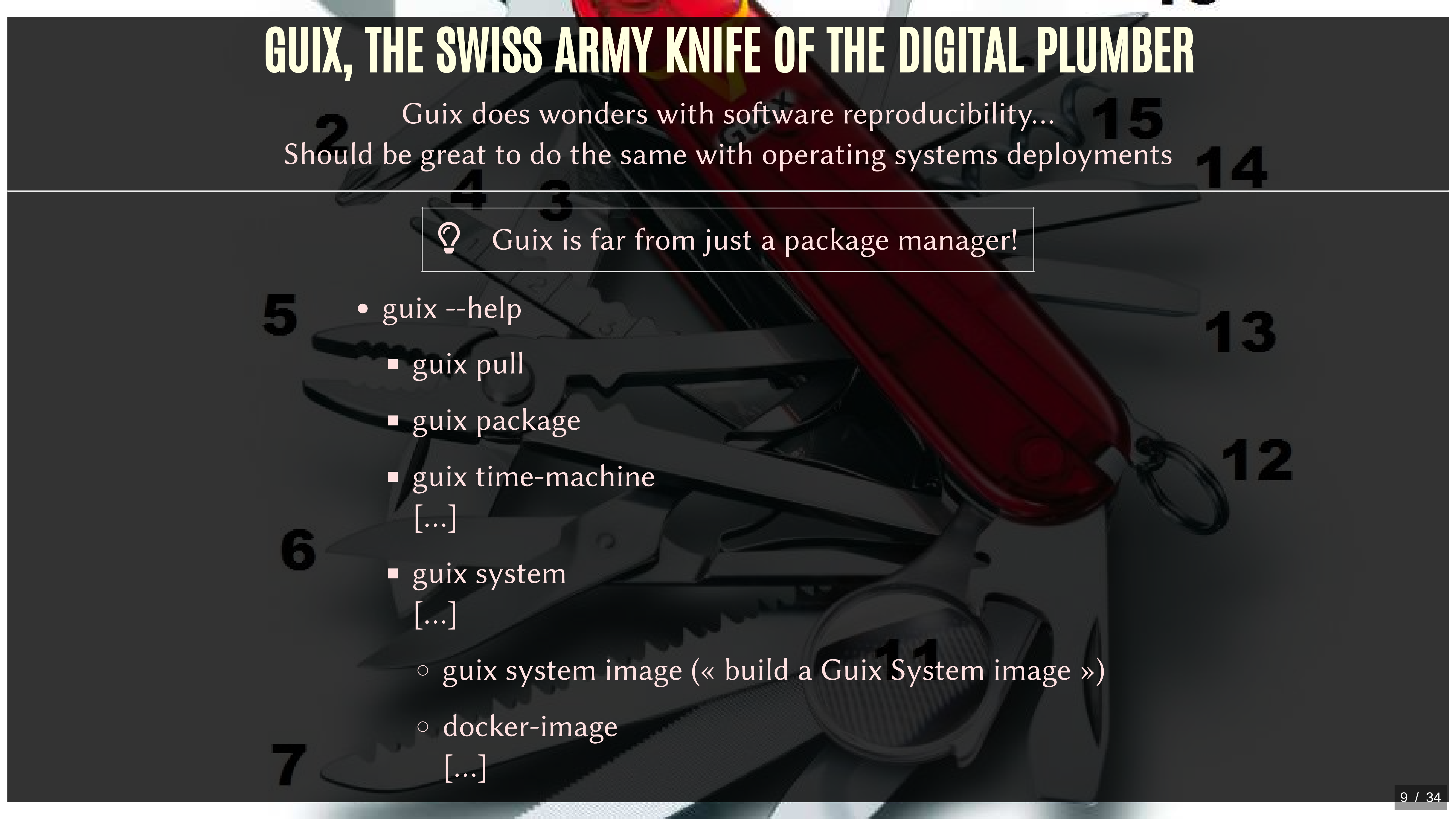
① “GLiCID” channel used in addition to the official channel

- Adds software packages and services :
  - Non-existent (slurmctld service, scientific packages)
  - Newer or incompatible versions (libfabric for RoCE)
  - Specific derivations adapted to GLiCID (slurm with openpmix v3 & openmpi-glicid)
  - Follow the evolution of the main branch (qemu-with-rbd)
  - Direct use possible in our VMs

```
$ 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, THE SWISS ARMY KNIFE OF THE DIGITAL PLUMBER

Guix does wonders with software reproducibility…
Should be great to do the same with operating systems deployments



💡 Guix is far from just a package manager!

- guix --help
  - guix pull
  - guix package
  - guix time-machine
    […]
  - guix system
    […]
    - guix system image (« build a Guix System image »)
    - docker-image
      […]

GUIX SYSTEM IMAGE

```
Usage: guix system [OPTION ...] ACTION [ARG ...] [FILE]
Build the operating system declared in FILE according to ACTION.
Some ACTIONS support additional ARGS.

The valid values for ACTION are:

[...]
 image            build a Guix System image
[...]
```

- File to provide: Guile **program**[1], returns an object of the type expected by the action.
- guix system image: expects “operating-system” object

[1] YES, a **program**, not a simple definition

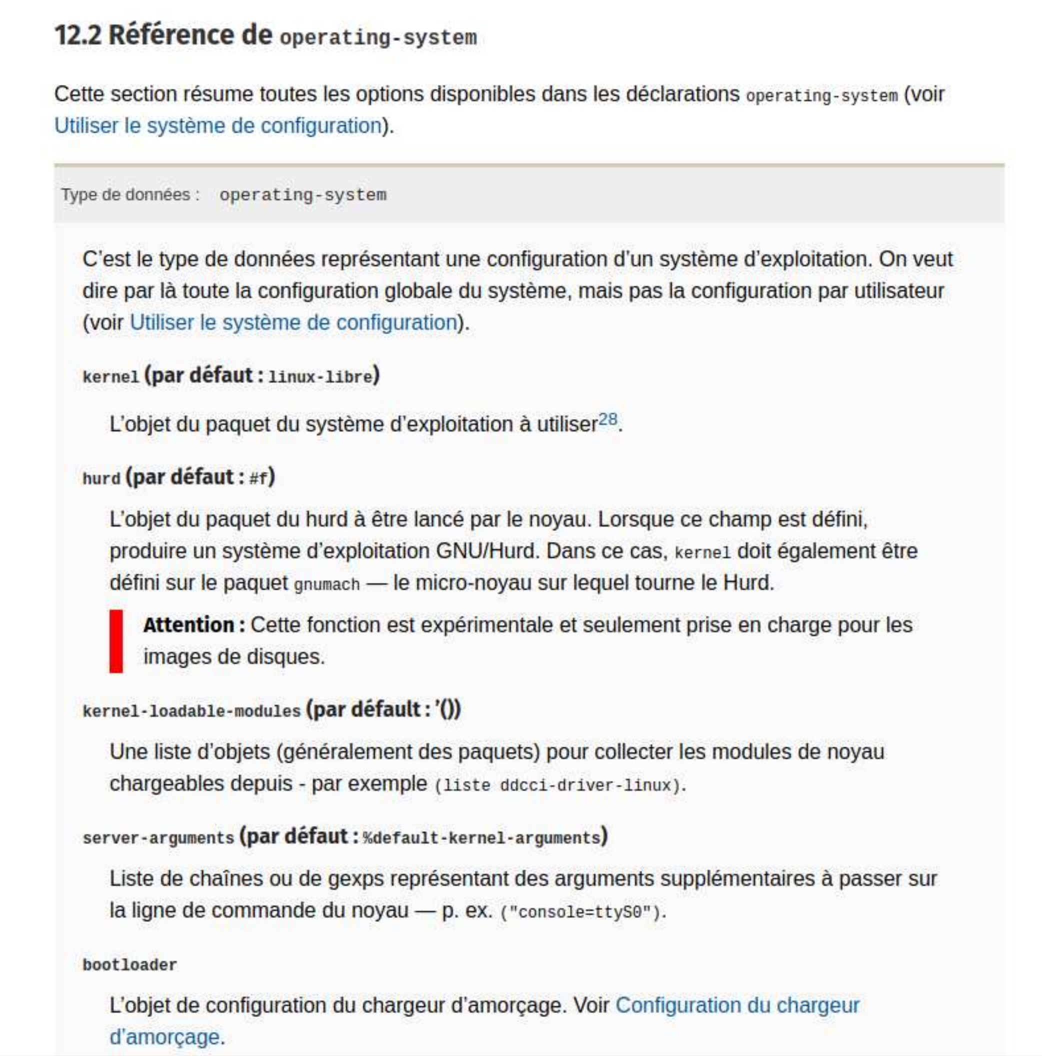
GUILE, SCHEME

```
(function arg1 arg2 (function-that-returns-arg3 arg1-of-this-one))
```

- Everything is programmed in Guile (functional language, Scheme dialect, Lisp family): definitions of packages, services and systems
- Knowledge not strictly necessary at the beginning (copy/paste of definitions)
- Declare complex systems ⟶ more complex writing ⟶ deeper knowledge of Guile
- Official documentation, cookbook, community: lists, GuixHPC, CaféGuix

⚠ Syntax rich in "(" and ")", like it or not...

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, many fields, but many default values

MINIMUM VM DEFINITION

```
(use-modules (gnu))

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

❶ 3 strictly necessary fields, all others are by default

❷ Optional, for launching qemu in text mode

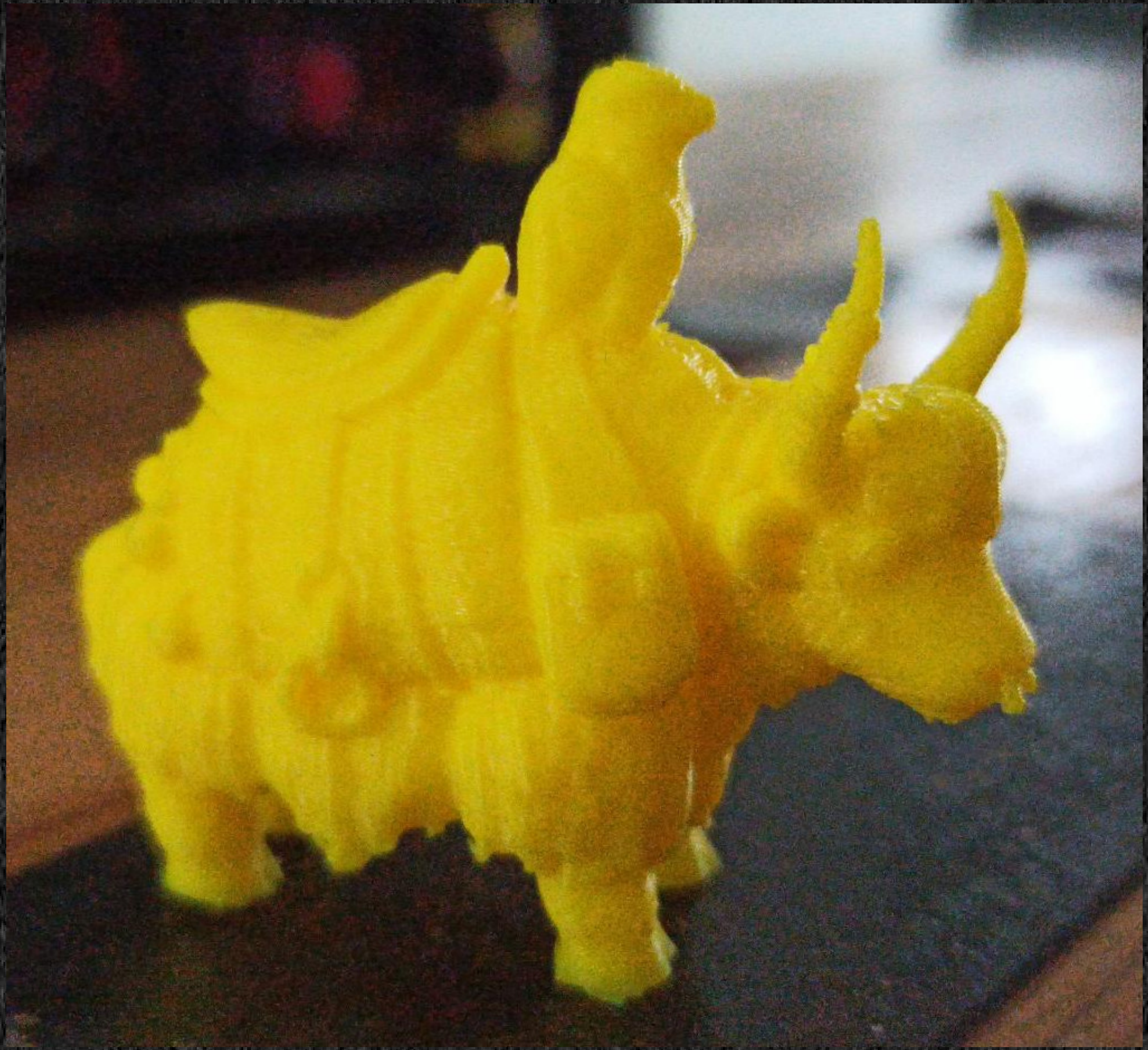
```
$ guix system image simple-1.scm -r virtsimple1.img
$ qemu-system-x86_64 -enable-kvm -nographic -m 4G virtsimple1.img

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
[    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 ~#
```

👍 IT WORKS 👍

- **Generation** of a complete operating system
  - Definition of less than 15 lines, with nothing more
  - Without downloading installation media
  - Up to date (e.g. kernel 6.5.9)
  - No need to customize afterwards (eg: ansible)
- The definition is minimal, but the VM is not
  - Basic packages are not required as part of VM.

THE ONLY THING LEFT TO DO IS



(Untranslatable bad french joke : Yack à faucon)

- Remove the superfluous, add packages AND services:
  add and modify list entries
  %base-packages and %base-services
- Factor functions and configurations between OS:
  create templates for simple writing
  guile module (glicid template v3)
- More advanced system for launching VMs:
  - libvirt/virt-manager
  - Proxmox

GLICID TEMPLATE

```
(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 Numerous definitions of networks, gateways, name servers…

2 Different service lists (also package lists), specific configs

3 Operating system definitions configured and ready for inheritance

GLICID FULL “DEBUG” VM

```
(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 ❶
  (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)  ❷

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

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

❶ Static-networking service specific to each VM instance

❷ “Debug” variant of the GLiCID template: the richest in options

INCLUDED:

- Network settings, DNS, NTP, SSH + admin keys, Syslog, Zabbix and Qemu Agents, Guix Channels…
- Debug variant adds: NSS LDAP, NSCD settings, configs by GIT, editors and numerous debug tools…

VM BUILDS

```
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-image①
dd to ceph, please wait, will take time ②

real	1m5.389s ③
```

① Image created in the local GUIX store

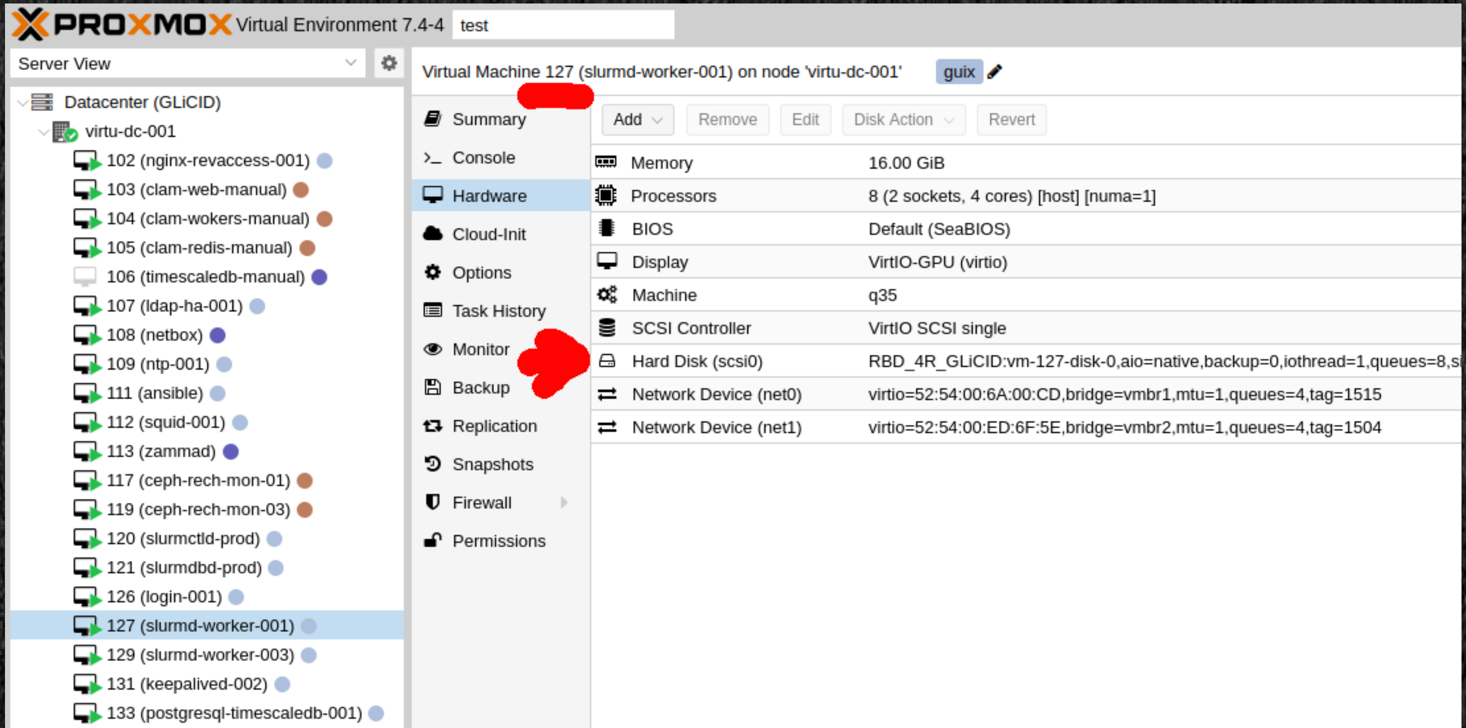
② Copied to Ceph pool RBD_4R_GLiCID (time-stamped name)

③ Image clone (VMroot_for-slides-test001_candidate)

A script allows you to rename this candidate to the name used by KVM or proxmox (vm-233-disk-0).

💡 Images usable wherever CEPH is available

DEPLOY ON PROXMOX



PROXMOX Virtual Environment 7.4-4 test

Server View

- Datacenter (GLiCID)
  - virtu-dc-001
    - 102 (nginx-revaccess-001) ●
    - 103 (clam-web-manual) ●
    - 104 (clam-wokers-manual) ●
    - 105 (clam-redis-manual) ●
    - 106 (timescaledb-manual) ●
    - 107 (ldap-ha-001) ●
    - 108 (netbox) ●
    - 109 (ntp-001) ●
    - 111 (ansible) ●
    - 112 (squid-001) ●
    - 113 (zammad) ●
    - 117 (ceph-rech-mon-01) ●
    - 119 (ceph-rech-mon-03) ●
    - 120 (slurmctld-prod) ●
    - 121 (slurmdbd-prod) ●
    - 126 (login-001) ●
    - 127 (slurmd-worker-001) ●
    - 129 (slurmd-worker-003) ●
    - 131 (keepalived-002) ●
    - 133 (postgresql-timescaledb-001) ●

Virtual Machine 127 (slurmd-worker-001) on node 'virtu-dc-001' guix

Memory	16.00 GiB
Processors	8 (2 sockets, 4 cores) [host] [numa=1]
BIOS	Default (SeaBIOS)
Display	VirtIO-GPU (virtio)
Machine	q35
SCSI Controller	VirtIO SCSI single
Hard Disk (scsi0)	RBD_4R_GLiCID:vm-127-disk-0,aio=native,backup=0,iothread=1,queues=8,s
Network Device (net0)	virtio=52:54:00:6A:00:CD,bridge=vmbr1,mtu=1,queues=4,tag=1515
Network Device (net1)	virtio=52:54:00:ED:6F:5E,bridge=vmbr2,mtu=1,queues=4,tag=1504

● VM Guix ● VM Debian ● VM Centos

- Disks identified as follows: vm-xxx-disk-yyy, here vm-127-disk-0
- 0 deployment scripts… just an image change

HERE WE GO !

(Some services are in high availability)

```
- crossed load balancers (keepalived) (4 Intra, 4 Pub)
- DNS
- LDAP
- SSH proxies
- “Block” and clustered NFS servers
- mirrors, proxies, reverse proxies, WWW servers
- zabbix server
- DB : mysql, postgresql + timescaledb
- slurmctld, slurmdbd
- login and devel machines (slurm client) ❶
- virtual pseudo computing nodes (slurm client) ❶
```

❶ Special trick required

💡 In certain cases, creation of new services (ldap, slurmctld, etc.)

IMMUTABLE MACHINES

- The configuration is **usually** embedded in /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 is read-only
- Immutable **AND** volatile machines : update = regenerate the machine!
- Easy, repeatable deployment in seconds
  - Generation → VM OFF → image change → VM ON
  - Stateless VM; some hold persistent data.
    - Use of %glicid-two-disks… templates : persistent LVM2 volumes
    - Snapshot persistent volume for deployment of new versions

DEPLOYMENTS FOR THE WHOLE CLUSTER

- all cluster machines mount `guix-store.intra.glicid.fr:/gnu` and `/var/gnu`
- all cluster machines define `GUIX_DAEMON_SOCKET="guix://guix-store.intra.glicid.fr"`
- consistent uid/gid thanks to LDAP
- guix-store.intra.glicid.fr is running guix-daemon and exports /gnu and /var/gnu
  - Cephfs should be the right way to do it…
    - Currently, MDS ceph servers (FS metadata) not enjoying tons of links…
  - For the moment : NFS
    - Transition problem : exporting Cephfs with NFS is a bad idea (locks, scalability AND stability problems)
  - So… NFS with ceph block devices (RBD) : no high availability

> 💡 Works well with foreign distros (no /gnu/store …)

SPECIAL CASE : GUIX-GENERATED NODES

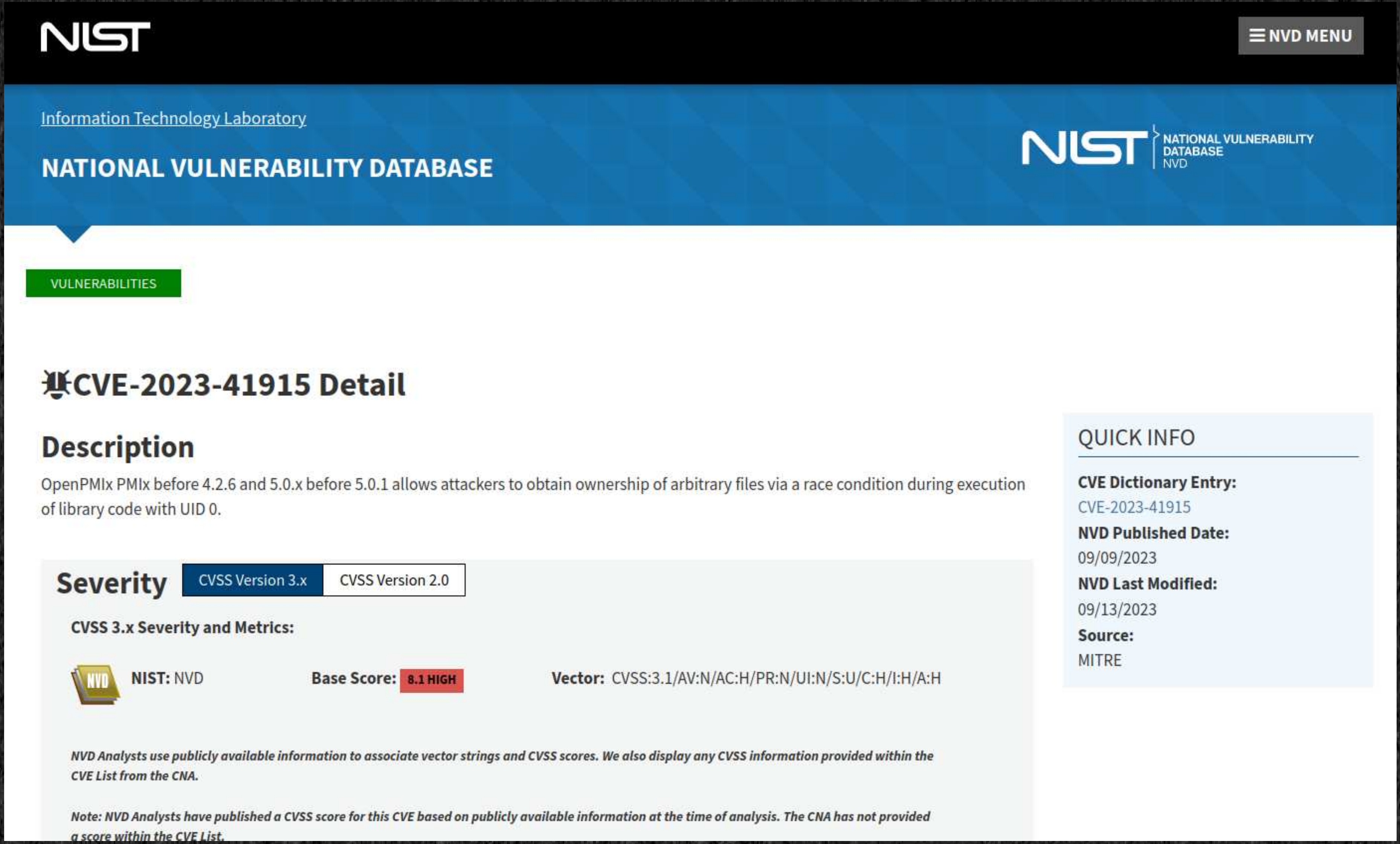
> 🔥 local `/gnu/store` already exists !

→ Special trick needed…

```
mount.nfs guix-store.intra.glicid.fr:/gnu /composite-guix/shared/gnu 1
mount -t overlay overlay -o lowerdir=/composite-guix/shared/gnu:/gnu /composite-guix/overlay/gnu 2
mount -o bind /composite-guix/overlay/gnu /gnu/ 3
```

1. mount the shared store on a specific path
2. create an overlayfs witch **2** stores as lowerdirs, effectively joining them, and mount on a specific path
3. the resulting mount is bind mounted on /gnu, "masking" the old local /gnu

RESPONSIVENESS AND CONTROL



Information Technology Laboratory

NATIONAL VULNERABILITY DATABASE

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

Early September 2023:

Our slurm uses pmix v3 for compatibility with the “Nautilus” slurm (installed by the manufacturer).

OPENPMIX PACKAGE UPDATE

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

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

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

❶ Inherit from a close version

❷ Change version and set it as new stable version

❸ Change the checksum of the package source.

Updates: slurm-glicid, openmpi-glicid… and the many packages that depend on it

Newly deployed VMs are not vulnerable

GUIX VS ANSIBLE/PUPPET/CHEF/SALT

💡 One does not prevent the others…

- **Generate** a complete and fully configured system
- Writing in a programming language, intelligence and freedom possible
- `guix deploy` : meets the needs of massive deployments

vs

- Configuration files specific to the system used
- **Modify and reconfigure** a pre-installed system (how/by whom?)

⚠ “immutable” aspect lost .

EXOTIC ARCHITECTURES

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

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

❶ Interesting for foresight

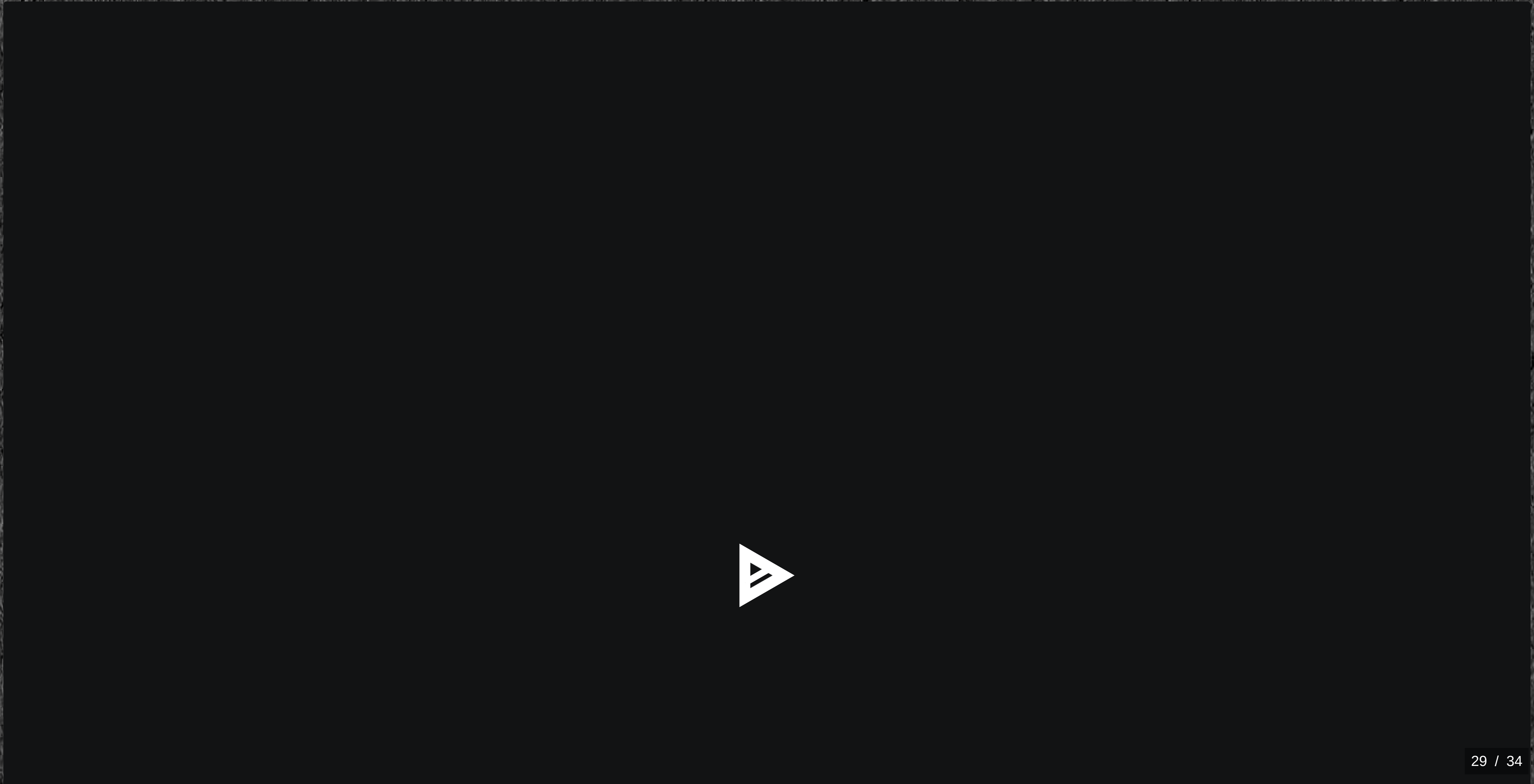
```
$ guix system image virtrv.scm --target=riscv64-linux-gnu
```

- … Works straight away! (tested on qemu and ARM and Risc-V physical machines)
- Limitations for certain packages (not cross-compilable or unsupported architecture)

CROSS-COMPILATION ?

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

CROSS-COMPILATION !

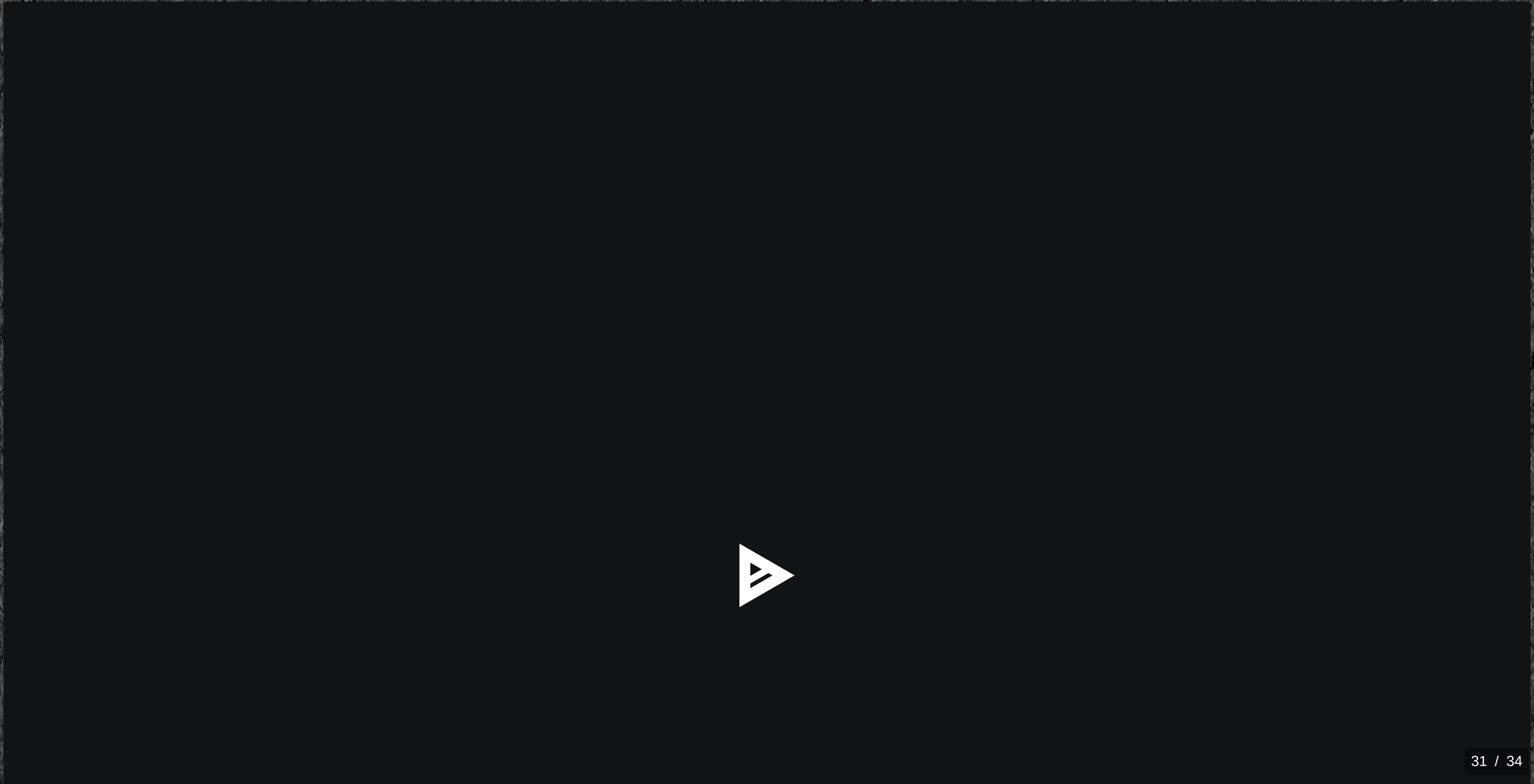


START VM

Use qemu, local uboot installation

```
guix shell --container --link-profile qemu opensbi-generic u-boot-qemu-riscv64-smode -- \
qemu-system-riscv64 -M virt  -nographic \
-bios ~/.guix-profile/fw_jump.elf \
-kernel ~/.guix-profile/libexec/u-boot.bin \
-m 3G \
-device virtio-blk-device,drive=hd0 -drive format=raw,file=virtrv.img,id=hd0
```

START VM



SUMMARY: PROS

- Composition of operating systems
  - (inherit base-os) = common core, risks of errors or omissions minimized
  - Effort pays off: capitalization, few rewrites observed in almost 3 years, time saved *ultimately*
- Control and chain of trust that go very far
  - Minimal boostrap, controlled sources, reproducible binaries and operating system
  - Fine control of what is installed, kernel "hardening" possible
  - Specific software and dependencies integrated consistently throughout
- Constants (networks, etc.) easy to modify: easy massive redeployment (GLiCID test/alpha/beta)
- Disposable machines that can be redeployed at will
  - No configuration after the fact
  - System reproducibility! (time-machine is usable)
    - Simple rollback (caution: snapshots for stateful VM volumes)
    - Machine definitions (and GUIX templates) are in GIT

SUMMARY: CONS

- Guile requires some learning
  - Different appropriation depending on team members
  - “Everyone does it differently.”
- If service or package not yet ported :
  - Get on with it : sometimes complicated
  - Spending time there: scarce resource…
    - Some packages or services require too much effort: **GLOBAL** effort required
      - Easy solution: deploy“ready-made” products temporarily
      - And try later :/
- Does not protect against bugs (package updates, etc.)
  - Beware of overconfidence and redeployment without verification
- “Bus factor”
  - 3 team members regularly generate packages and VMs.

THANK YOU FOR YOUR ATTENTION

Questions ?