INCUS — Migrer un conteneur vers la prochaine version LTS

Référence : Incus 6.0 LTS — conteneurs Ubuntu non privilégiés — migration 22.04 vers 24.04

Guide de procédure. Il décrit comment faire traverser une version LTS à un conteneur en service, sans le reconstruire.

⚠ Une ligne de ce document dépasse 84 caractères et s'enroulera dans le PDF sans signe visible : la ligne de source apt de la section *Réparer apt*. Une entrée de `sources.list` n'accepte aucune continuation — la recopier depuis le fichier `.md`, jamais depuis le PDF. Toutes les autres commandes longues ont été coupées par continuation `\`.

Le fichier `.md` de cette page : ouvrir le `.md`

Ce que cette page couvre : relever l'état d'un conteneur, répéter sa migration sur une copie, la conduire, réparer ce que l'outil laisse derrière lui, et vérifier que la machine fait toujours ce qu'on croit. La procédure est écrite pour Ubuntu et Incus, mais rien n'y dépend d'un service en particulier.

Contexte : un hôte Incus qui porte plusieurs conteneurs en macvlan, chacun avec sa propre adresse sur le réseau domestique, et une machine de sauvegarde qui vient tirer leurs données chaque nuit. Les adresses, les noms d'hôtes et les noms de dépôts sont des exemples à transposer.

Ce qui vaut le détour même sans suivre la procédure de bout en bout. Une migration de version peut ressusciter un dépôt tiers qu'on croyait enterré — section *Réparer apt*. Un paquet renommé reprend les fichiers de configuration de son prédécesseur sans poser de question et sans laisser de trace — section *Le relevé d'après*. Et un test de service qui interroge la racine d'un site peut mesurer autre chose que ce qu'on croit depuis des années — section *Le test*

fonctionnel de référence.

Convention : chaque bloc de commandes indique la machine où il s'exécute. Un bloc unique se copie tel quel ; des blocs séparés signifient qu'une édition ou une décision intervient entre eux. Les commandes `incus` se tapent sur l'hôte, dans une session ouverte par `ssh hostadmin@192.168.0.11`.

Migrer n'est pas reconstruire

Rebâtir une machine et la faire changer de version sont deux travaux différents, et ils n'ont pas les mêmes pièges. Une reconstruction part d'un système nu et pose ce qu'on veut ; une migration emmène ce qui est là — y compris ce qu'on a oublié d'y mettre, ce qui n'y sert plus, et les décisions prises quatre ans plus tôt.

Trois propriétés gouvernent tout le reste.

Une migration ne retire rien. L'outil met à niveau les paquets présents ; il ne désinstalle pas ce qui est devenu inutile. Un agent d'invité de machine virtuelle installé par recommandation ressort simplement dans sa version neuve. Tout ménage est un geste explicite — et il vaut mieux le faire avant : ce qui est retiré n'a pas à être migré.

Une migration ne voit pas ce qui est installé hors d'apt. Un logiciel posé par son propre installeur n'est ni mis à jour ni cassé : la migration passe à côté. C'est une bonne nouvelle pour sa survie, et une mauvaise pour le ménage automatique — voir *Le pré-nettoyage*.

Une migration conserve l'état. Les bases de données, les enregistrements locaux, les listes, les fichiers des sites : c'est précisément ce qu'une reconstruction perdrait. C'est la raison de migrer plutôt que de rebâtir, et c'est aussi ce qu'il faut vérifier après coup, puisque rien ne le signalera.

Choisir l’ordre et le moment

Du plus simple au plus chargé. La première migration du parc sert à apprendre la mécanique : elle doit porter sur le conteneur qui a le moins à perdre. Un conteneur qui ne dépend que de l’archive de la distribution, sans périphérique attaché ni logiciel hors paquets, se migre en une heure et enseigne les cinq écrans de l’outil.

Ce qui décide de la difficulté, dans l’ordre où ça fait mal :

- un dépôt tiers, qui doit publier pour la suite visée et sera désactivé en chemin ;
- un logiciel installé hors d’apt, qui devra peut-être être reconfiguré après coup ;
- un disque attaché, qui complique la répétition sur copie ;
- des services dont dépend le reste du réseau, qui imposent de savoir ce qui prend le relais pendant l’opération.

 Le bon créneau est le petit matin, pas la soirée. Les sauvegardes de la nuit viennent de réussir — le filet a moins de sept heures — et la fenêtre avant les suivantes est maximale. Une migration entamée le soir se déroule au contraire juste avant les tâches nocturnes, avec une sauvegarde vieille de vingt heures.

Un service interrompu n’est pas un problème, un service à moitié migré en est un. Décider d’avance qu’on accepte l’interruption simplifie tout : la répétition sur copie n’a plus à protéger la disponibilité, elle sert uniquement à éclairer les inconnues là où une erreur ne coûte rien.

Relever l’état initial

Rien de ce qui suit ne se devine. Chaque relevé sert deux fois : à préparer la migration, et à savoir après coup si quelque chose a changé.

Les sources et les clés

```
incus exec <conteneur> -- bash -c "grep -rh ^deb /etc/apt/sources.list \
  /etc/apt/sources.list.d/ 2>/dev/null"
incus exec <conteneur> -- ls -la /etc/apt/sources.list.d/
incus exec <conteneur> -- ls -la /etc/apt/trusted.gpg.d/
```

Un dépôt tiers peut être déclaré sans qu’aucun document ne le mentionne. La machine est la seule source fiable sur ce point.

Une clé posée dans `trusted.gpg.d/` sans `signed-by` vaut pour *toutes* les sources de la machine, pas seulement pour celle qui l’a apportée. Une clé oubliée là est un droit de signature laissé à un tiers.

⚠ `incus exec` ne lance pas d’interpréteur de commandes. Un `*`, un tube ou une redirection écrits directement après `--` sont traités par le shell de l’hôte, pas par celui du conteneur : `incus exec c1 -- du -sh /var/www/*` cherche un fichier littéralement nommé `*`. Pour donner un motif au conteneur, il faut `bash -c "…"`.

Ce que le dépôt tiers fournit vraiment

Un dépôt ajouté pour une chose en livre souvent d’autres. La question n’est donc pas « que fait-on des paquets qui portent son nom » mais « quels paquets viennent de lui » :

```
incus exec <conteneur> -- bash -c "apt-cache policy | grep -A2 sury"
incus exec <conteneur> -- bash -c "dpkg-query -W -f='\${Package} \${Version}\n' \
  | grep sury"
```

Des pièces du système peuvent en venir, pas seulement le logiciel visé — une bibliothèque d'images, une bibliothèque d'expressions régulières, un analyseur XML. Ce ne sont pas des dépendances du logiciel : ce sont des morceaux de la distribution que le dépôt tiers remplace par ses propres versions, et le serveur web s'en sert aussi.

Le dépôt tiers publie-t-il pour la suite visée

C'est l'inconnue qui peut faire dérailler toute l'opération, et elle se tranche en une commande — posée depuis la machine, pas cherchée sur le web :

```
incus exec <conteneur> -- bash -c \
  "curl -s https://packages.sury.org/php/dists/noble/InRelease | head -8"
```

Une réponse qui annonce la bonne suite et une date récente clôt la question. ⚠ Une absence de réponse, elle, arrête le chantier : migrer d'abord et découvrir ensuite que le dépôt ne publie pas pour la nouvelle base laisse une machine sans source pour la moitié de ses paquets.

La documentation ne suffit pas. Un script officiel d'ajout de dépôt construit sa suite à partir du nom de code de la machine, sans aucune liste de distributions acceptées : il ne prouve rien. C'est l'index du dépôt qui répond.

Ce qui est installé hors d'apt

```
incus exec <conteneur> -- ls -la /usr/local/
incus exec <conteneur> -- ls -la /opt/
```

Ce relevé décide de la section *Reconfigurer ce qui est installé hors d'apt*. Un panneau d'administration, un serveur applicatif posé par un installeur maison, un binaire déposé à la main : chacun survivra à la migration, et chacun peut avoir à être averti que le socle a changé.

L’instance elle-même

```
incus config show <conteneur> | grep -vE "^ +(volatile|image)"
incus config device list <conteneur>
incus storage info <pool>
```

⚠ Les périphériques attachés changent la nature de l’opération. Un disque déclaré ici est un système de fichiers qui vit hors du pool du conteneur : l’instantané ne le couvrira pas, et une copie ne doit surtout pas le monter en même temps que l’original.

 C’est `incus storage info` qui donne l’espace d’un pool, pas `incus storage list` — cette dernière n’affiche aucune colonne de taille et répond fidèlement à une autre question.

Le réseau et le nom d’hôte

```
incus exec <conteneur> -- cat /etc/netplan/50-static-public-ip.yaml
incus exec <conteneur> -- cat /etc/hosts
incus exec <conteneur> -- hostname -f
```

⚠ Les résolveurs du conteneur décident de ce qu’il peut faire pendant l’opération. Un conteneur qui résout par un autre conteneur du parc dépend de lui ; un conteneur qui se résout lui-même — cas d’un service DNS — devra aller chercher ses propres paquets par son résolveur de secours, précisément pendant que son service est arrêté. Si ce repli ne tient pas, la migration s’arrête au milieu, sans DNS pour se terminer.

`hostname -f` qui répond « Name or service not known » est un écart à corriger avant, pas après. Le fichier `/etc/hosts` d’un conteneur doit porter ses deux lignes ; sans elles, des services se plaignent au démarrage sans que personne ne les écoute depuis des années.

`/etc/hosts` complet, pour un conteneur nommé `squid-proxy` en 192.168.0.7 :

```
127.0.0.1 squid-proxy.example.com squid-proxy localhost
192.168.0.7 squid-proxy.example.com squid-proxy

# The following lines are desirable for IPv6 capable hosts
::1 ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
ff02::3 ip6-allhosts
```

Corriger `/etc/hosts` plutôt que de poser un nom dans la configuration d’un service : le remède vaut pour tout ce qui tourne dans le conteneur. Et il se double d’un enregistrement dans le résolveur du réseau — le fichier rend le nom résolvable de l’intérieur, l’enregistrement le rend résolvable de partout. Les deux gestes ne se remplacent pas.

Le test fonctionnel de référence

Il se bâtit avant la migration et se rejoue après, identique, pour que tout écart soit attribuable à l’opération et à rien d’autre.

⚠ Avant de croire un code de réponse, vérifier ce que la machine relaie réellement. Un mandataire inverse monté sur un sous-chemin rend aveugle tout test de la racine : interroger `/` sur un tel site mesure la présence d’un fichier d’accueil déposé à la création du site, jamais le service qu’on croit tester. Le défaut ne se révèle qu’en interrogeant un service volontairement arrêté — la seule condition où un mandataire vivant et un mandataire mort se distinguent.

```
incus exec <conteneur> -- bash -c \
  "grep -nE '<Location|ProxyPass ' /etc/apache2/sites-enabled/*.vhost"
```

Le test corrigé interroge le chemin réellement relayé par chaque site :

```
incus exec web-srv -- bash -c '
c() {
  printf "%-9s " "$1"
  curl -s -o /dev/null -w "%{http_code}\n" -k \
    --resolve "$2":443:127.0.0.1 "https://$2$3"
}
c blog     blog.example.com     /
c photos   photos.example.com   /
c dav      dav.example.com      /radicale/
c lt       lt.example.com       /v2/languages
c musique  musique.example.com  /
c routeur  routeur.example.com  /
'
```

Ce que chaque code prouve :

```
blog      200   le moteur du site répond
photos    401   l'authentification Apache est en place
dav       401   l'agenda est joint et réclame ses identifiants
lt        200   le correcteur rend sa liste de langues
musique   302   le lecteur redirige vers son interface
routeur   401   l'administration du routeur est protégée
```

Ce ne sont pas tous des `200`, et c’est voulu : un `200` partout signalerait qu’une protection a sauté.

Interroger un point d’entrée qui répond, pas le sous-chemin nu. `/v2/languages` rend une liste ; `/v2/` seul pourrait rendre `404` sans que rien ne soit cassé — et on retomberait dans le même piège, à l’envers.

Un `curl` prouve qu’un service répond, pas qu’il sert les bonnes données. Pour un agenda, la preuve qui vaut est un client qui synchronise réellement.

Suspendre les tâches nocturnes

Le danger n’est pas l’interruption du service, ce sont les sauvegardes. Les tirages partent avec `--delete` : sauvegarder une machine à moitié migrée écrase la bonne sauvegarde par une mauvaise, en silence.

La liste s’établit par la machine visée, pas par le nom des scripts — plusieurs portent un nom générique :

```
grep -l 192.168.0.6 /home/hostadmin/scripts/rsync/auto/*.sh
```

⚠ La crontab d’un compte n’est pas l’inventaire des tâches d’une machine. Une tâche lancée par la crontab de root n’apparaît nulle part dans celle de l’utilisateur qui porte les sauvegardes. Relire les deux, et au besoin le journal de la veille, qui montre ce qui a réellement tourné :

```
crontab -l
sudo crontab -l
```

Sur l’hôte — la copie de sûreté, puis l’éditeur :

```
crontab -l > ~/crontab-hostadmin-avant-migration.txt
```

```
crontab -e
```

Mettre un `#` devant chaque ligne qui vise le conteneur.

Sur l’hôte — les tâches internes au conteneur, mises de côté plutôt que commentées :

```
incus exec web-srv -- bash -c \
  "crontab -u webadmin -l > /root/crontab-webadmin-avant-migration.txt"
incus exec web-srv -- crontab -u webadmin -r
```

```
incus exec web-srv -- crontab -u webadmin -l
```

La dernière commande doit répondre `no crontab for webadmin`.

Cette étape peut être sans objet, et c’est une décision, pas un oubli. Une migration entamée à six heures du matin, alors que les tâches nocturnes ont toutes tourné et que les suivantes ne partent pas avant le soir, dispose de dix-sept heures de marge pour une opération d’une heure. Le vérifier plutôt que le supposer, puis le noter, parce que l’étape de rétablissement tombe avec celle-ci.

Le pré-nettoyage

Ce qui est retiré n’a pas à être migré. Quinze paquets devenus inutiles deviendraient autant de transitions, de questions et de minutes — pour un logiciel que rien n’appelle. Les retirer d’abord supprime une classe entière de risques au lieu de la traverser.

Simuler d’abord, et lire la liste, qui diffère d’un conteneur à l’autre :

```
incus exec <conteneur> -- apt-get -s purge open-vm-tools \
  | grep -E "^(Purg|Remv)"
```

⚠ `open-vm-tools` est l’agent invité des machines virtuelles VMware, et il n’a aucun objet dans un conteneur. Il arrive par une recommandation du métapaquet `ubuntu-server`, donc sur toutes les images de serveur, et c’est lui qui déclenche le bruit de périphériques pendant la migration. Une migration ne le retire pas — le geste est explicite.

Recenser sur tout le parc évite de chercher de mémoire :

```
for c in squid-proxy pi-hole navidrome languagetool web-srv; do
  printf "%-18s " "$c"
```

```
  incus exec "$c" -- dpkg-query -W -f='${Status}\n' open-vm-tools \
    2>/dev/null || echo "absent"
done
```

□□ Lire le résultat correctement : un paquet purgé rend `unknown ok not-installed` ; un paquet jamais installé n’a pas de fiche du tout et tombe sur le `absent` de repli. Les deux réponses ne disent pas la même chose.

L’instantané de sûreté, puis la purge :

```
incus snapshot create <conteneur> avant-menage
incus exec <conteneur> -- apt-get -y purge open-vm-tools
incus exec <conteneur> -- apt-get -s autoremove --purge \
  | grep -E "^(Purg|Remv)"
```

Cette liste d’orphelins ramasse les restes de toutes les opérations précédentes, pas seulement de la dernière. Un paquet sans rapport avec ce qu’on vient de purger y figure parce que personne n’a lancé d’`autoremove` depuis un ménage antérieur. La lire avant d’appliquer.

⚠□ `apt autoremove` ne connaît pas les dépendances des logiciels installés hors d’apt. Une bibliothèque dont un binaire posé par un installeur maison a besoin apparaît comme orpheline, et sa suppression produit une panne qu’apt ne pouvait pas prévoir. Le symptôme peut n’être qu’un style ou un script servi avec le mauvais type de contenu — aucun message.

Protéger ce qui doit l’être, plutôt que d’espérer :

```
incus exec <conteneur> -- apt-mark manual mailcap
```

`apt-mark auto` défait ce marquage le jour où on voudra vraiment retirer le paquet. Et le même moyen sert à reporter une décision : une extension devenue orpheline la veille d’une migration est exactement le geste dont on se demandera trois jours plus tard s’il est la cause de quelque chose.

Appliquer :

```
incus exec <conteneur> -- apt-get -y autoremove --purge
```

⚠ `--purge` n’est pas optionnel sur un `autoremove` : sans lui, apt garde les fichiers de configuration et fabrique autant de paquets en état `rc`.

Retirer les sources et les clés devenues inutiles

Le fichier de source se localise plutôt qu’il ne se devine :

```
incus exec <conteneur> -- grep -rl goaccess \
  /etc/apt/sources.list /etc/apt/sources.list.d/
```

Sur l’hôte — la mise à l’écart :

```
incus exec <conteneur> -- mkdir -p /root/apt-retire
incus exec <conteneur> -- mv /etc/apt/sources.list.d/goaccess.list \
  /root/apt-retire/
incus exec <conteneur> -- bash -c "mv \
  /etc/apt/trusted.gpg.d/goaccess.gpg* \
  /root/apt-retire/"
incus exec <conteneur> -- ls -l /etc/apt/trusted.gpg.d/
incus exec <conteneur> -- apt-get update
```

On déplace plutôt qu’on supprime. Une clé apt n’a d’effet que par sa présence dans le répertoire : l’en sortir est un correctif complet et réversible.

⚠ Commenter une source ne retire pas sa clé, et retirer une clé ne désactive pas sa source. Les deux gestes vont ensemble — et la section *Réparer apt* montre pourquoi commenter ne suffit pas.

`apt update` doit se dérouler sans `NO_PUBKEY` ni `EXPKEYSIG`, et le dépôt qu’on garde doit toujours répondre.

Le redémarrage de validation

```
incus restart <conteneur>
sleep 40
incus exec <conteneur> -- ls -l /var/run/reboot-required
incus exec <conteneur> -- systemctl --failed --no-legend
```

 Faire ce redémarrage séparément sépare les causes. Si quelque chose ne remonte pas, on le sait tout de suite et on sait de quoi — pas trois heures plus tard, au milieu d'une migration où tout est suspect à la fois.

⚠ `systemctl is-active` interrogé trop tôt après un démarrage rend une réponse fausse. Cinq secondes après le démarrage d'un conteneur, un service peut se déclarer `inactive` alors qu'il démarre. Attendre vingt secondes — quarante pour un conteneur qui lève une base de données et plusieurs interpréteurs.

La répétition sur copie

Elle sert à éclairer les inconnues là où une erreur ne coûte rien, pas à prouver que tout ira bien. Ce qu'elle établit est vrai ; ce qu'elle ne montre pas reste à découvrir sur la vraie machine.

Créer la copie, dans l'ordre

Sur l'hôte :

```
incus copy web-srv ns2-essai
incus config set ns2-essai boot.autostart false
incus config device remove ns2-essai web
incus config get ns2-essai boot.autostart
incus config device list ns2-essai
incus config device list web-srv
```

 L'ordre n'est pas négociable. `incus copy` produit une copie arrêtée, mais elle

porte l’adresse de l’original et la déclaration de ses périphériques. Deux instances qui montent le même système de fichiers ne produisent pas un message d’erreur : elles produisent une corruption. Les deux gestes qui suivent la copie viennent avant tout démarrage.

⚠ `boot.autostart false` se relit après l’avoir posé. Une consigne prescrite et jamais exécutée ne se voit pas — et un redémarrage de l’hôte ferait alors démarrer deux conteneurs sur la même adresse.

Trois relectures, pas une : `boot.autostart` de la copie à `false`, périphériques de la copie vides, périphériques de l’original intacts. La troisième compte autant que les deux autres : elle prouve qu’on a retiré le disque de la copie et non de la machine en service.

La copie emporte les instantanés de l’original, et ils partent avec elle. La MAC régénérée n’expose à rien tant que l’adressage est statique : la copie emporte `dhcp4: no` et ne demande jamais d’adresse au routeur.

Le renommage du conteneur d’essai est inutile. Incus pose le nom de l’instance comme nom d’hôte au démarrage : les deux se distinguent déjà dans les journaux tant qu’on passe par `incus exec <nom> -- …` depuis l’hôte plutôt que par un shell intérieur.

Lui donner sa propre adresse

On ne peut pas ouvrir d’éditeur dans un conteneur arrêté. Le fichier se prépare sur l’hôte, puis se dépose.

Sur l’hôte :

```
nano /tmp/netplan-essai.yaml
```

⚠ Ce bloc n’est pas une commande — c’est le contenu du fichier, à coller dans l’éditeur ouvert ci-dessus. Il est identique à celui de l’original, sauf l’adresse :

```
network:
  version: 2
  renderer: networkd
  ethernets:
    eth0:
      dhcp4: no
      dhcp6: no
      addresses: [192.168.0.4/24]
      nameservers:
        addresses:
          - 192.168.0.5
          - 208.67.222.222
      routes:
        - to: default
          via: 192.168.0.1
```

Sur l’hôte — le dépôt et sa relecture :

```
incus file push /tmp/netplan-essai.yaml \
  ns2-essai/etc/netplan/50-static-public-ip.yaml --mode 0600
incus file pull ns2-essai/etc/netplan/50-static-public-ip.yaml -
```

⚠ La relecture n’est pas décorative. Si le chemin de dépôt était fautif, le dépôt réussirait quand même — en créant un fichier ailleurs — et la copie démarrerait sur l’adresse de l’original, en conflit avec le service en marche.

`--mode 0600` préserve les permissions ; sans lui, netplan se plaint d’un fichier trop ouvert.

 Les résolveurs de la copie reproduisent ceux de l’original, pas ceux qui seraient commodes. Une copie qui résoudrait autrement ne répéterait pas la bonne situation — et c’est particulièrement vrai d’un conteneur qui se désigne lui-même comme résolveur : en le faisant pointer vers sa propre adresse d’essai, on reproduit le moment où son service est arrêté et où le repli doit tenir.

Le premier démarrage

Sur l’hôte :

```
incus start ns2-essai
sleep 30
incus exec ns2-essai -- ip -br addr show eth0
incus exec ns2-essai -- df -h /var/www
incus exec ns2-essai -- ls -la /var/www
incus exec ns2-essai -- systemctl --failed --no-legend
```

L’adresse doit être celle de la copie, et le point de montage du disque retiré doit avoir disparu.

`ls -l` ne montre pas les fichiers cachés, et le nombre qu’il affiche pour un répertoire n’est pas un décompte d’entrées — c’est un nombre de blocs. Un répertoire déclaré vide sur cette foi peut très bien contenir le fichier qui explique tout. `ls -la`, toujours.

□□ Un point de montage cache ce qu’il y a en dessous. Une arborescence peut vivre sous lui, invisible depuis des années, et une seule vue ne suffit pas à conclure — il faut regarder depuis le conteneur et depuis l’hôte.

Ce qu’une copie ne reproduit pas

⚠□ L’état volatil ne traverse pas la copie. Le drapeau `/var/run/reboot-required`, les verrous, les sockets, les processus lancés à la main vivent dans un système de fichiers en mémoire : `incus copy` copie le disque. Une copie part donc d’un état plus propre que l’original, et la répétition ne peut pas révéler ce genre d’obstacle — l’outil de migration refusera de démarrer sur la vraie machine en réclamant un redémarrage que la copie n’exigeait pas.

⚠□ Le nom d’hôte d’une copie ne survit pas au redémarrage. Incus pose le nom de l’instance comme nom d’hôte à chaque démarrage : sur une copie nommée autrement, `hostname -f` ne résout plus, les services s’en plaignent, et un panneau d’administration qui se sert de ce nom peut s’en trouver troublé. Le

reposer après chaque redémarrage de la copie :

```
incus exec ns2-essai -- hostnamectl set-hostname web-srv
```

Sur la vraie machine, ce geste est inutile : l’instance porte déjà son nom.

Les tâches planifiées de la copie

Une copie emporte les crontabs de l’original, et certaines écrivent réellement — un vidage de bases de données à minuit, par exemple.

Sur l’hôte :

```
incus exec ns2-essai -- bash -c \
  "crontab -u webadmin -l > /root/crontab-webadmin-avant-essai.txt"
incus exec ns2-essai -- crontab -u webadmin -r
```

Celles qui tourneront pendant la vraie migration se laissent en place. Une tâche d’administration qui s’exécute chaque minute continuera de le faire pendant l’opération réelle : c’est donc ce qu’une répétition doit montrer.

Redonner à la copie le contenu d’un disque attaché

Une copie privée de son disque ne démarre pas ses services, et l’erreur se découvre par couches parce qu’Apache s’arrête à la première faute : on ne voit la deuxième qu’après avoir levé la première. D’abord une racine de site manquante, ensuite un certificat introuvable, et ainsi de suite.

Trois passes suffisent. Sur l’hôte :

```
incus exec web-srv -- bash -c "cd /var/www && find . -type d -print0 \
```

```
  | tar -cf - --numeric-owner --no-recursion --null -T -" < /dev/null \
  | incus exec ns2-essai -- tar -xf - --numeric-owner -C /var/www

incus exec web-srv -- tar -cf - --numeric-owner -C /var/www \
  --exclude=./clients/client1/web2 . < /dev/null \
  | incus exec ns2-essai -- tar -xf - --numeric-owner -C /var/www

incus exec web-srv -- tar -cf - --numeric-owner -C /var/www \
  --exclude=./clients/client1/web2/web/galleries \
  --exclude=./clients/client1/web2/web/_data \
  ./clients/client1/web2 < /dev/null \
  | incus exec ns2-essai -- tar -xf - --numeric-owner -C /var/www
```

Ce que fait chaque passe :

1. La charpente — tous les répertoires, sans un octet de contenu, avec leurs propriétaires. Instantané. C’est elle qui donne au serveur web ses chemins de journaux et au panneau un endroit où écrire.
2. Les contenus sauf le plus gros site — ce qui tournera donc pour de vrai.
3. Le gros site sans ses originaux ni ses vignettes — son code et sa configuration seulement.

□□ Le `< /dev/null` n’est pas décoratif : sans lui, `incus exec` peut ouvrir un terminal, ce qui abîmerait le flux binaire de `tar`.

`--numeric-owner` préserve les propriétaires par leur numéro, ce qu’exigent les scripts d’un panneau d’administration qui vérifient à qui appartient ce qu’ils exécutent.

⚠□ `--exclude` s’applique à TOUT répertoire portant ce nom, où qu’il soit. Un motif donné par un nom simple peut écarter bien plus que prévu — et sans le moindre avertissement. Un motif d’exclusion se donne par son chemin complet.

□ Un conteneur non privilégié ne peut pas créer de fichier de périphérique. Un environnement confiné qui porte son propre `/dev` miniature fait échouer `tar` sur

ses trois nœuds :

```
tar: ./clients/client1/web6/dev/urandom: Cannot mknod: Operation not permitted
```

`tar` signale et poursuit : il sort en échec sans avoir interrompu la copie. La conséquence vaut bien au-delà de la répétition : restaurer une arborescence par `tar` dans un conteneur non privilégié laissera ces fichiers de côté. À savoir avant d'en avoir besoin.

L'instantané, et ce qu'il ne couvre pas

Sur l'hôte :

```
incus snapshot create <conteneur> avant-migration
incus snapshot list <conteneur>
```

 L'instantané ne couvre pas un disque attaché en brut. Il photographie le système de fichiers du conteneur — donc `/etc`, les comptes, et les bases de données, qui vivent dans `/var/lib/mysql`. Il ne photographie pas les fichiers portés par un périphérique déclaré à part.

Geste	Bases de données	Fichiers d'un disque attaché
`incus snapshot`	photographiées	hors cadre
La migration elle-même	inchangées	inchangés
`incus restore`	ramenées en arrière	restent au présent

 Les deux premières lignes disent que le raisonnement intuitif est bon : service arrêté, tout se fige, on migre, bases et fichiers restent d'accord. Le décalage n'existe qu'au retour en arrière, et il reste théorique tant que rien n'écrit dans le disque attaché entre-temps.

 « J'ai un instantané » ne veut donc pas dire « j'ai tout ». Ce qui ramènerait

les fichiers du disque attaché, c’est le tirage nocturne de la machine de sauvegarde. Deux filets différents, et il faut savoir lequel attrape quoi. Avant de commencer, vérifier la fraîcheur du dernier tirage du module qui couvre ce disque : c’est le seul filet de ce côté.

Le rattrapage, avant la migration

Mettre la machine à jour dans sa version actuelle avant de changer de version. Sur l’hôte :

```
incus exec <conteneur> -- apt-get update
incus exec <conteneur> -- apt-get -y \
  -o Dpkg::Options::=--force-confold full-upgrade
incus exec <conteneur> -- cat /etc/update-manager/release-upgrades
incus exec <conteneur> -- ls -l /var/run/reboot-required
```

`--force-confold` est délibéré ici : on garde la configuration actuelle pour ce rattrapage, parce que la décision sur les fichiers de configuration appartient à la migration, et qu’on ne veut pas qu’elle se joue par accident dans une étape préparatoire.

`Prompt=lts` doit figurer dans `release-upgrades`, sans quoi la version LTS suivante ne sera pas proposée.

`needrestart` peut demander quels services relancer : accepter sa sélection telle quelle. Ce n’est pas une question de configuration.

Si le drapeau de redémarrage est levé, redémarrer — l’outil de migration refuse de partir sinon :

```
incus restart <conteneur>
sleep 40
incus exec <conteneur> -- ls -l /var/run/reboot-required
incus exec <conteneur> -- systemctl --failed --no-legend
```

Le `ls` doit répondre `No such file or directory`.

La migration

Sur l’hôte :

```
incus exec <conteneur> -- do-release-upgrade
```

Les cinq réponses, dans l’ordre où elles arrivent :

1. « Do you want to start the upgrade? » → oui
2. Écran de texte plein écran avec une invite `:` en bas à gauche → `q`. C’est le lecteur des notes des mainteneurs ; il revient plusieurs fois et il n’y a rien à décider. ⚠ Ne pas le confondre avec les questions sur les fichiers de configuration.
3. Tout écran sur un fichier de configuration modifié → garder la version locale. Dans la forme encadrée, la bonne ligne est déjà surlignée et `Entrée` suffit ; dans la forme en lettres, c’est `N`. ⚠ `N` ne veut pas dire « non » : il désigne la deuxième ligne de la liste, `keep your currently-installed version`.
4. « Remove obsolete packages? » → oui
5. « Restart the system…? » → non, on redémarre le conteneur soi-même après avoir regardé.

Compter vingt à quarante minutes.

⚠ Une coupure de la session vers l’hôte n’interrompt pas la migration : l’outil se relance lui-même dans un `screen`. La preuve en est le journal qu’il laisse dans `/var/log/dist-upgrade/`.

`dpkg` dépose la version neuve de chaque fichier conservé à côté, en `.dpkg-dist`. Rien n'est perdu, on comparera plus tard.

Deux familles de messages ressemblent à des erreurs et n'en sont pas.

Les `Failed to write 'change' to '/sys/…/uevent': Permission denied` viennent d'un déclenchement de règles de périphériques lancé par un script d'installation. Dans un conteneur non privilégié, `/sys` est en lecture seule. Les chemins le disent mieux que n'importe quelle explication : ils nomment le contrôleur disque de l'hôte et ses ports. Le conteneur voit le matériel de l'hôte et ne peut pas y toucher — c'est un message d'erreur qui prouve que le confinement fonctionne.

Les `Installing new version of config file …` sont l'inverse d'un problème : dpkg pose la version neuve parce que ces fichiers-là n'avaient pas été modifiés.

Le message de fin est déroutant. Une fois la migration faite, relancer la commande rend « There is no development version of an LTS available. To upgrade to the latest non-LTS development release set Prompt=normal ». Cela veut dire « il n'y a plus de LTS suivante », pas « la migration a échoué ». Ne surtout pas suivre le conseil affiché : `Prompt=normal` ferait viser une version de développement non-LTS.

Réparer apt

C'est l'étape qu'on saute le plus facilement, et celle qui laisse les dégâts les plus discrets.

```
incus exec <conteneur> -- ls -l /etc/apt/sources.list.d/
```

⚠ Une migration de version peut RESSUSCITER un dépôt tiers. L'outil trouve la ligne d'un dépôt commentée depuis longtemps, la convertit au format moderne — un fichier `.sources` — et la laisse active : sans `Enabled: no`, un tel fichier est

en service. Vers un dépôt vidé de ses paquets, et toujours pointé sur l’ancienne suite.

```
Types: deb
URIs: https://ppa.launchpadcontent.net/ondrej/php/ubuntu/
Suites: jammy
Components: main
```

 Désactiver un dépôt en commentant sa ligne ne suffit donc pas : il faut retirer le fichier.

Sur l’hôte :

```
incus exec <conteneur> -- mkdir -p /root/apt-retire
incus exec <conteneur> -- mv \
  /etc/apt/sources.list.d/ondrej-ubuntu-php-jammy.sources \
  /root/apt-retire/
```

Le dépôt tiers légitime, lui, a été proprement désactivé : son fichier porte désormais un suffixe `.distUpgrade`. Il est donc à réécrire, pointé sur la nouvelle suite.

Sur l’hôte — préparer le fichier :

```
nano /tmp/php.list
```

⚠ Ce bloc n’est pas une commande — c’est le contenu du fichier, à coller dans l’éditeur ouvert ci-dessus. Une seule ligne :

```
deb [signed-by=/usr/share/keyrings/deb.sury.org-php.gpg]
https://packages.sury.org/php/ noble main
```

⚠ Une entrée de source apt n’accepte aucune continuation. Cette ligne dépasse la largeur d’une page et s’enroulera dans le PDF sans signe visible : la

recopier depuis le fichier `.md`, jamais depuis le PDF. Le seul mot qui change est le nom de code de la distribution.

Sur l'hôte — la pose et les contrôles :

```
incus file push /tmp/php.list \
  <conteneur>/etc/apt/sources.list.d/php.list --uid 0 --gid 0 --mode 0644
incus exec <conteneur> -- cat /etc/apt/sources.list.d/php.list
incus exec <conteneur> -- apt-get update
incus exec <conteneur> -- pro status
```

⚠ `incus file push` ne pose pas le fichier en `root:root`. Sans `--uid 0 --gid 0`, il arrive avec le propriétaire par défaut du conteneur. Les deux options font partie de la commande, pas de ses raffinements.

`apt update` doit récupérer l'index du dépôt sur la nouvelle suite, sans erreur de clé. Et si la machine porte un abonnement de maintenance étendue, `pro status` doit montrer ses services toujours `enabled` — la migration ne les perd pas.

Vérifier cela par deux commandes indépendantes plutôt que par une lecture de fichiers. Un `grep` récursif sans `-H` sur un répertoire concatène ses fichiers et perd leur provenance : une ligne lue là peut appartenir à tout autre chose que ce qu'on croit examiner.

Le sort des bibliothèques venues du dépôt tiers

Deux cas se produisent, et c'est le numéro de version qui départage :

- la version de la distribution est plus récente → apt remplace la bibliothèque du dépôt tiers, et le paquet rentre dans le rang ;
- la version du dépôt tiers est plus haute → apt ne remplace pas, et le paquet reste sans source jusqu'à ce que le dépôt soit repointé.

Un paquet qui reste figé sur une compilation pour l'ancienne distribution

fonctionne, mais plus personne ne le corrigera. C’est à trancher à part, pas dans le feu de la migration.

Aligner les paquets sur la nouvelle suite

Le dépôt tiers repointé, il reste à récupérer ses paquets reconstruits pour la nouvelle base.

Sur l’hôte :

```
incus exec <conteneur> -- apt list --upgradable
```

Le garde-fou, qui est ce qui rend l’opération sûre :

```
incus exec <conteneur> -- bash -c '
if apt-get -s full-upgrade 2>&1 | grep -q "^Remv" ; then
  echo "ARRET — apt voudrait retirer des paquets :"
  apt-get -s full-upgrade 2>&1 | grep "^Remv"
else
  apt-get -y -o Dpkg::Options::=--force-confold full-upgrade
fi'
```

```
incus exec <conteneur> -- apt list --upgradable
```

La seconde liste doit être vide.

□□ Ces paquets sont des reconstructions à l’identique : même numéro de version, seul le socle change. Il n’y a aucune migration du logiciel à faire — juste un changement de base. C’est ce qui rend `--force-confold` démontrablement sans risque à cette étape : les versions étant rigoureusement les mêmes des deux côtés, il ne peut pas exister de changement de fichier de configuration à

arbitrer.

Des paquets deviennent orphelins et ne sont pas retirés. La liste se lit avant de s’appliquer, et elle se lit d’autant plus attentivement que la machine porte du logiciel installé hors d’apt. Ce ménage appartient au froid, pas à la migration.

Le redémarrage complet et les vérifications

Il révèle ce qui ne tenait qu’à un service lancé à chaud. Sur l’hôte :

```
incus restart <conteneur>
sleep 40
incus exec <conteneur> -- lsb_release -ds
incus exec <conteneur> -- hostname -f
incus exec <conteneur> -- ls -l /var/run/reboot-required
incus exec <conteneur> -- systemctl --failed --no-legend
incus exec <conteneur> -- bash -c "ss -ltn"
```

Puis le test fonctionnel de référence, rejoué exactement tel qu’il a été pris avant.

⚠ Un service qui ressort `inactive` ne se conclut pas d’une seule mesure, et un service qui répond non plus. La famille de pannes qui traverse tout ce travail est celle du mécanisme qui se déclare en service sans agir : un dépôt vidé qui répond encore, un pare-feu `enabled` et `inactive` à la fois, une tâche planifiée qui réussit à ne rien faire, un service mort dont le port semble répondre. Aucune de ces pannes ne produit de message ; toutes se trouvent en comparant deux mesures indépendantes — l’unité et le processus, le port et le contenu servi, le journal et l’état.

⚠ Ne jamais tester un service macvlan depuis l’hôte. Les paquets partent par

l’interface parente, où le macvlan interdit le dialogue entre l’hôte et ses conteneurs. Le test se fait depuis une autre machine du réseau :

```
rsync --list-only rsync://test@192.168.0.6/
```

□□ Un test comparatif se valide lui-même ; un test isolé, non. Si l’original et la copie figurent dans la même commande, un obstacle de chemin les frapperait tous les deux — et l’écart resterait lisible.

Reconfigurer ce qui est installé hors d’apt

Un panneau d’administration posé par son propre installeur a traversé la migration sans être touché. Deux questions se posent, et les deux ont une réponse mesurable.

Garde-t-il mémoire de l’ancienne distribution ? S’il croyait encore tourner sur l’ancienne base, sa prochaine régénération de configuration emploierait les gabarits de la mauvaise distribution — une panne différée et silencieuse.

```
incus exec <conteneur> -- bash -c \
  'grep -rlE "jammy|22\.04" /usr/local/ispconfig/ 2>/dev/null | head -20'
```

□□ La commande ne demande que les noms des fichiers, pas leur contenu : celui de la configuration porte le mot de passe de la base.

Un fichier de détection du système qui contient une table des distributions connues n’est pas une mémoire de ce que la machine a été : c’est une liste de ce qu’elle pourrait être. La réponse qui tranche vient de l’exécution elle-même, qui annonce le système qu’elle voit.

Faut-il lui faire réécrire ses configurations ? Oui, si ce sont elles qui

portent les hôtes virtuels et les réglages de service.

Un outil de mise à jour refuse souvent de tourner quand aucune version plus récente n'existe. La reconfiguration après une migration de système demande donc `--force`, et l'avertissement sur les retours en arrière ne s'applique pas : la version stable *est* celle qui est installée.

 Une reconfiguration réécrit les hôtes virtuels et les pools. C'est le moment où une modification faite à la main dans un de ces fichiers disparaîtrait. Mettre de côté ce dont on veut être sûr, avant :

```
incus exec <conteneur> -- bash -c "cp -aL \
  /etc/apache2/sites-enabled/100-dav.example.com.vhost \
  /root/100-dav.vhost.avant"
```

Sur l'hôte — la reconfiguration :

```
incus exec <conteneur> -- ispconfig_update.sh --force
```

Réponses : source `stable`, sauvegarde complète `yes`, permissions de la base maître `no`, services `yes`, nouveau certificat `no`, tâches planifiées `yes`.

Puis la comparaison, qui transforme une déduction en mesure :

```
incus exec <conteneur> -- bash -c "diff /root/100-dav.vhost.avant \
  /etc/apache2/sites-enabled/100-dav.example.com.vhost && echo identique"
```

Des directives saisies dans le champ prévu du panneau sont réécrites à l'identique : elles font partie de la configuration, il n'y a rien à « réintégrer ». Ce qui a été écrit à la main dans un fichier, en revanche, est perdu — et ne se découvrirait qu'à la panne suivante.

Une étape de reconfiguration qui écrit des règles de pare-feu ne met pas le pare-feu en marche. La crainte est légitime, la mesure l'écarte : `systemctl is-`

`enabled`, `systemctl is-active` et l’état du pare-feu lui-même se lisent après, et disent la même chose qu’avant.

Le relevé d’après — ce qu’aucun avertissement ne signalera

⚠ Une migration de version se termine par un relevé des réglages qui comptent, pas par la lecture des avertissements — il n’y en a pas.

Un paquet renommé reprend les conffiles sans rien demander

Quand un paquet en remplace un autre sous un nom différent — un serveur de bases de données dont le numéro de version fait partie du nom de paquet, par exemple — il déclare `Replaces:` sur son prédécesseur. Et dpkg ne pose aucune question dans ce cas : pas d’écran de configuration, pas de `.dpkg-old`, pas une ligne au journal. Le fichier local est remplacé par celui du paquet neuf, en silence.

C’est ainsi qu’un réglage aussi net que l’adresse d’écoute d’un serveur de bases de données disparaît sans laisser de trace.

```
incus exec <conteneur> -- mysql -e "SELECT @@bind_address;"
incus exec <conteneur> -- ss -lntp | grep 3306
```

Le remède n’est pas de rééditer le fichier du paquet — la prochaine reprise de nom le reprendrait de la même façon. C’est de poser les réglages locaux dans un fichier qui n’appartient à aucun paquet, dans le répertoire de surcharge prévu pour ça, avec un nom qui passe en dernier.

Sur l’hôte :

```
nano /tmp/99-webadmin.cnf
```

⚠ Ce bloc n’est pas une commande — c’est le contenu du fichier, à coller dans l’éditeur ouvert ci-dessus :

```
# Réglages locaux — ce fichier n'appartient à aucun paquet
# et survit donc aux migrations de version et aux reprises de nom.
[mysqld]
bind-address            = 127.0.0.1
```

Sur l’hôte — la pose et le contrôle :

```
incus file push /tmp/99-webadmin.cnf --uid 0 --gid 0 \
  <conteneur>/etc/mysql/mariadb.conf.d/99-webadmin.cnf
incus exec <conteneur> -- systemctl restart mariadb
incus exec <conteneur> -- mysql -e "SELECT @@bind_address;"
incus exec <conteneur> -- ss -lntp | grep 3306
```

L’écoute ne doit apparaître que sur l’adresse locale.

La liste des fichiers de configuration se relève, elle ne se prédit pas

```
incus exec <conteneur> -- bash -c "find /etc -name '*.dpkg-dist' | sort"
```

C’est la machine qui donne la liste exacte — les fichiers qu’on s’attendait à voir proposer ne le sont pas toujours, et d’autres apparaissent qu’on n’avait pas nommés. Chacun se compare à sa version locale avant d’être écarté, et la plupart ne portent qu’un écart cosmétique.

 Un seul peut porter la ligne dont dépend tout le service. Le `apache2.conf` local d’un serveur qui héberge des sites par panneau porte :

```
IncludeOptional sites-enabled/
```

Et non `sites-enabled/*.conf`, comme la version du paquet. Les hôtes virtuels engendrés par un panneau ne finissent pas forcément par `.conf` : prendre la version du paquet déchargerait tous les sites d'un coup — sans erreur de syntaxe, sans avertissement, Apache démarrant parfaitement et ne servant rien.

Le remède est celui de tous les `.dpkg-dist` : comparer, garder le local, et retirer le fichier neuf une fois la comparaison faite.

Les types de contenu

Le même passage en revue révèle des écarts qui n'ont rien à voir avec la migration. Le fichier `/etc/mime.types` est un bon endroit où regarder : un type absent fait servir les fichiers correspondants en `application/octet-stream` — c'est-à-dire « octets bruts », ce que les navigateurs traitent comme un téléchargement plutôt que comme une image.

```
incus exec <conteneur> -- grep -nE "^image/webp" /etc/mime.types
```

Si la ligne manque, la poser après une copie de sûreté du fichier. Contenu de la ligne, tabulations comprises :

```
image/webp                webp
```

Compter les occurrences d'un mot ne prouve pas qu'un type est déclaré. Un `grep -c` lancé sur une liste de formats tombe sur des sous-chaînes — des noms de types qui contiennent le mot sans être lui. Ancrer le motif sur la structure de la ligne, comme ci-dessus, et non sur un mot qui peut apparaître ailleurs.

La preuve est du côté du service, pas du fichier : un `curl` sur un fichier de ce format doit rendre le bon `Content-Type`.

Rétablir les tâches nocturnes et éprouver la chaîne

Sur l’hôte :

```
crontab -e
```

Retirer les # posés à l’étape *Suspendre les tâches nocturnes*.

Sur l’hôte — la crontab interne, remise depuis sa copie :

```
incus exec web-srv -- bash -c \
  "crontab -u webadmin /root/crontab-webadmin-avant-migration.txt"
incus exec web-srv -- crontab -u webadmin -l
```

 Cette étape ne se saute pas. Une sauvegarde suspendue et jamais rétablie est exactement la panne silencieuse que toute cette procédure cherche à éviter — et personne ne la découvre avant d’en avoir besoin.

Puis éprouver la chaîne sans attendre la nuit, en déclenchant les tâches à la main dans l’ordre réel :

```
L=/var/log/backup/$(date +%F).log
D=/home/hostadmin/scripts/rsync/auto
incus exec web-srv -- su - webadmin -c \
  "bash /home/webadmin/scripts/backup_mysql.sh"
bash $D/backup_mysql-ns2.sh >> $L 2>&1
bash $D/backup_web-ns2.sh >> $L 2>&1
grep -E "^===" $L | tail -12
```

L’ordre n’est pas indifférent. Un script de sauvegarde qui porte le nom d’une base de données ne fabrique souvent rien : il tire ce que la machine distante a produit. Commencer par le tirage, c’est tester la seconde moitié de la chaîne en supposant la première.

⚠ Un bilan à `échecs : 0` mesure l'exécution, pas la destination. Si un chemin d'écriture était faux, rsync le créerait et réussirait tout aussi bien. Le contrôle qui tranche cherche le résultat là où il doit être, pas dans le journal — et c'est aussi vrai d'une migration de version : c'est la chaîne de sauvegarde qui est le point où une panne serait invisible, puisque la machine de sauvegarde continuerait de tirer fidèlement une sauvegarde qui vieillit.

Chercher un marqueur structurel plutôt qu'un mot. Un `grep` sur `erreur` ou `failed` dans un journal de sauvegarde rend surtout des noms de fichiers que rsync énumère en les copiant. Ce sont les marqueurs de début et de fin qui font foi.

Le ménage à froid

Rien de ce qui suit n'est urgent, et rien ne se fait le même jour. Le relevé de sécurité de la section précédente, lui, passe avant.

Les paquets orphelins — lire la liste avant d'appliquer, en écartant ce dont un logiciel hors d'apt pourrait dépendre :

```
incus exec <conteneur> -- apt-get -s autoremove --purge \
  | grep -E "^(Purg|Remv)"
```

Les paquets restés en état `rc` — désinstallés, configuration conservée — qu'`autoremove` ne touche pas :

```
incus exec <conteneur> -- bash -c "dpkg -l | awk '/^rc/ {print \$2}'"
```

Les résidus laissés par la migration : les fichiers `.distUpgrade` dans les sources apt, les `.dpkg-dist` comparés, les anciens fichiers de configuration réseau à suffixe. Les déplacer dans un répertoire daté plutôt que les supprimer — un résidu inerte se confond avec un fichier actif au moment d'un

dépannage, mais il peut encore servir à comprendre.

Les protections temporaires posées avant la migration : un `apt-mark manual` mis pour reporter une décision se défait par `apt-mark auto` le jour où on tranche.

Les instantanés

Ils se retirent après quelques jours d’usage réel, pas le soir même. Une migration qui passe tous les contrôles peut encore révéler, une semaine plus tard, un service qu’on n’utilise qu’une fois par mois.

```
incus snapshot list <conteneur>
incus snapshot delete <conteneur> avant-migration
```

 Le vrai critère n’est pas le temps écoulé, c’est la couverture d’usage : un instantané se garde tant que quelque chose d’important n’a pas encore été exercé au moins une fois depuis la migration.