

Serveur web – Apache multi-PHP et ISPConfig

Référence : Ubuntu 22.04 – PHP 8.3, 8.4 et 8.5

Guide de reconstruction. Il décrit comment rebâtir le conteneur qui héberge les sites, du système nu jusqu'aux sauvegardes qui en sortent.

Ce que cette page couvre : rebâtir de bout en bout un conteneur qui sert huit hôtes virtuels – deux applications PHP et quatre mandataires inverses – administrés par ISPConfig, avec trois versions de PHP disponibles simultanément, un serveur FTP, un serveur de calendriers et le démon rsync qui expose les données aux sauvegardes.

Contexte : un conteneur Incus sous Ubuntu Server 22.04.5, Apache 2.4, MariaDB 10.6, PHP 8.3 / 8.4 / 8.5 tirés de `packages.sury.org`, ISPConfig 3.3.1p1. Le conteneur porte l'adresse 192.168.0.6 sur le réseau domestique et monte un disque de l'hôte comme `/var/www`. Les adresses, noms d'hôtes et noms de domaine sont des exemples : les transposer à son propre parc.

Trois choses valent le détour même sans suivre le guide de bout en bout.

Le répertoire des sockets PHP-FPM porte le nom d'une version de PHP qui n'existe plus sur la machine, il est unique pour tout le serveur, et c'est délibéré – section *Les trois pièges du multi-PHP*.

Un dépôt de paquets qui a cessé de publier se comporte exactement comme un dépôt à jour : `apt update` réussit, `apt upgrade` n'a rien à proposer, et la machine cesse de recevoir ses correctifs sans qu'aucun message ne le dise – section *Le dépôt multi-PHP*.

Le fichier `php.ini` qui décide de la taille des envois n'est pas celui qu'on modifie spontanément. Sur une machine où tous les sites tournent en FPM, régler `/etc/php/8.3/apache2/php.ini` ne change rien – section *Où vivent réellement les*

limites.

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.

Ce que cette machine fait

Le conteneur `web-srv` est le serveur web du réseau. Il porte tout ce qui répond en HTTP, et rien d'autre.

Deux applications PHP, servies par PHP-FPM : le blogue WordPress et la galerie Piwigo.

Quatre mandataires inverses, sans PHP du tout : ils reçoivent la requête sur un nom public muni d'un certificat, et la relaient vers un service qui vit ailleurs – le serveur de calendriers du même conteneur, le correcteur linguistique, le serveur de musique, l'administration du routeur.

Un panneau d'administration, ISPConfig, qui écrit les hôtes virtuels Apache, les pools PHP-FPM, les comptes FTP et les bases de données. C'est lui qui fait autorité : un vhost modifié à la main est réécrit à la première régénération.

Un démon rsync qui expose en lecture les répertoires que l'hôte vient tirer chaque nuit.

S'y ajoutent, sans lien avec le web : Radicale pour les calendriers et carnets d'adresses, PureFTPd pour le dépôt de fichiers, et `memcached`.

⚠ Ce conteneur ne sauvegarde rien lui-même. Il produit ses vidages de bases dans un répertoire local, et c'est nas-host qui vient les chercher.

Reconstruire ce conteneur sans reconstruire les tirages de l'hôte donne un serveur qui fonctionne et qui n'est pas sauvegardé – voir **Sauvegarde**.

Le conteneur, vu de l'hôte

La création d'un conteneur Incus, les profils et le réseau macvlan sont décrits dans [INCUS – Hôte et conteneur & serveur NAS](#). Seules les particularités de celui-ci figurent ici.

Sur `[nas-host]` :

```
incus launch images:ubuntu/22.04 web-srv --profile default --profile macvlan
incus config set web-srv boot.autostart true
```

Sur `[nas-host]` – les limites de ressources :

```
incus config set web-srv limits.cpu.allowance 40%
incus config set web-srv limits.memory 6000MB
incus config set web-srv limits.memory.enforce soft
```

□□ `limits.memory.enforce soft` laisse le conteneur dépasser sa limite quand l'hôte a de la mémoire libre, plutôt que de faire tuer un processus. Sur un serveur de bases de données, c'est le réglage qui évite qu'un pic d'indexation se termine par un `mariadb` abattu.

Sur `[nas-host]` – le disque dédié aux sites, désigné par son identifiant matériel :

```
incus config device add web-srv web disk \
    source=/dev/disk/by-id/scsi-SATA_DISK-SERIAL-part2 \
    path=/var/www
```

△□ Jamais `/dev/sdX`. Ces noms dépendent de l'ordre de détection des disques au démarrage : le jour où un disque est ajouté ou retiré, le conteneur monte le mauvais.

△ Le conteneur est non privilégié, et le disque exige un décalage d'identifiants. Ce que le conteneur voit comme l'utilisateur 0 est l'utilisateur 1000000 sur l'hôte. Un périphérique bloc n'est pas concerné par les montages idmappés du noyau – cette fonction ne s'applique qu'aux sources de type répertoire –, il faut donc décaler les propriétaires du disque une fois pour toutes. La procédure, ses pièges et sa forme rejouable sont décrits dans l'article de l'hôte.

Sur `[nas-host]` – vérifier la carte d'identifiants réellement en vigueur :

```
incus config get web-srv volatile.idmap.current
```

Préparer le système

Neutraliser cloud-init

Sur `[nas-host]` :

```
incus exec web-srv -- touch /etc/cloud/cloud-init.disabled
```

Sur `[nas-host]` :

```
echo "network: {config: disabled}" | incus exec web-srv -- \
tee /etc/cloud/cloud.cfg.d/99-disable-network-config.cfg
```

Pourquoi. cloud-init réécrit la configuration réseau quand il croit voir une instance neuve, et il le fait au démarrage, en silence. Sur un conteneur dont l'adresse est fixe et dont six noms de domaine dépendent, c'est exactement ce qu'on ne veut pas.

Ouvrir un shell dans le conteneur

Sur `[nas-host]` :

```
incus exec web-srv -- bash
```

Toutes les commandes qui suivent portent la marque `[dans web-srv]` et s'exécutent en root – c'est ce que donne `incus exec`, il n'y a pas de `sudo` à ajouter.

L'adresse fixe

`[dans web-srv]` :

```
nano /etc/netplan/50-static-public-ip.yaml
```

Contenu :

```
network:
  version: 2
  renderer: networkd
  ethernets:
    eth0:
      dhcp4: no
      dhcp6: no
      addresses: [192.168.0.6/24]
      nameservers:
        addresses:
          - 192.168.0.5
          - 208.67.222.222
      routes:
        - to: default
          via: 192.168.0.1
```

`[dans web-srv]` :

```
chmod 600 /etc/netplan/50-static-public-ip.yaml
```

```
netplan apply
ip -br addr show eth0
```

□□ Le second résolveur n'est pas décoratif. Le premier, 192.168.0.5, est le Pi-hole, qui tourne dans un conteneur voisin sur la même machine. Sans résolveur de secours, l'arrêter priverait ce serveur de résolution de noms – y compris pour aller chercher les paquets nécessaires à sa réparation.

△□ Netplan ne lit que les fichiers en `.yaml`. Un fichier renommé `.yaml.old` pour le mettre de côté est ignoré, ce qui est le comportement voulu – mais il reste lisible et se confond avec le fichier actif lors d'un dépannage. Le ranger ailleurs que dans `/etc/netplan/` vaut mieux que le renommer sur place.

Nom d'hôte

[dans web-srv] :

```
nano /etc/hosts
```

```
127.0.0.1 web-srv.example.com web-srv localhost
192.168.0.6 web-srv.example.com web-srv

# 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
```

[dans web-srv] :

```
echo web-srv > /etc/hostname
hostname web-srv
hostname -f
```

`hostname -f` doit répondre `web-srv.example.com`. L'installateur d'ISPConfig s'en sert pour nommer le serveur dans sa base : une réponse vide ou fausse à cette étape se paie plus tard.

Fuseau horaire

[dans web-srv] :

```
timedatectl set-timezone America/Toronto
timedatectl
```

⚠ Les images de conteneur arrivent en `Etc/UTC`. Sans ce réglage, les journaux d'Apache portent une heure décalée de quatre heures par rapport à ceux de l'hôte, ce qui rend toute corrélation d'incident pénible. Et la tâche cron de vidage des bases s'exécuterait à un tout autre moment que celui qu'on croit avoir programmé.

L'interpréteur par défaut

[dans web-srv] :

```
dpkg-reconfigure dash
```

Répondre No à « Use dash as the default system shell (/bin/sh)? ».

⚠ Cette étape n'est pas cosmétique : l'installation d'ISPConfig échoue sans elle. Ses scripts emploient des constructions propres à bash tout en s'annonçant en `/bin/sh`.

[dans web-srv] – vérifier :

```
ls -l /bin/sh
```

Le lien doit pointer vers `bash`.

AppArmor

`[dans web-srv]` :

```
systemctl stop apparmor
apt remove -y apparmor apparmor-utils
```

□□ Dans un conteneur non privilégié, les profils AppArmor sont de toute façon appliqués par l'hôte : ceux de l'intérieur ne font que gêner MariaDB et PureFTPd sans rien apporter.

Mise à jour et horloge

`[dans web-srv]` :

```
apt update
apt -y full-upgrade
apt -y install ntp
```

Le dépôt multi-PHP — `packages.sury.org`

Ubuntu 22.04 ne livre qu'une version de PHP. Les trois versions simultanées viennent d'un dépôt tiers.

△□ Ne pas employer le PPA `ondrej/php`. Il a été vidé de ses paquets au profit de `packages.sury.org`, et il annonçait ce déménagement dans le champ de version de chacun de ses paquets — mention que personne ne lit. Une machine restée sur ce PPA continue de répondre à `apt update` sans erreur, propose zéro mise à jour, et cesse simplement de recevoir les correctifs de sécurité de PHP. C'est la même famille de panne qu'un service qui démarre sans rien faire : l'échec se déclare en succès.

La clé

[dans web-srv] :

```
curl -fsSL https://packages.sury.org/php/apt.gpg \  
-o /usr/share/keyrings/deb.sury.org-php.gpg
```

[dans web-srv] – vérifier l’empreinte avant de s’en servir :

```
gpg --show-keys --fingerprint /usr/share/keyrings/deb.sury.org-php.gpg
```

Empreinte attendue :

```
1505 8500 A023 5D97 F5D1 0063 B188 E2B6 95BD 4743
```

☐☐ Cette clé a déjà expiré une fois. La copie installée en 2021 portait une date de validité dépassée, ce qui fait échouer `apt update` avec un `EXPKEYSIG` alors que la clé, elle, est toujours la bonne : c’est la même, dont la validité a été prolongée jusqu’au 4 février 2028. Le correctif est de retélécharger le fichier, pas de chercher une autre clé.

La source

[dans web-srv] :

```
nano /etc/apt/sources.list.d/php.list
```

Une seule ligne :

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

△☐ Cette ligne ne se coupe pas – une entrée de source apt tient sur une seule

ligne, sans continuation possible. Elle s'enroule donc dans le PDF sans signe visible : la recopier depuis le fichier `.md`, jamais depuis le PDF.

△ `signed-by` restreint la clé à ce dépôt, et c'est tout l'intérêt. Une clé déposée dans `/etc/apt/trusted.gpg.d/` vaut pour toutes les sources de la machine : le jour où l'un de ces dépôts est compromis, il peut signer n'importe quel paquet, y compris un remplacement de `openssh-server`. C'est la forme à retenir pour tout dépôt tiers.

△ Une clé retirée d'usage doit être retirée tout court. Désactiver une source en commentant sa ligne ne supprime pas sa clé de `trusted.gpg.d/`, qui continue d'accorder sa confiance à l'échelle de la machine. Sur une reprise d'installation, vérifier ce répertoire.

`[dans web-srv]` :

```
apt update
apt policy php8.3-fpm
```

Le candidat doit venir de `packages.sury.org`. Prendre l'habitude de `apt policy` avant toute installation depuis un dépôt tiers : c'est le seul moyen de savoir d'où viendra réellement un paquet.

MariaDB

`[dans web-srv]` :

```
apt -y install mariadb-server mariadb-client openssl binutils quota quotatool
```

Le mot de passe root

MariaDB s'installe sans mot de passe root et n'en réclame pas. ISPConfig et phpMyAdmin, eux, en exigent un.

[dans web-srv] :

```
mysql
```

Dans l'invite de MariaDB :

```
ALTER USER root@localhost IDENTIFIED VIA mysql_native_password
  USING PASSWORD("<mot-de-passe>");
\q
```

[dans web-srv] – vérifier que la connexion sans mot de passe est maintenant refusée :

```
mysql
```

N'écouter que la boucle locale

[dans web-srv] :

```
nano /etc/mysql/mariadb.conf.d/50-server.cnf
```

La ligne à trouver, et à laisser active :

```
bind-address          = 127.0.0.1
```

⚠ Les tutoriels d'ISPConfig demandent de commenter cette ligne. Ne pas le faire sur un serveur unique. Ce conseil vise une installation à plusieurs serveurs, où un nœud interroge la base d'un autre. Ici, tout ce qui touche la base est local : le panneau, phpMyAdmin, les deux applications PHP et le script de vidage. Laisser MariaDB écouter sur `0.0.0.0` expose le port 3306 à tout le réseau – et sur un conteneur en macvlan, « tout le réseau » veut dire chaque appareil de la maison.

Réglages

[dans web-srv] :

```
nano /etc/mysql/my.cnf
```

À ajouter en fin de fichier :

```
max_allowed_packet = 128M

[mysqld]
wait_timeout = 86400
interactive_timeout = 86400
innodb_log_file_size = 128MB
```

[dans web-srv] :

```
nano /etc/security/limits.conf
```

À ajouter en fin de fichier :

```
mysql soft nfile 65535
mysql hard nfile 65535
```

[dans web-srv] :

```
mkdir -p /etc/systemd/system/mysql.service.d/
```

[dans web-srv] :

```
nano /etc/systemd/system/mysql.service.d/limits.conf
```

```
[Service]
LimitNOFILE=infinity
```

[dans web-srv] :

```
systemctl daemon-reload
systemctl restart mariadb
ss -ltn | grep 3306
```

La seule ligne attendue montre `127.0.0.1:3306`.

Apache, PHP 8.3 et phpMyAdmin

Installation

[dans web-srv] – Apache et ses compléments :

```
apt -y install apache2 apache2-doc apache2-utils apache2-suexec-pristine \
  libapache2-mod-fcgid libapache2-reload-perl libapache2-mod-python \
  libruby memcached
```

[dans web-srv] – PHP 8.3, jeu d’extensions complet :

```
apt -y install php8.3 php8.3-common php8.3-cli php8.3-cgi php8.3-fpm \
  php8.3-bz2 php8.3-curl php8.3-gd php8.3-imagick php8.3-imap \
  php8.3-intl php8.3-mbstring php8.3-mysql php8.3-openssl \
  php8.3-pspell php8.3-readline php8.3-soap php8.3-sqlite3 \
  php8.3-tidy php8.3-xml php8.3-xmlrpc php8.3-xsl php8.3-zip \
  libapache2-mod-php8.3 php-pear php-memcache
```

[dans web-srv] – phpMyAdmin :

```
apt -y install phpmyadmin
```

Réponses attendues :

```
Web server to reconfigure automatically:      <-- apache2
Configure database for phpmyadmin with dbconfig-common? <-- Yes
MySQL application password for phpmyadmin:    <-- vide
Password of the database's administrative user: <-- mot de passe root
```

Les modules

[dans web-srv] :

```
a2enmod suexec rewrite ssl actions include cgi alias proxy proxy_fcgi
```

[dans web-srv] :

```
a2enmod proxy_http proxy_html xml2enc dav dav_fs auth_digest headers deflate
```

Module	Ce qui en dépend
suexec	l'exécution des scripts sous l'identité du site
proxy_fcgi	le dialogue avec les pools PHP-FPM
proxy_http, proxy_html, xml2enc	les quatre mandataires inverses
dav, dav_fs, auth_digest	le partage WebDAV
actions, cgi, alias	les gabarits d'hôtes virtuels d'ISPConfig

Fermer HTTPXY

[dans web-srv] :

```
nano /etc/apache2/conf-available/httpoxy.conf
```

```
<IfModule mod_headers.c>
RequestHeader unset Proxy early
</IfModule>
```

[dans web-srv] :

```
a2enconf httpoxy
```

Pourquoi. Un en-tête `Proxy:` envoyé par un client se retrouve dans la variable d'environnement `HTTP_PROXY` du script CGI, que beaucoup de bibliothèques lisent pour router leurs propres requêtes sortantes. Un visiteur peut ainsi détourner le trafic que le serveur émet.

Les fichiers Ruby

[dans web-srv] :

```
nano /etc/mime.types
```

Mettre la ligne en commentaire :

```
#application/x-ruby          rb
```

Sans quoi Apache sert les fichiers `.rb` comme un type MIME au lieu de les exécuter.

Discretion du serveur

[dans web-srv] :

```
nano /etc/apache2/conf-available/security.conf
```

Les deux lignes à modifier :

```
ServerTokens Prod
ServerSignature Off
```

△ □ Le défaut d'Ubuntu est bavard. `ServerTokens OS` fait annoncer « Apache/2.4.52 (Ubuntu) » dans chaque réponse HTTP, et `ServerSignature On` répète la même information en pied de chaque page d'erreur. Ce n'est pas une faille, mais c'est le renseignement qui permet à un balayage automatisé de ne retenir que les serveurs dont la version l'intéresse.

Chercher les alias globaux

C'est le contrôle qu'on oublie de faire. Un `Alias` déclaré dans `conf-enabled/` ne vaut pas pour un site : il vaut pour tous, y compris ceux qui répondent sur Internet.

`[dans web-srv]` :

```
grep -R -niE '^[[:space:]]*(Alias|ScriptAlias)' /etc/apache2/conf-enabled/
```

△ □ `grep -R` et non `-r`. Le répertoire `conf-enabled/` ne contient que des liens symboliques, et `-r` ne les suit pas : la recherche renvoie zéro résultat et donne l'impression rassurante qu'il n'y a rien à voir.

Trois alias sont attendus, et un seul est souhaitable :

Alias	Origine	Sort
<code>/phpmyadmin</code>	<code>phpmyadmin.conf</code>	conservé, mais restreint – voir plus bas
<code>/cgi-bin/</code>	<code>serve-cgi-bin.conf</code>	conservé, employé par les gabarits
<code>/manual</code>	<code>apache2-doc.conf</code>	désactivé

`[dans web-srv]` :

```
a2disconf apache2-doc
```

△ □ `apache2-doc.conf` publie la documentation d'Apache sur tous les sites, avec `Require all granted` et `Options Indexes`. Le paquet est utile pour consulter la documentation localement ; sa configuration Apache, elle, expose l'arborescence à qui visite n'importe lequel des noms hébergés. Rien de grave, mais c'est un répertoire de plus à servir, indexable, et qui annonce la version du serveur par ricochet.

Restreindre phpMyAdmin

phpMyAdmin est un panneau d'administration de bases de données servi sur tous les hôtes virtuels du serveur, donc joignable depuis Internet dès qu'un site l'est. Il ne doit répondre qu'au réseau local et au VPN.

[dans web-srv] :

```
nano /etc/apache2/conf-available/zzz-phpmyadmin-restrict.conf
```

```
<Location /phpmyadmin>
    Require ip 192.168.0.0/24
    Require ip 10.6.0.0/24
</Location>
```

[dans web-srv] :

```
a2enconf zzz-phpmyadmin-restrict
```

□ □ Le préfixe `zzz-` est ce qui fait fonctionner la restriction. Apache charge les fichiers de `conf-enabled/` dans l'ordre alphabétique, et une directive `Require` posée avant `phpmyadmin.conf` serait écrasée par ce que celui-ci déclare ensuite. Le nom porte donc une contrainte technique, pas une convention de rangement.

Limites d'envoi de phpMyAdmin

[dans web-srv] :

```
nano /etc/php/8.3/apache2/php.ini
```

```
memory_limit = 512M  
upload_max_filesize = 10G  
post_max_size = 10G
```

△ [] Ce fichier ne concerne QUE phpMyAdmin et les autres applications servies par `mod_php`. Aucun des sites hébergés ne le lit – ils sont tous en FPM. Voir la section suivante, qui est celle qui compte.

△ [] Ces valeurs s'ajoutent en fin de fichier plutôt qu'à l'endroit où la directive figure déjà. Un `grep` sur `post_max_size` renvoie alors deux lignes avec deux valeurs différentes : c'est la dernière lue qui s'applique. Le savoir évite un diagnostic faux au moment où l'on cherche pourquoi une limite n'est pas celle qu'on croit.

[dans web-srv] :

```
apache2ctl configtest
```

[dans web-srv] :

```
systemctl restart apache2
```

Les versions supplémentaires de PHP

Le serveur porte trois versions : 8.3, qui sert les deux sites et fait le

défaut système, 8.4 et 8.5, déclarées et prêtes à recevoir un site.

[dans web-srv] – PHP 8.4 :

```
apt -y install php8.4 php8.4-common php8.4-cli php8.4-cgi php8.4-fpm \
  php8.4-bz2 php8.4-curl php8.4-gd php8.4-imagick php8.4-imap \
  php8.4-intl php8.4-mbstring php8.4-mysql php8.4-openssl \
  php8.4-pspell php8.4-readline php8.4-soap php8.4-sqlite3 \
  php8.4-tidy php8.4-xml php8.4-xmlrpc php8.4-xsl php8.4-zip
```

[dans web-srv] – PHP 8.5 :

```
apt -y install php8.5 php8.5-common php8.5-cli php8.5-cgi php8.5-fpm \
  php8.5-bz2 php8.5-curl php8.5-gd php8.5-imagick php8.5-imap \
  php8.5-intl php8.5-mbstring php8.5-mysql \
  php8.5-pspell php8.5-readline php8.5-soap php8.5-sqlite3 \
  php8.5-tidy php8.5-xml php8.5-xmlrpc php8.5-xsl php8.5-zip
```

□□ `php8.5-openssl` n'existe pas, et ce n'est pas une lacune du dépôt :
l'extension est intégrée autrement à partir de la 8.5. Le vérifier plutôt que
le supposer :

```
php8.5 -m | grep -i "zend openssl"
```

□□ L'installation d'une version supplémentaire ne bascule rien. Le module
Apache reste celui qui était chargé – apt le dit d'ailleurs, « php8.3 module
already enabled, not enabling PHP 8.4 » –, les alternatives de la ligne de
commande ne bougent pas, et aucun site ne change de version. C'est ISPConfig
qui décide, site par site.

Une opération de masse qui refuse d'agir

Sur un serveur en production, une installation de vingt paquets peut en retirer
d'autres sans qu'on l'ait vu passer. Ce bloc simule d'abord et ne s'exécute que
si rien ne doit être retiré :

```
if apt-get install -s $L 2>&1 | grep -q "^Remv" ; then
    echo "ARRÊT – apt voudrait retirer des paquets :"
    apt-get install -s $L 2>&1 | grep "^Remv"
else
    apt-get install -y $L
fi
```

⚠ Pour une purge, le motif de recherche n'est pas le même. `apt-get purge -s` écrit `Purg`, pas `Remv`. Un motif erroné fait croire qu'aucun paquet ne partira.

Les trois pièges du multi-PHP sous ISPConfig

Ce sont eux qui coûtent le plus de temps, et aucun ne se devine.

1. Le défaut de la ligne de commande n'est pas le défaut des sites

`php --version` peut répondre 8.3 pendant que tout site réglé sur « défaut système » reçoit une autre version. Ce sont deux réglages sans rapport :

- la ligne de commande suit l'alternative `update-alternatives` ;
- les sites suivent les chemins inscrits dans Server Config → Web d'ISPConfig, plus le module chargé par Apache.

`[dans web-srv]` – les deux se vérifient séparément :

```
update-alternatives --display php | head -4
apache2ctl -M | grep -i php
```

2. Le répertoire des sockets porte un nom trompeur, et il est global

Le pool d'un site en PHP 8.3 vit sous `/etc/php/8.3/fpm/pool.d/` mais écoute dans `/var/lib/php8.1-fpm/`.

Ce n'est pas une incohérence. Le champ `php_fpm_socket_dir` de chaque version déclarée est vide – y compris pour celle qui fonctionne –, et retombe donc sur un réglage unique, au niveau du serveur :

```
php_fpm_socket_dir=/var/lib/php8.1-fpm
```

Il a été fixé à l'installation d'après la version qui était alors le défaut, et il ne suit pas les versions. Tous les pools y écrivent, quelle que soit la leur.

⚠ Ce répertoire n'appartient à aucun paquet – `dpkg -S` ne lui trouve pas de propriétaire, c'est ISPConfig qui le crée. C'est ce qui rend la purge de l'ancienne version inoffensive : les sockets survivent. C'est la question qui décide de tout l'ordre des opérations lors d'un changement de version : si ce répertoire appartenait au paquet, sa purge couperait tous les sites d'un coup.

☐ Le renommer coûte plus cher qu'il ne rapporte. Changer `php_fpm_socket_dir` oblige à régénérer tous les sites en même temps – le seul geste qui les touche tous simultanément. Le nom reste trompeur ; c'est un défaut de lisibilité, pas de fonctionnement.

3. Deux champs du formulaire ne sont pas des chemins

Dans System → Additional PHP Versions, deux champs attendent un nom et non un chemin :

Champ	Attendu	Erreur classique
Path to the PHP FastCGI binary	<code>php-cgi8.4</code>	<code>/usr/bin/php-cgi8.4</code>
Path to the PHP-FPM init script	<code>php8.4-fpm</code>	<code>/etc/init.d/php8.4-fpm</code>

Les autres champs de la même fiche :

PHP Name	PHP 8.4
Path to the php.ini directory	/etc/php/8.4/cgi (onglet FastCGI)
Path to the php.ini directory	/etc/php/8.4/fpm (onglet PHP-FPM)
Path to the PHP-FPM pool directory	/etc/php/8.4/fpm/pool.d
PHP-FPM socket directory	(vide)
Client	(vide)

☐☐ `FastCGI php.ini Path` est sur un onglet distinct de celui du FPM. Les deux champs portent le même libellé et attendent deux valeurs différentes.

☐☐ Laisser `Client` vide. Une version associée à un client ne serait proposée qu'à lui – ce qui est le comportement voulu chez un hébergeur, pas chez soi.

☐☐ Prendre modèle sur une entrée existante plutôt que de supposer. C'est ce qui départage un nom de binaire d'un chemin absolu sans avoir à essayer.

Où vivent réellement les limites d'envoi

C'est la section à lire avant de chercher pourquoi un envoi échoue.

Sur cette machine, tous les sites tournent en PHP-FPM. La chaîne des valeurs est donc :

/etc/php/8.3/fpm/php.ini	le défaut, pour tout site sans réglage propre
↓ surchargé par	
/etc/php/8.3/fpm/pool.d/webN.conf	écrit par ISPConfig, un fichier par site

△☐ `/etc/php/8.3/apache2/php.ini` n'intervient pas. C'est pourtant le fichier qu'on modifie spontanément, et c'est celui que décrivent la plupart des guides – parce qu'ils datent d'une époque où les sites passaient par `mod_php`. Le régler

à 10 Go ne change rien au site.

△ Le `php.ini` du FPM est resté au défaut de la distribution :

`upload_max_filesize = 2M`, `post_max_size = 8M`, `memory_limit = 128M`. Un site pour lequel aucune valeur n'a été posée dans le panneau hérite donc de 2 Mio d'envoi maximum – quelle que soit la valeur, généreuse, inscrite ailleurs.

On ne modifie pas le pool à la main : ISPConfig le réécrit. Les valeurs se posent dans Sites → le site → onglet Options → Custom php.ini settings.

Pour le blogue WordPress :

```
upload_max_filesize = 64M
post_max_size = 64M
memory_limit = 256M
max_execution_time = 120
```

Pour la galerie Piwigo, dont les fichiers sont plus gros et le traitement plus lourd :

```
upload_max_filesize = 200M
post_max_size = 200M
memory_limit = 400M
max_execution_time = 300
```

Le raisonnement derrière ces nombres :

- `memory_limit` doit dépasser nettement la taille des fichiers traités, parce que le redimensionnement d'une image consomme bien davantage que le poids du fichier. Le rapport d'environ deux pour un retenu pour la galerie n'est pas arbitraire.
- 256 Mio est ce que l'administration de WordPress vise elle-même pour ses mises à jour et son traitement d'images ; 128 Mio la laisse échouer sur les grosses opérations.
- `post_max_size` doit rester supérieur ou égal à `upload_max_filesize`, sinon la

limite affichée n'est pas celle qui s'applique.

- `post_max_size` plafonne l'envoi entier, pas chaque fichier. Pour un dépôt par lots – une galerie, typiquement – c'est le lot qui bloque en premier, jamais le fichier isolé.

[dans web-srv] – vérifier ce qui a réellement été écrit, une minute après avoir enregistré :

```
grep -nE 'php_admin_value' /etc/php/8.3/fpm/pool.d/web1.conf
```

☐☐ ISPConfig n'écrit pas au moment où l'on clique : c'est sa tâche cron, qui passe chaque minute, qui régénère le pool et recharge `php8.3-fpm`.

ISPConfig

Installation

[dans web-srv] :

```
cd /tmp
wget -O ispconfig.tar.gz \
  https://www.ispconfig.org/downloads/ISPConfig-3-stable.tar.gz
tar xzf ispconfig.tar.gz
```

[dans web-srv] :

```
cd /tmp/ispconfig3*/install/
php -q install.php
```

Réponses :

Select language (en,de) [en]:	<ENTRÉE>
Installation mode (standard,expert) [standard]:	<ENTRÉE>
Full qualified hostname (FQDN):	web-srv.example.com
MySQL server hostname [localhost]:	<ENTRÉE>
MySQL server port [3306]:	<ENTRÉE>
MySQL root username [root]:	<ENTRÉE>
MySQL root password []:	<mot-de-passe>
MySQL database to create [dbispconfig]:	<ENTRÉE>
MySQL charset [utf8]:	<ENTRÉE>
ISPConfig Port [8080]:	<ENTRÉE>
Admin password [admin]:	<mot-de-passe>
Do you want a secure (SSL) connection (y,n) [y]:	<ENTRÉE>

☐☐ La liste des questions varie d’une version d’ISPConfig à l’autre, et de nouvelles apparaissent au fil des versions. Deux seulement demandent de la réflexion : le nom pleinement qualifié du serveur et le mot de passe root de MariaDB. Pour toutes les autres, la valeur entre crochets est la bonne sur un serveur unique.

Le certificat auto-signé du panneau se crée à ce moment-là. Les champs demandés :

Country Name (2 letter code)	CA
State or Province Name	QC
Locality Name	Ville
Organization Name	webadmin
Organizational Unit Name	IT
Common Name	example.com
Email Address	postmaster@example.com

Le panneau

Le panneau répond sur <https://192.168.0.6:8080/>. Un second hôte virtuel, celui des applications d’ISPConfig, écoute sur le port 8081 et partage le même certificat.

Le certificat du panneau est auto-signé, émis pour dix ans à l’installation. Le

navigateur avertit à chaque première visite, et c'est normal : ce certificat n'est vu que depuis le réseau local, il n'a rien à prouver à une autorité publique.

□□ Le remplacer par le certificat commercial des sites est possible et purement cosmétique. Cela supprime l'avertissement du navigateur ; cela n'ajoute aucune protection, le panneau n'étant pas exposé sur Internet. Et cela crée une obligation nouvelle : penser à recopier le certificat dans `/usr/local/ispconfig/interface/ssl/` à chaque renouvellement, sans quoi c'est le panneau qui tombe en erreur. Le choix retenu ici est de garder l'auto-signé.

△□ Le certificat du panneau sert aussi à PureFTPd – `/etc/ssl/private/pure-ftpd.pem` est un lien symbolique vers `ispserver.pem`. Le remplacer touche donc deux services, pas un.

Mise à jour

`[dans web-srv]`, après avoir pris un instantané depuis l'hôte :

```
ispconfig_update.sh
```

Réponses pour un serveur unique :

Select update method	stable
Create full backup before update	yes
Reconfigure permissions in master database	no
Reconfigure services	yes
Create new ISPConfig SSL certificate	no
Reconfigure crontab	yes

△□ `Reconfigure services = yes` réécrit tous les hôtes virtuels à partir des gabarits de la nouvelle version. C'est ce qu'on veut – c'est ce qui propage les réglages du panneau – mais c'est aussi le moment où une modification faite à la main dans un vhost disparaît.

Sur `[nas-host]` – l’instantané à prendre avant, jamais après :

```
incus snapshot create web-srv avant-ispconfig
```

Deux messages sont attendus pendant l’exécution, sans conséquence :

```
chattr: Operation not permitted while setting flags on  
/var/www/php-fcgi-scripts/ispconfig/.php-fcgi-starter  
mysqlcheck: unknown variable 'max_allowed_packet=128M'
```

□□ Le premier est la limite d’un conteneur non privilégié : `chattr +i` y est interdit. ISPConfig pose cet attribut par précaution et poursuit sans lui.

Les certificats des sites

Les six sites actifs partagent un certificat générique `*.example.com`, émis par une autorité commerciale, valable six mois, déposé site par site dans le panneau.

Emplacement, pour un site donné :

```
/var/www/clients/client1/web1/ssl/blog.example.com.crt  
/var/www/clients/client1/web1/ssl/blog.example.com.key
```

Le dépôt se fait dans le panneau, jamais dans ces fichiers : Sites → le site → onglet SSL, en collant le certificat et sa clé, action Save certificate. ISPConfig écrit les fichiers et recharge Apache.

△□ Rien sur cette machine ne renouvelle ce certificat. `certbot` est installé mais ne détient aucun certificat, et l’agent ACME d’ISPConfig n’a aucun dossier. Le renouvellement est un geste manuel, à faire avant la date d’expiration, sur les six sites. C’est le genre d’échéance qui ne se rappelle

pas toute seule : elle appartient à un calendrier, pas à un serveur.

`[dans web-srv]` – lire la date d’expiration réellement en vigueur :

```
openssl x509 -noout -subject -dates -ext subjectAltName \
-in /var/www/clients/client1/web1/ssl/blog.example.com.crt
```

□□ Le mécanisme Let’s Encrypt d’ISPConfig reste en place et fonctionnel – l’alias qui sert les défis de validation est déclaré dans `999-acme.conf` :

```
Alias /.well-known/acme-challenge \
/usr/local/ispconfig/interface/acme/.well-known/acme-challenge
```

Il suffit de cocher Let’s Encrypt SSL sur un site pour qu’ISPConfig prenne en charge l’émission et le renouvellement. C’est la voie à privilégier pour tout nouveau site, et le remplacement naturel du certificat commercial le jour où celui-ci ne sera pas renouvelé.

Les sites hébergés

Site	Actif	Mode	PHP Version	Ce qu’il sert
<code>blog.example.com</code>	☐	php-fpm	8.3	WordPress
<code>photos.example.com</code>	☐	php-fpm	8.3	Piwigo
<code>dav.example.com</code>	☐	aucun	–	mandataire vers Radicale
<code>lt.example.com</code>	☐	aucun	–	mandataire vers LanguageTool
<code>musique.example.com</code>	☐	aucun	–	mandataire vers Navidrome
<code>routeur.example.com</code>	☐	aucun	–	mandataire vers le routeur
<code>cumulus.example.com</code>	☐	mod	défaut	désactivé
<code>ha.example.com</code>	☐	aucun	–	désactivé

□□ Les deux sites désactivés sont ce qui rend un changement de défaut système inoffensif. Seuls les sites en « défaut système » suivent la version globale, et aucun de ceux qui tournent n’en fait partie : les deux applications ont une version nommée, les quatre mandataires n’exécutent pas de PHP du tout.

Les mandataires inverses

Chacun se déclare dans Sites → le site → onglet Redirect, avec le type proxy et l'adresse de destination. Les quatre destinations en service :

```
dav.example.com      → http://localhost:5232/  
lt.example.com       → http://192.168.0.10:8084/v2/  
musique.example.com → http://192.168.0.8:4533/  
routeur.example.com → https://192.168.0.1:8443/
```

Deux portent en plus une directive propre, ajoutée dans le champ Apache Directives de l'onglet Options :

```
ProxyPassReverseCookieDomain 192.168.0.1 routeur.example.com
```

```
ProxyPreserveHost On
```

□□ `ProxyPassReverseCookieDomain` réécrit le domaine des témoins émis par le service d'origine. Sans lui, le routeur pose ses témoins sur `192.168.0.1`, que le navigateur refuse de renvoyer à `routeur.example.com` – et l'authentification tourne en rond sans message d'erreur.

□□ `ProxyPreserveHost On` transmet le nom demandé par le visiteur au lieu du nom de la destination. Navidrome en a besoin pour construire ses propres liens ; les trois autres services s'en passent.

△□ Un mandataire vers une machine du réseau ne doit pas devenir une porte d'entrée. `routeur.example.com` publie l'administration du routeur derrière un nom public : sa protection ne tient qu'à l'authentification exigée en amont. C'est un montage à n'exposer qu'accompagné d'une restriction d'accès – et à revoir si l'accès distant passe désormais par un VPN, auquel cas le nom public n'a plus de raison d'être.

Radicale

Le serveur de calendriers et de carnets d'adresses tourne dans ce conteneur, lié à `127.0.0.1:5232`, et n'est joignable que par le mandataire ci-dessus. Son installation et sa configuration font l'objet de leur propre page : [Radical – CARDAV & CALDAV](#).

PureFTPd

`[dans web-srv]` :

```
apt -y install pure-ftpd-common pure-ftpd-mysql
```

`[dans web-srv]` :

```
nano /etc/default/pure-ftpd-common
```

Les deux réglages qui comptent :

```
STANDALONE_OR_INETD=standalone  
VIRTUALCHROOT=true
```

△ `VIRTUALCHROOT=true` est requis par ISPConfig. Il permet au confinement de suivre les liens symboliques, ce dont dépendent les répertoires des sites.

`[dans web-srv]` – activer TLS :

```
echo 1 > /etc/pure-ftpd/conf/TLS  
echo HIGH > /etc/pure-ftpd/conf/TLSCipherSuite
```

△ `TLS` à `1` autorise le clair ET le chiffré, en laissant le client choisir. La

valeur `2` refuse toute session non chiffrée. La première est retenue ici pour la compatibilité des clients anciens ; sur un serveur atteignable depuis Internet, la seconde s'impose – FTP transmet autrement les mots de passe en clair.

Les autres réglages en service, un fichier par valeur dans `/etc/pure-ftpd/conf/` :

<code>ChrootEveryone</code>	<code>yes</code>
<code>NoAnonymous</code>	<code>yes</code>
<code>UnixAuthentication</code>	<code>no</code>
<code>PAMAuthentication</code>	<code>yes</code>
<code>MinUID</code>	<code>1000</code>
<code>DisplayDotFiles</code>	<code>yes</code>
<code>DontResolve</code>	<code>yes</code>
<code>BrokenClientsCompatibility</code>	<code>yes</code>
<code>FSCharset</code>	<code>UTF-8</code>
<code>AltLog</code>	<code>clf:/var/log/pure-ftpd/transfer.log</code>
<code>MySQLConfigFile</code>	<code>/etc/pure-ftpd/db/mysql.conf</code>
<code>PureDB</code>	<code>/etc/pure-ftpd/pureftpd.pdb</code>

□□ `UnixAuthentication no` et `MinUID 1000` ensemble, c'est ce qui interdit à un compte du système d'ouvrir une session FTP : seuls les comptes virtuels créés dans ISPConfig sont acceptés.

□□ `DontResolve yes` n'est pas un réglage de confidentialité : il évite que chaque connexion attende la résolution inverse de l'adresse du client, attente qui se paie en secondes quand le DNS ne répond pas.

`[dans web-srv]` – le certificat, partagé avec le panneau :

```
ln -sf /usr/local/ispconfig/interface/ssl/ispserver.pem \  
    /etc/ssl/private/pure-ftpd.pem  
systemctl restart pure-ftpd-mysql
```

Le démon rsync

Le conteneur n'envoie rien : il expose ses répertoires en lecture, et nas-host vient les tirer chaque nuit sur le port 873.

Le fichier de configuration

[dans web-srv] :

```
nano /etc/rsyncd.conf
```

Le fichier complet, treize modules :

```
log file = /var/log/rsync.log

[blog]
uid = web1
gid = client1
path = /var/www/clients/client1/web1/web
hosts allow = 192.168.0.11
comment = Backup blog.example.com
read only = true
auth users = test
secrets file = /etc/rsyncd.scr

[webdav]
uid = web1
gid = client1
path = /var/www/clients/client1/web1/webdav/laflaf
hosts allow = 192.168.0.11
comment = Backup blog.example.com/webdav
read only = true
auth users = test
secrets file = /etc/rsyncd.scr

[photos]
uid = web2
gid = client1
```

```
path = /var/www/clients/client1/web2/web
hosts allow = 192.168.0.11
comment = Backup photos.example.com
read only = true
auth users = test
secrets file = /etc/rsyncd.scr
```

```
[gallerie]
uid = web2
gid = client1
path = /var/www/clients/client1/web2/web/galleries
hosts allow = 192.168.0.11
comment = Upload photos.example.com
read only = false
auth users = test
secrets file = /etc/rsyncd.scr
```

```
[cumulus]
uid = web6
gid = client1
path = /var/www/clients/client1/web6/web
hosts allow = 192.168.0.11
comment = Backup cumulus.example.net
read only = true
auth users = test
secrets file = /etc/rsyncd.scr
```

```
[mysql]
uid = webadmin
gid = webadmin
path = /home/webadmin/backups/mysql_DB
hosts allow = 192.168.0.11
comment = Backup mysql
read only = true
auth users = test
secrets file = /etc/rsyncd.scr
```

```
[scripts_NS2]
uid = webadmin
gid = webadmin
path = /home/webadmin/scripts/
hosts allow = 192.168.0.11
```

```
comment = Backup scripts_NS2
read only = true
auth users = test
secrets file = /etc/rsyncd.scr

[radicale_users]
uid = root
gid = root
path = /etc/radicale/
hosts allow = 192.168.0.11
comment = Backup radicale users and config file
read only = true
auth users = test
secrets file = /etc/rsyncd.scr

[radicale_data]
uid = root
gid = root
path = /var/lib/radicale/collections/
hosts allow = 192.168.0.11
comment = Backup radicale data
read only = true
auth users = test
secrets file = /etc/rsyncd.scr

[rsync_NS2]
uid = root
gid = root
path = /etc/
exclude = lost+found/ **
include = rsyncd.conf
hosts allow = 192.168.0.11
comment = Backup rsync_NS2
read only = true
auth users = test
secrets file = /etc/rsyncd.scr

[crontab_NS2]
uid = root
gid = root
path = /var/spool/cron/crontabs/
exclude = lost+found/ **
```

```

include = webadmin
hosts allow = 192.168.0.11
comment = Backup crontab_NS2
read only = true
auth users = test
secrets file = /etc/rsyncd.scr

[www-full]
uid = root
gid = root
path = /var/www/
hosts allow = 192.168.0.11
comment = Backup full path of www disk
read only = true
auth users = test
secrets file = /etc/rsyncd.scr

[log_NS2]
uid = webadmin
gid = webadmin
path = /var/log/backup
hosts allow = 192.168.0.11
comment = Journal des taches nocturnes de ns2
read only = true
auth users = test
secrets file = /etc/rsyncd.scr

```

△ [gallerie] est le seul module en écriture – `read only = false`. Il sert à déposer des photos dans la galerie depuis un poste du réseau. Un module en écriture est un chemin d'écriture vers le disque des sites : il ne se justifie que par un usage précis, et il se relit à chaque revue.

△ Le filtre serveur n'est pas décoratif. Deux modules pointent sur des répertoires bien plus larges que ce dont on a besoin – `/etc/` et `/var/spool/cron/crontabs/` –, et six tournent avec `uid = root`. Le couple `exclude = lost+found/ **` et `include = <fichier>` restreint ce que le démon accepte de servir. Il vit côté serveur, n'est pas transmis au client, et ne se contourne pas.

⚠ Ne jamais écrire `include = ***` en pensant filtrer. Les `include` étant évalués avant les `exclude`, `***` retient absolument tout, y compris ce que la ligne `exclude` prétend écarter ensuite. Le module n'a alors aucun filtre, tout en paraissant en porter deux.

⚠ Pas de commentaire en fin de ligne dans ce fichier. `rsyncd` ne reconnaît `#` comme marque de commentaire qu'en début de ligne. Un `hosts allow = 192.168.0.11 #ici votre IP` fait lire `#ici`, `votre` et `IP` comme trois noms d'hôtes supplémentaires. Ils ne résolvent pas, donc ils sont ignorés, et le filtrage semble fonctionner – par accident.

⚠ `www-full` sert `/var/www/` en entier, sans filtre. C'est voulu – c'est la copie de sûreté du disque des sites – mais cela veut dire que le mot de passe du démon donne accès à la totalité des données hébergées. Le filtrage par `hosts allow` est ici la seule barrière qui reste.

Le mot de passe

[dans `web-srv`] :

```
nano /etc/rsyncd.scr
```

Une ligne, `utilisateur:motdepasse` :

```
test:<mot-de-passe>
```

[dans `web-srv`] :

```
chmod 640 /etc/rsyncd.scr
```

⚠ Supprimer un fichier qui contenait un mot de passe ne rattrape pas ses copies. Un secret qui a circulé – dans un script, dans une sauvegarde, dans un journal – n'est refermé que par sa rotation. Le retrait du fichier n'est que du ménage.

Durcir le service

[dans web-srv] :

```
mkdir -p /etc/systemd/system/rsync.service.d
```

[dans web-srv] :

```
nano /etc/systemd/system/rsync.service.d/override.conf
```

```
[Service]
Restart=always
RestartSec=5
```

[dans web-srv] :

```
systemctl daemon-reload
systemctl enable --now rsync
systemctl show rsync -p ProtectSystem -p PrivateDevices -p NoNewPrivileges
```

□□ Le durcissement est déjà dans l'unité fournie par le paquet :

`ProtectSystem=full`, `PrivateDevices=yes` et `NoNewPrivileges=yes` y figurent. Il n'y a rien à ajouter – la vérification ci-dessus sert à le constater plutôt qu'à le supposer.

△□ Ne jamais ajouter `ProtectHome=`. C'est la première option que suggèrent les guides de durcissement, et elle masquerait `/home` au service – or trois modules servent `/home/webadmin`. Ils échoueraient tous d'un coup, et sans bruit.

□□ Un *drop-in* plutôt qu'une copie de l'unité dans `/etc/systemd/system/`. Une unité copiée fige la version du jour : les correctifs apportés par les mises à jour du paquet ne l'atteignent plus.

□□ Aucun redémarrage n'est nécessaire après modification de `rsyncd.conf` : le

démon relit sa configuration à chaque connexion entrante. C'est aussi pourquoi un module cassé n'empêche jamais le démon de tourner – et pourquoi il peut échouer en silence pendant des années.

Le vidage des bases

Le conteneur produit ses vidages localement ; l'hôte vient les chercher par le module `mysql`.

[dans web-srv] – d'abord le répertoire du journal, qui demande les droits de root :

```
mkdir -p /var/log/backup
chown webadmin:webadmin /var/log/backup
```

△ Le `chown` est indispensable. La tâche tourne sous `webadmin`, qui ne peut pas écrire dans `/var/log`. Un répertoire resté à `root` produirait un « Permission denied » dans la redirection elle-même – donc une erreur sans destination, invisible.

[dans web-srv] – quitter le shell root pour ne pas créer les fichiers suivants en root :

```
su - webadmin
```

[dans web-srv, compte webadmin] :

```
mkdir -p /home/webadmin/backups/mysql_DB /home/webadmin/scripts
```

Les identifiants, hors de la ligne de commande

[dans web-srv, compte webadmin] :

```
nano /home/webadmin/.my.cnf
```

```
[client]
user = root
password = <mot-de-passe>
```

[dans web-srv, compte webadmin] :

```
chmod 600 /home/webadmin/.my.cnf
```

⚠ Jamais `-p<motdepasse>` sur la ligne de commande. Elle est visible dans `ps` par n'importe quel utilisateur de la machine pendant toute la durée de l'exécution, et elle se retrouve dans l'historique du shell.

Le script

[dans web-srv, compte webadmin] :

```
nano /home/webadmin/scripts/backup_mysql.sh
```

```
#!/bin/bash
# Vidage des bases MySQL vers ~/backups/mysql_DB/
# Identifiants dans ~/.my.cnf – jamais sur la ligne de commande.
# En cas d'échec, la sauvegarde précédente est CONSERVÉE.

DEST="/home/webadmin/backups/mysql_DB"
ERREURS=0
echo "=== $(basename "$0") – début $(date -Is)"

mkdir -p "$DEST"

# Découverte des bases, en excluant les schémas système
BASES=$(mysql -N -B -e "SHOW DATABASES" \
    | grep -Ev '^(information_schema|performance_schema|mysql|sys)$')
[ -z "$BASES" ] && { echo "ÉCHEC : aucune base listée"; exit 1; }
```

```

for db in $BASES; do
    TMP="$DEST/.$db.sql.tmp"
    if mysqldump --single-transaction --quick \
                --routines --triggers --events \
                --databases "$db" > "$TMP"; then
        mv "$TMP" "$DEST/$db.sql"
        echo "OK      $db"
    else
        rm -f "$TMP"
        echo "ÉCHEC  $db – la sauvegarde précédente est conservée"
        ERREURS=$((ERREURS + 1))
    fi
done

# Retrait des vidages dont la base n'existe plus.
# Uniquement si tout a réussi : sur une liste partielle, on effacerait
# des sauvegardes valides.
if [ "$ERREURS" -eq 0 ]; then
    for f in "$DEST"/*.sql; do
        [ -e "$f" ] || continue
        db=$(basename "$f" .sql)
        if ! printf '%s\n' "$BASES" | grep -qx "$db"; then
            rm -f "$f" && echo "RETIRÉ $db.sql – la base n'existe plus"
        fi
    done
fi

echo "=== $(basename "$0") – fin $(date -Is) – échecs : $ERREURS"
exit "$ERREURS"

```

[dans web-srv, compte webadmin] :

```
chmod 700 /home/webadmin/scripts/backup_mysql.sh
```

Les décisions inscrites dans ce script :

- Découverte dynamique des bases – une base ajoutée par le panneau est sauvegardée sans qu'on y pense.

-  Vérification de liste vide. Sans elle, un MariaDB qui ne répond pas produirait une boucle qui ne tourne pas, un compteur d'échecs à zéro, et un succès annoncé avec zéro base sauvegardée. Un compteur d'échecs ne voit pas l'absence de tentative.
-  Écriture dans un fichier temporaire, puis renommage. Une redirection directe vers le fichier final le tronque avant de savoir si le vidage réussira : un échec laisse alors un fichier vide à la place de la sauvegarde de la veille.
-  Aucune purge si un vidage a échoué. Supprimer une ancienne sauvegarde quand la nouvelle vient d'échouer est la façon classique de se retrouver sans rien.
- Code de sortie égal au nombre d'échecs – celui d'un script est autrement celui de sa dernière commande, si bien qu'un échec au milieu passe inaperçu.
- Marqueurs de début et de fin – les tâches écrivent dans un journal commun ; sans eux, rien ne permet d'attribuer une ligne à la bonne exécution.

La tâche planifiée

[dans web-srv, compte webadmin] :

```
crontab -e
```

```
MAILTO=""
00 00 * * * bash /home/webadmin/scripts/backup_mysql.sh >> /var/log/backup/$(date
+\\%F).log 2>&1
30 03 1 * * find /var/log/backup/ -name '*.log' -mtime +180 -delete
```

 Une ligne de cron ne se coupe pas. Cron n'accepte aucune continuation par barre oblique inverse : chaque tâche tient sur une seule ligne, aussi longue soit-elle. Dans un PDF elle s'enroule sans signe visible – la recopier depuis le fichier `.md`, jamais depuis le PDF.

△ Le `\%` est obligatoire. Dans une crontab, `%` marque une fin de ligne et bascule le reste vers l'entrée standard de la commande. Un `date +%F` non échappé casse la ligne de façon parfaitement déroutante.

△ `bash` et non `sh`. Le préfixe `sh` court-circuite le `#!/bin/bash` du script et l'exécute sous l'interpréteur par défaut. Un script écrit pour bash mais lancé par `sh` échoue uniquement en cron, jamais à la main – le pire des modes de défaillance.

La crontab de root

ISPConfig y inscrit ses deux tâches à l'installation. Elles ne se modifient pas :

```
* * * * * /usr/local/ispconfig/server/server.sh 2>&1 | while read line; do echo  
`/bin/date` "$line" >> /var/log/ispconfig/cron.log; done  
* * * * * /usr/local/ispconfig/server/cron.sh 2>&1 | while read line; do echo  
`/bin/date` "$line" >> /var/log/ispconfig/cron.log; done
```

△ Ces deux lignes non plus ne se coupent pas, et elles sont longues. Elles s'enroulent dans le PDF sans marque – les relire depuis le fichier `.md` en cas de reconstruction. Cela dit, elles n'ont normalement pas à être retapées : l'installateur d'ISPConfig les inscrit lui-même.

□□ C'est cette tâche à la minute qui applique les changements du panneau. Un réglage enregistré dans l'interface n'existe sur le disque qu'après son passage : attendre une minute avant de conclure qu'un changement n'a pas pris.

Ce qui est installé et ne sert pas

Un serveur de plusieurs années accumule des outils qu'on a cessé d'employer. Les nommer vaut mieux que les découvrir.

Outil	État
fail2ban	installé, service désactivé
ufw	installé, <code>ENABLED=no</code> , aucune règle, service désactivé
awstats	installé, tâche cron entièrement commentée
webalizer, vlogger, goaccess	installés, sans emploi
rkhunter	installé, sans exécution planifiée
jailkit	installé, requis par ISPConfig pour les shells confinés
certbot	installé, ne détient aucun certificat

Il n’y a donc pas de pare-feu sur ce conteneur, et c’est un choix. Ce qui écoute sur le réseau se réduit à SSH, FTP, le démon rsync, Apache et le panneau. Le démon rsync est filtré par lui-même – `hosts allow` dans chaque module –, et ce filtre-là, contrairement à une règle de pare-feu, ne peut pas être court-circuité par une route inattendue. Les autres services doivent rester joignables depuis le réseau local.

△ Un service `enabled` qui ne fait rien est pire que pas de service du tout. `ufw` répondait `enabled` à `systemctl is-enabled` alors que `ufw status` répondait `inactive` et qu’`iptables -S` ne montrait que trois politiques `ACCEPT`. Une machine qui se déclare protégée sans l’être fausse toute revue de sécurité. Si le pare-feu n’est pas en service, le dire :

```
systemctl disable ufw
```

□ Le filtrage réseau dans un conteneur non privilégié demande vérification avant d’être promis. Il ne suffit pas d’installer `ufw` et de l’activer : il faut constater que les règles s’appliquent réellement. Un pare-feu installé sans être éprouvé rassure sans protéger.

△ `fail2ban` mérite d’être remis en service ou retiré, pas laissé dans l’entre-deux. Sa configuration doit alors ne décrire que des services qui existent – une prison qui surveille un journal qu’aucun service n’alimente ne signale jamais rien, et donne pourtant l’impression d’une surveillance. Pour ce serveur, la seule prison qui aurait un objet :

```
[pure-ftpd]
enabled = true
port = ftp
filter = pure-ftpd
logpath = /var/log/syslog
maxretry = 5
```

Échéances

Deux dates sont connues, aucune n'est urgente, et les ignorer coûterait cher plus tard.

Ubuntu 22.04 cesse d'être maintenu en avril 2027. C'est cette date qui gouverne tout le reste : elle borne les correctifs de sécurité du système, d'Apache et de MariaDB.

Le support communautaire de MariaDB 10.6 a pris fin le 6 juillet 2026. Ubuntu 22.04 continue d'y rétroporter les correctifs de sécurité jusqu'à sa propre fin de support. La version reste donc sûre, mais elle est en deçà de ce que WordPress recommande aujourd'hui – sans conséquence fonctionnelle observée.

Ce qu'une reconstruction neuve choisirait aujourd'hui, et qui ne s'obtient pas par retouches :

Élément	Aujourd'hui	Une reconstruction
Système	Ubuntu 22.04.5	Ubuntu 24.04 LTS, comme l'hôte
Base de données	MariaDB 10.6	MariaDB 10.11 des dépôts de 24.04
PHP des sites	8.3	8.3 ou 8.4, sury restant nécessaire
Certificats	générique commercial, manuel	Let's Encrypt par ISPConfig
<code>php_fpm_socket_dir</code>	<code>/var/lib/php8.1-fpm</code>	un nom sans numéro de version

▲ La montée de version est une reconstruction, pas une mise à jour. ISPConfig, les pools FPM, les hôtes virtuels et les bases sont tous liés à la version du système : la voie sûre est un conteneur neuf en 24.04, monté en parallèle, vers lequel on bascule après vérification – pas un `do-release-upgrade` sur celui-ci.

Vérifications après reconstruction

[dans web-srv] – le système :

```
hostname -f
ip -br addr show eth0
systemctl --failed
```

[dans web-srv] – les services :

```
systemctl is-active apache2 mariadb rsync pure-ftpd-mysql radicale \
php8.3-fpm php8.4-fpm php8.5-fpm
```

[dans web-srv] – ce qui écoute, et sur quelle interface :

```
ss -ltn
```

MariaDB, Radicale et memcached doivent apparaître sur `127.0.0.1` uniquement.

[dans web-srv] – Apache et PHP :

```
apache2ctl configtest
apache2ctl -M | grep -i php
grep -R -niE '^[[:space:]]*(Alias|ScriptAlias)' /etc/apache2/conf-enabled/
```

[dans web-srv] – les six sites, d'un coup :

```
for s in blog photos dav lt musique routeur; do
  printf '%-8s %s\n' "$s" \
    "$(curl -s -o /dev/null -w '%{http_code}' -k \
      --resolve $s.example.com:443:127.0.0.1 https://$s.example.com/)"
done
```

Les codes attendus : `200`, `401`, `200`, `200`, `302`, `401`. Ce ne sont pas tous des 200, et c'est normal – deux sites exigent une authentification, un troisième redirige vers sa page d'accueil. Un `200` partout signifierait qu'une protection a disparu.

□ Le `--resolve` force la requête vers la machine elle-même au lieu de sortir par le routeur. C'est ce qui rend le test fiable depuis l'intérieur du conteneur, où le nom public pointe vers une adresse externe.

Sur une autre machine du réseau – le tirage de sauvegarde, en conditions réelles :

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

⚠ Ne jamais tester un service macvlan depuis l'hôte. Un `curl` lancé depuis nas-host vers ce conteneur peut geler sans message : les paquets partent par l'interface parente, où le macvlan interdit précisément le dialogue hôte↔conteneur. Le test se fait depuis une autre machine – ou depuis la seconde carte réseau de l'hôte, montage décrit dans l'article de nas-host.

⚠ Redémarrer le conteneur avant de considérer l'installation terminée. C'est ce test qui révèle ce qui a été fait à la main sans être inscrit : un service lancé mais non `enable`, un montage absent de `fstab`, un réglage appliqué à chaud. Rien de cela ne se voit tant qu'on ne redémarre pas.

Ce que ce guide ne couvre pas

- la création d'un conteneur Incus, les profils et le réseau macvlan – [article de l'hôte](#)
- les tirages nocturnes qui consomment le démon rsync décrit ici – [Sauvegarde](#)
- la restauration d'un site ou d'une base – [Récupération, restauration, reconstruction](#)
- Radicale, Piwigo et WordPress – un article par application
- la sécurisation de l'accès distant – article dédié