SQUID proxy — Mandataire de contournement

Référence : Squid 6.14 — Ubuntu 24.04 — conteneur Incus

Guide de reconstruction. Il décrit comment rebâtir cette machine et pourquoi elle est réglée ainsi.

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



Ce que cette page couvre : un mandataire HTTP posé dans un conteneur, qui donne aux navigateurs de la maison une voie de sortie qui ne passe pas par le filtrage du réseau. L'installation, la configuration locale — deux lignes —, le réglage des navigateurs, et les vérifications qui distinguent un service qui répond d'un service qui fait son travail.

Contexte : un hôte Incus portant plusieurs conteneurs en macvlan, chacun avec son adresse sur le réseau domestique, et un filtreur DNS qui sert le reste de la maison. Les adresses, les noms d'hôtes et les noms de règles sont des exemples à transposer.

Trois choses valent le détour même sans suivre le guide de bout en bout. Dans Squid, le placement d'une règle décide de tout, et un mauvais placement n'émet aucun message — section *Où vit la configuration locale*. Un fichier d'état que Squid écrit par défaut grossit indéfiniment, sans que rien ne le signale — section *La table netdb*. Et sur Ubuntu 24.04, un service peut se déclarer `inactive` alors que son port est bien ouvert — section *Vérifier*.

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`.

À quoi sert cette machine

Le réseau domestique passe par un filtreur DNS qui bloque les domaines de publicité et de pistage pour tous les appareils, sans que chacun ait à être réglé. C'est le bon réglage par défaut, et il a un défaut : quand un site refuse de fonctionner parce qu'une de ses ressources est bloquée, il n'y a pas de bouton pour voir ce que donnerait la page sans filtrage.

Ce mandataire est ce bouton. Un navigateur réglé pour passer par lui sort du réseau sans consulter le filtreur, le temps d'une consultation, et le reste de la maison n'est pas touché. Le basculement se fait dans le navigateur, en un clic, par une extension.

Il ne cache rien sur disque et ne conserve rien. Il n'est ni un accélérateur, ni un filtre, ni un anonymiseur : il relaie. Les requêtes qui le traversent sortent avec l'adresse publique de la maison, exactement comme les autres.

Le conteneur

La création d'un conteneur, les profils et le stockage sont décrits dans Incus — hôte et conteneur & serveur NAS. Ce qui suit ne retient que ce qui est propre à cette machine.

Le conteneur est non privilégié, sans aucun périphérique attaché, et il démarre avec l'hôte. Sa mémoire est plafonnée à 1 Gio, sans limite de processeur. Il porte les deux profils du parc, `default` et `macvlan`, ce dernier lui donnant sa propre adresse sur le réseau local plutôt qu'une adresse privée derrière

l’hôte.

Sur l’hôte — poser le plafond de mémoire :

```
incus config set squid-proxy limits.memory 1GiB
```

 1 Gio est environ trois fois ce que la machine consomme, qui tourne autour de 320 Mio. Un plafond n’est pas une réservation : il sert à empêcher qu’un emballement affame les autres conteneurs, pas à serrer au plus près. Le raisonnement complet, et la commande qui relève la consommation de tout le parc, sont dans l’article de l’hôte.

Sur l’hôte — vérifier ce que porte l’instance :

```
incus config show squid-proxy
```

`devices: {}` et `boot.autostart: "true"` sont les deux valeurs qui comptent. Un mandataire n’a aucun disque à monter, et il doit remonter seul après une coupure de courant.

Le réseau, et pourquoi ses résolveurs sont extérieurs

Le conteneur porte une adresse fixe et ne résout pas par le filtreur DNS de la maison. C’est délibéré, et c’est même la condition pour qu’il serve à quelque chose : une voie de contournement qui demanderait ses adresses au filtreur qu’elle contourne ne contournerait rien.

Cela lui donne un second avantage : il ne dépend d’aucun conteneur voisin. Le filtreur peut être arrêté, redémarré ou reconstruit sans que le mandataire cesse de fonctionner.

Sur l’hôte — ouvrir l’éditeur :

```
nano /tmp/50-static-public-ip.yaml
```

Contenu du fichier, à coller dans l’éditeur ouvert ci-dessus :

```
network:
  version: 2
  renderer: networkd
  ethernets:
    eth0:
      dhcp4: no
      dhcp6: no
      addresses: [192.168.0.7/24]
      nameservers:
        addresses:
          - 208.67.220.222
          - 208.67.220.220
      routes:
        - to: default
          via: 192.168.0.1
```

Sur l’hôte — la pose et la relecture :

```
incus file push /tmp/50-static-public-ip.yaml \
  squid-proxy/etc/netplan/50-static-public-ip.yaml \
  --uid 0 --gid 0 --mode 0600
incus file pull squid-proxy/etc/netplan/50-static-public-ip.yaml -
```

⚠ La relecture n’est pas décorative. Un chemin de dépôt fautif ne produit pas d’erreur : le fichier est simplement créé ailleurs, et la machine démarre avec une configuration réseau qui n’est pas celle qu’on croit avoir posée.

Le mode `0600` n’est pas un raffinement : netplan se plaint d’un fichier plus ouvert que cela. Et `--uid 0 --gid 0` non plus — sans eux, le fichier arrive avec le propriétaire par défaut du conteneur.

Le conteneur ne configure pas son réseau par cloud-init, qui est neutralisé par deux fichiers posés à la construction : `/etc/cloud/cloud-init.disabled` et

`/etc/cloud/cloud.cfg.d/99-disable-network-config.cfg`.

Le nom d’hôte, des deux côtés

Squid cherche à connaître le nom complet de la machine sur laquelle il tourne ; il le met dans ses pages d’erreur. S’il ne le trouve pas, il le dit à chaque lecture de sa configuration :

```
WARNING: 'squid-proxy' rDNS test failed: (0) No error.
WARNING: Could not determine this machines public hostname.
```

Le remède est dans `/etc/hosts`, et il vaut pour tout ce qui tourne dans le conteneur plutôt que pour le seul Squid.

Sur l’hôte — ouvrir l’éditeur :

```
nano /tmp/hosts
```

Contenu du fichier, à coller dans l’éditeur ouvert ci-dessus :

```
127.0.0.1      localhost
192.168.0.7    squid-proxy.internal 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
```

Sur l’hôte — la pose :

```
incus file push /tmp/hosts squid-proxy/etc/hosts \
  --uid 0 --gid 0 --mode 0644
```

```
incus exec squid-proxy -- hostname -f
```

La dernière commande doit répondre `squid-proxy.internal`.

Le second geste se fait ailleurs : déclarer `squid-proxy.internal` dans les enregistrements DNS locaux du filtreur, pour qu'il soit résolvable depuis tout le réseau et pas seulement de l'intérieur du conteneur. Les deux ne font pas le même travail et ne se remplacent pas.

Installer Squid

Sur l'hôte :

```
incus exec squid-proxy -- apt-get update
incus exec squid-proxy -- apt-get -y install squid
```

Le paquet vient de l'archive Ubuntu. Cette machine n'a aucun dépôt tiers ni PPA — c'est ce qui rend sa maintenance et ses montées de version sans histoire.

Le service est mis en marche et activé au démarrage par le paquet lui-même ; il n'y a pas de `systemctl enable` à faire.

□□ Ne pas faire de copie de sûreté de `squid.conf`. C'est l'usage hérité des installations où l'on édite le fichier livré, et la suite de cette page ne l'édite pas. Une telle copie vieillit d'ailleurs très mal : elle est prise une fois, à la première installation, et reste figée pendant que le fichier du paquet évolue d'une version majeure à l'autre. Le fichier de référence du paquet courant, c'est `/etc/squid/squid.conf.dpkg-dist`, déposé par le système chaque fois qu'un fichier modifié est conservé.

Où vit la configuration locale

Tout ce que cette installation ajoute au fichier livré tient en deux lignes : la désignation du réseau local, et son autorisation.

Le fichier

Elles ne sont pas écrites dans `squid.conf`, mais dans un fichier séparé que Squid lit déjà.

Sur l’hôte — ouvrir l’éditeur :

```
nano /tmp/99-local.conf
```

Contenu du fichier, à coller dans l’éditeur ouvert ci-dessus :

```
# Réglages locaux de Squid. Ce fichier n'appartient à aucun paquet :
# il survit aux montées de version, et squid.conf reste celui d'Ubuntu.
#
# Il est lu par l'include de squid.conf, placé après les refus
# « deny to_localhost » et « deny to_linklocal » et avant « deny all ».
# L'autorisation ci-dessous arrive donc au bon rang : elle ne peut pas
# annuler les deux protections, et elle passe avant le refus final.

acl fortytwo_network src 192.168.0.0/24
http_access allow fortytwo_network

# La table netdb ne sert qu'à arbitrer entre mandataires pairs.
# Cette machine n'en a aucun. Sans cette ligne, Squid ajoute sa table
# au même fichier toutes les heures, sans jamais le remplacer.
netdb_filename none
```

Sur l’hôte — la pose, puis le contrôle avant tout redémarrage du service :

```
incus file push /tmp/99-local.conf \
  squid-proxy/etc/squid/conf.d/99-local.conf \
```

```
  --uid 0 --gid 0 --mode 0644
incus exec squid-proxy -- squid -k parse
```

`squid -k parse` lit toute la configuration et énumère les directives dans l'ordre où elles seront appliquées. C'est cette sortie qu'il faut regarder, et pas seulement l'absence d'erreur :

```
Processing: http_access allow localhost
Processing: http_access deny to_localhost
Processing: http_access deny to_linklocal
Processing: include /etc/squid/conf.d/*.conf
Processing Configuration File: /etc/squid/conf.d/99-local.conf (depth 1)
Processing: acl fortytwo_network src 192.168.0.0/24
Processing: http_access allow fortytwo_network
Processing: netdb_filename none
Processing: http_access deny all
```

Sur l'hôte — appliquer :

```
incus exec squid-proxy -- systemctl restart squid
```

□ Le placement décide de tout

Squid lit ses règles d'accès dans l'ordre et s'arrête à la première qui correspond ; les suivantes ne sont jamais consultées. Une autorisation placée trop haut n'est pas une nuance de style : elle annule tout ce qui vient après.

Le fichier livré par Ubuntu 24.04 porte deux refus que les versions plus anciennes n'avaient pas :

```
http_access deny to_localhost
http_access deny to_linklocal
```

Ils empêchent qu'on se serve du mandataire pour atteindre le `127.0.0.1` de la machine elle-même, ou les adresses de lien-local en `169.254`. Une autorisation

du réseau local placée avant eux les rend inopérants pour tout le réseau — sans message, sans erreur au démarrage, et sans que rien dans le comportement quotidien ne l'indique. C'est exactement ce que ferait un « je remets ma ligne au même endroit qu'avant » après une montée de version.

L'autorisation doit venir après les deux refus et avant le `http_access deny all` qui termine la liste. Le répertoire `conf.d/` est lu à cet endroit précis : c'est ce que montre la sortie ci-dessus, et c'est ce qui rend le fichier séparé possible.

⚠ À relire après chaque montée de version, puisque la position de cette ligne de lecture appartient au paquet, pas à nous.

Pourquoi pas « allow localnet »

Le fichier livré définit déjà un réseau local, `localnet`, couvrant l'ensemble des plages privées — dont `192.168.0.0/16`, soit 65 536 adresses. Son autorisation existe aussi, en commentaire, dans le fragment fourni par Debian. Décommenter cette ligne serait plus simple et plus large : elle ouvrirait le mandataire à des plages d'adresses que ce réseau n'emploie pas.

La règle locale fait le même travail sur le seul `/24` réellement utilisé. C'est une décision, pas un oubli : ne pas la « simplifier » à la reconstruction.

Pourquoi un fichier séparé plutôt qu'une édition

Le fichier livré fait 353 000 caractères, dont l'essentiel est de la documentation. Il appartient au paquet. Chaque fois qu'Ubuntu en publie une version neuve alors qu'il a été modifié localement, le système demande quoi faire : garder la version locale ou prendre celle du paquet. Il faut répondre juste, des années plus tard, en se rappelant pourquoi.

 Et il existe un cas où la question n'est pas posée. Quand un paquet en remplace un autre sous un nom différent — ce qui arrive lors d'un changement de version majeure —, le nouveau reprend les fichiers de configuration de son prédécesseur sans écran de configuration, sans copie de l'ancien, et sans une

ligne au journal. Un réglage local peut disparaître de cette façon.

Un fichier qui n'appartient à aucun paquet ne peut être ni proposé au remplacement ni repris en silence. `squid.conf` reste alors rigoureusement celui d'Ubuntu, ce qui se vérifie d'une commande. Sur l'hôte :

```
incus exec squid-proxy -- diff /etc/squid/squid.conf.dpkg-dist \
  /etc/squid/squid.conf
```

Aucune sortie : les deux fichiers sont identiques, il n'y a rien à arbitrer.

Ce que le paquet apporte, et qu'on ne touche pas

Le fragment de Debian

`/etc/squid/conf.d/debian.conf` est posé par le paquet dans le même répertoire que le fichier local, et lu juste après lui. Il tient en une directive utile :

```
logfile_rotate 0
```

Elle dit à Squid de ne pas faire tourner ses journaux lui-même, parce que c'est `logrotate` qui s'en charge, par le fragment `/etc/logrotate.d/squid` : rotation quotidienne, compression, deux jours conservés. C'est peu, et c'est cohérent avec une machine dont les journaux ne servent qu'à un dépannage immédiat.

Le port et l'écoute

```
http_port 3128
```

C'est le port par défaut, et il est conservé. L'écoute se fait sur toutes les interfaces — ce n'est pas elle qui protège la machine, c'est la liste des

règles d’accès. Une requête venue d’ailleurs est acceptée par le système puis refusée par Squid.

Le port peut être changé par cette même directive ; ce n’est pas une mesure de sécurité, seulement une commodité.

Pas de cache disque

Aucune directive `cache_dir` n’est déclarée : Squid ne cache qu’en mémoire, et tout est perdu à chaque redémarrage. C’est le bon réglage pour cet usage — la machine sert à contourner un filtrage pendant quelques minutes, pas à accélérer quoi que ce soit. Il n’y a donc rien à sauvegarder dans ce conteneur, et rien à reconstruire après une panne : le paquet, le fichier local, le nom d’hôte.

`/var/spool/squid` existe quand même : c’est le répertoire où Squid déposerait un fichier de diagnostic en cas de plantage.

⚠ La table netdb

Squid tient une table de mesures de distance et de temps de réponse vers les réseaux qu’il a visités. Elle sert à arbitrer entre plusieurs mandataires qui se parlent, quand il faut décider lequel est le mieux placé pour atteindre un site. Cette machine n’a aucun pair : personne ne consulte cette table.

Elle continue pourtant d’être écrite, environ une fois l’heure. Et le réglage par défaut l’écrit comme un journal : chaque sauvegarde s’ajoute aux précédentes, le fichier n’est jamais remplacé ni purgé. Il grossit d’environ 150 Mio par an, indéfiniment, dans un répertoire que personne ne regarde.

La ligne `netdb_filename none` du fichier local referme le mécanisme. Sur une machine où il a déjà tourné, le fichier se supprime ensuite, une fois le service redémarré. Sur l’hôte :

```
incus exec squid-proxy -- rm -f /var/spool/squid/netdb.state
incus exec squid-proxy -- ls -la /var/spool/squid/
```

□□ Sur un système de fichiers à instantanés, l’espace ne se libère qu’au retrait des instantanés qui retiennent encore l’ancien fichier.

Vérifier

Le service

Sur l’hôte :

```
incus exec squid-proxy -- systemctl is-active squid
incus exec squid-proxy -- systemctl --failed --no-legend
incus exec squid-proxy -- ss -ltn
```

⚠□ Interrogé trop tôt après un démarrage, `is-active` répond faux. Squid se déclare `inactive` pendant les quelques secondes de son initialisation. Laisser vingt secondes avant de conclure.

Les trois codes

C’est le contrôle qui compte, et il ne se fait pas depuis l’hôte.

⚠□ Un service en macvlan ne se teste pas depuis la machine qui le porte : les paquets partent par l’interface parente, où le macvlan interdit le dialogue entre l’hôte et ses conteneurs.

Depuis une autre machine du réseau :

```
for url in http://example.com/ https://example.com/ http://127.0.0.1/; do
  printf "%-24s " "$url"
  curl -s -o /dev/null -w "%{http_code}\n" --max-time 10 -x 192.168.0.7:3128
"$url"
done
```

Attendu, et ce que chacun prouve :

```
http://example.com/      200   le relais fonctionne en clair
https://example.com/     200   le relais fonctionne en tunnel
http://127.0.0.1/        403   « deny to_localhost » AGIT
```

 Le troisième est le seul qui ait une valeur de preuve. Les deux premiers montrent qu'une règle d'autorisation existe quelque part ; seul le `403` montre qu'elle est au bon rang, puisqu'une autorisation mal placée rendrait ici le code d'un relais qui a bien essayé de joindre la destination. Un refus est une réponse de Squid ; un échec de connexion est une réponse du réseau, et les deux ne se ressemblent pas.

 `example.com` est réservé par la RFC 2606 pour la documentation : le test ne sollicite aucun tiers réel.

⚠ Le piège du service SSH

Un relevé d'état de cette machine montre ceci :

```
systemctl is-active ssh        →  inactive
ss -ltn | grep :22             →  LISTEN  0  4096  0.0.0.0:22
```

Les deux sont exacts. Depuis Ubuntu 24.04, le serveur SSH est démarré à la demande : c'est `systemd` lui-même qui tient le port ouvert et lance le service à la première connexion. Un contrôle qui s'arrête à `is-active` conclut que SSH est tombé sur une machine où il fonctionne parfaitement.

C'est la famille de pannes apparentes — et de fausses alertes — qui coûte le plus de temps : deux mesures indépendantes valent mieux qu'une mesure nette.

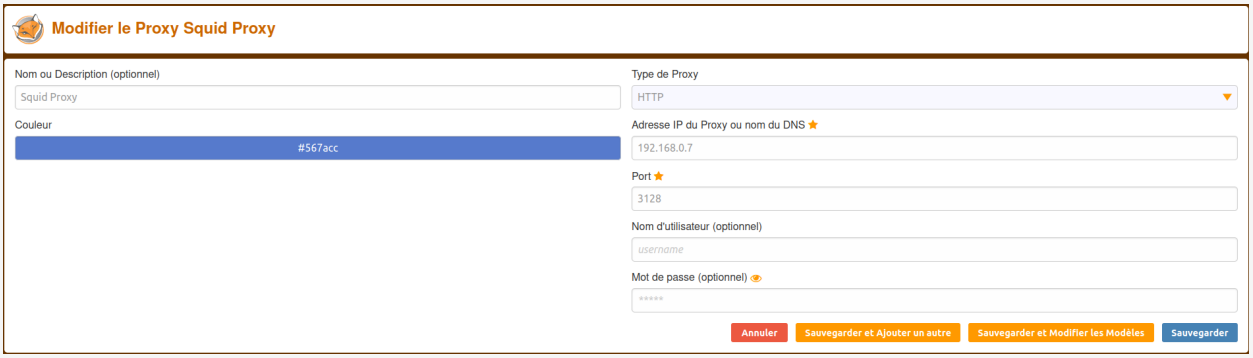
Côté client

Le mandataire ne sert à rien tant qu'aucun navigateur ne sait qu'il existe. Le réglage se fait par extension plutôt que dans les préférences du système, pour

deux raisons : il ne touche que le navigateur, et il se bascule en un clic au moment où une page refuse de s’afficher.

Dans tous les cas, le réglage est le même — serveur `192.168.0.7`, port `3128`, sans authentification.

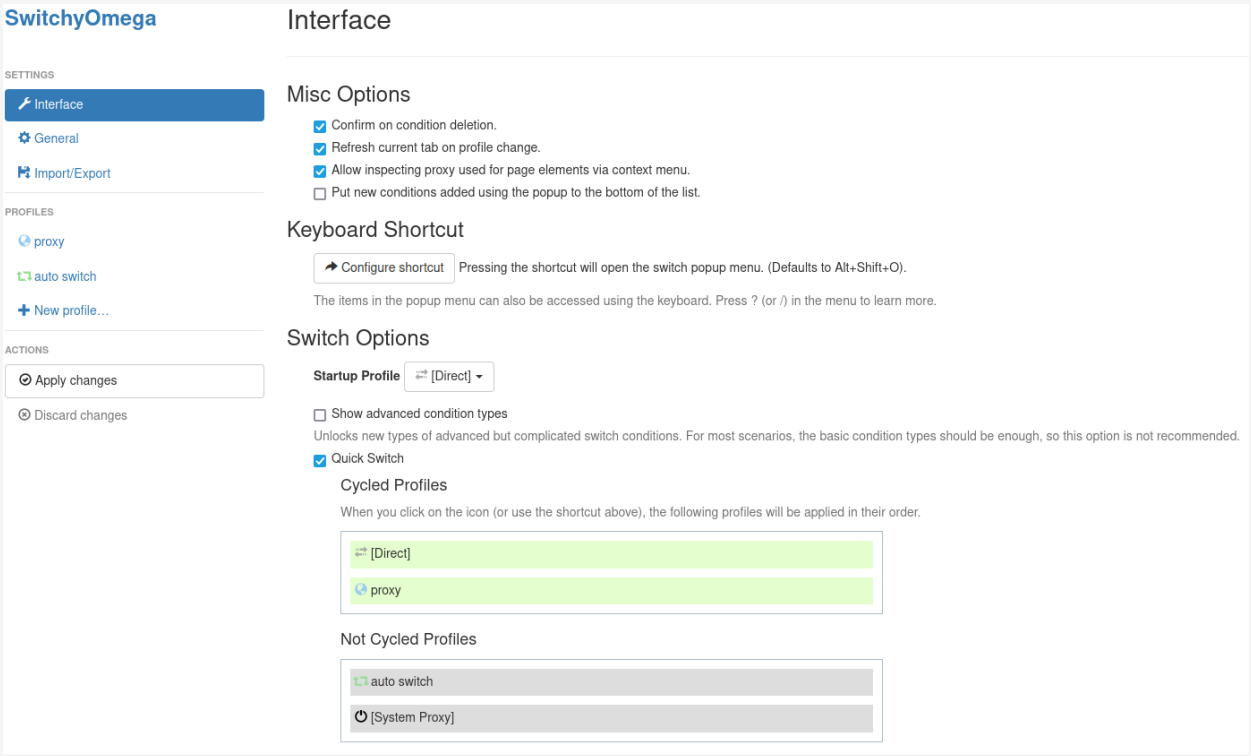
Firefox — FoxyProxy



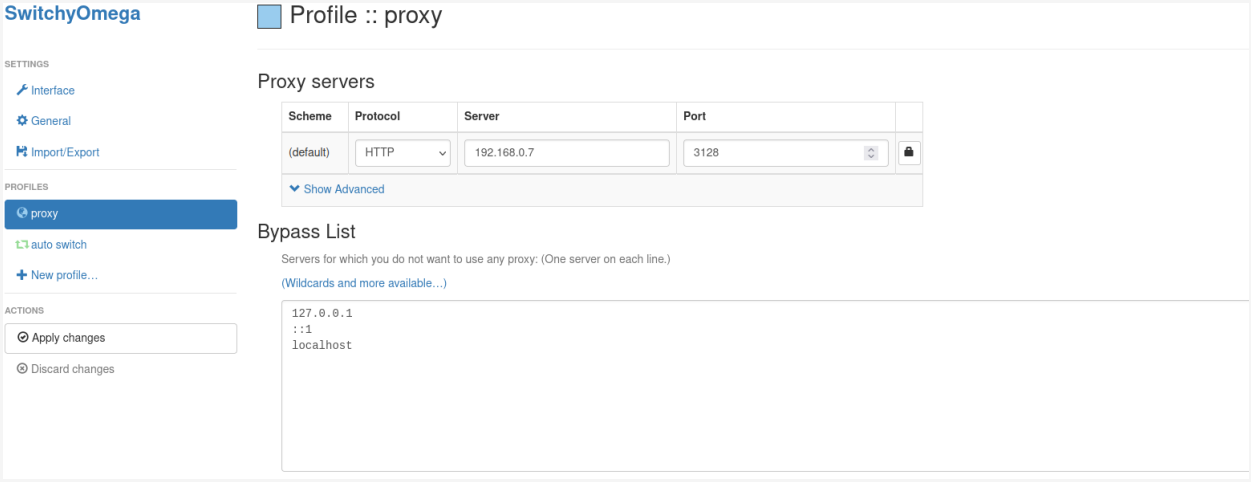
FoxyProxy — la configuration du mandataire

Un seul serveur déclaré, activé ou désactivé depuis l’icône de la barre d’outils.

Chromium — SwitchyOmega



SwitchyOmega — l’interface de réglage



SwitchyOmega — le profil du mandataire

L’extension travaille par profils : un profil direct, un profil mandataire, et le choix se fait depuis l’icône.

Android

FoxyProxy existe pour Firefox sur Android et s’y règle de la même façon. Les navigateurs fondés sur Chrome n’acceptent pas d’extension sur cette plateforme ; une application dédiée qui redirige le trafic vers un mandataire rend le même

service.

⚠ Un mandataire déclaré ne traverse pas les frontières du réseau. Ce réglage ne fonctionne qu'à la maison : hors du réseau local, le navigateur tentera de joindre une adresse privée et n'affichera plus rien du tout. C'est la première chose à regarder devant un téléphone qui « n'a plus Internet » en dehors de la maison.

Ce qui n'est pas dans cette page

La création du conteneur, les profils, le stockage et les sauvegardes sont dans Incus — hôte et conteneur & serveur NAS.

Le passage d'une version LTS à la suivante est dans Migrer un conteneur vers la prochaine version LTS. Cette machine est le meilleur candidat pour apprendre cette procédure : elle ne dépend que de l'archive Ubuntu, n'a aucun disque attaché, aucun logiciel hors paquets, et un seul service dont on sait mesurer la santé en trois codes.

Un enseignement de sa propre migration vaut d'être connu avant d'y aller : une configuration écrite pour Squid 5 fait tourner Squid 6 sans le moindre avertissement. La crainte de devoir reprendre une configuration ancienne était infondée — un `acl src` et un `http_access` sont stables depuis très longtemps. Ce qui change entre deux versions majeures, c'est le fichier livré autour d'eux, et c'est précisément ce dont le fichier local séparé affranchit.