LES GUIDES DE
GNU LINUX MAGAZINE / FRANCE
HORS-SÉRIE N°68

France METRO : 12.90 € — CH : 18.00 CHF — BEL/PORT.CONT : 13,90 € — DOM TOM : 13,90 € — CAN : 18,00 $ cad — MAR : 130 MAD

SERVEURS

LE GUIDE POUR CRÉER ET GÉRER VOS SERVICES À LA CARTE

100% pratique

TOUS LES CONSEILS POUR INSTALLER LE SERVEUR DONT VOUS AVEZ BESOIN SOUS LINUX

Introduction
Rappels et notions de base concernant le réseau et la sécurité

HTTP/web
Apache, Lighttpd, Nginx, trois solutions pour installer votre serveur web

Mail
Des solutions pour mettre en place votre propre serveur de mails

Les autres services utiles
Échange de fichiers (FTP), suivi de versions (Git) et réseau privé (OpenVPN)

Applications web
Blog, galerie de photos, wiki, boutique en ligne, ... tous les outils pour faire votre place sur la Toile

Édité par Les Éditions Diamond
L 15066 - 68 H - F: 12,90 € - RD
www.ed-diamond.com

Retrouvez toutes nos publications

LES ÉDITIONS DIAMOND

sur **www.ed-diamond.com**

GNU/Linux Magazine Hors-Série
est édité par **Les Éditions Diamond**

B.P. 20142 / 67603 Sélestat Cedex

Tél. : 03 67 10 00 20 / **Fax :** 03 67 10 00 21

E-mail : cial@ed-diamond.com
lecteurs@gnulinuxmag.com

Service commercial : abo@gnulinuxmag.com

Sites : www.gnulinuxmag.com
www.ed-diamond.com

Directeur de publication : Arnaud Metzler

Rédacteur en chef : Denis Bodor

Remerciements à Sébastien Maccagnoni-Munch

Conception graphique : Kathrin Scali

Responsable publicité : Tél. : 03 67 10 00 27

Service abonnement : Tél. : 03 67 10 00 20

Impression : pva, Druck und Medien-Dienstleistungen GmbH, Landau, Allemagne

Distribution France :
(uniquement pour les dépositaires de presse)

MLP Réassort :
Plate-forme de Saint-Barthélemy-d'Anjou.
Tél. : 02 41 27 53 12
Plate-forme de Saint-Quentin-Fallavier.
Tél. : 04 74 82 63 04

Service des ventes :
Distri-médias : Tél. : 05 34 52 34 01

IMPRIMÉ en Allemagne - PRINTED in Germany

Dépôt légal : A parution

N° ISSN : 0183-0864

Commission Paritaire : K78 976

Périodicité : Bimestrielle

Prix de vente : 12,90 Euros

Les articles non signés contenus dans ce numéro ont été rédigés par les membres de l'équipe rédactionnelle des Éditions Diamond.

Membre April
www.april.org

PRÉFACE

Internet : nous sommes nombreux à y être connectés en ce début de XXIème siècle. On a accès à des millions de sites web, des ressources diverses et variées ; on peut même participer : Facebook, Twitter et autres Google+ forment des plateformes de réseaux sociaux, permettant d'interagir avec ses amis. Mais on reste toujours lié à une plateforme, à un fournisseur, à un éditeur... On peut vouloir être indépendant, avoir un coin d'Internet à soi, son propre site web, ses propres services...

Pour se créer son espace personnel sur Internet, on a plusieurs possibilités : les solutions « clés en main » (qui permettent de publier rapidement, mais en restant dépendant d'un prestataire, même si certains services sont gratuits), l'hébergement mutualisé (parfait si on ne veut qu'un site web sans s'embêter à gérer un système) et différentes formes de serveurs hébergés (serveur virtuel « VPS » ou « cloud », serveur dédié, auto-hébergement...). À chacun de choisir la solution qui lui convient.

Dans ce numéro, nous allons aborder les solutions permettant d'avoir son propre serveur « entier » : un serveur virtuel, un PC auto-hébergé, ou de préférence un serveur dédié. Avec ces solutions, on sera capable non seulement d'héberger des sites web, mais également de toucher à toutes ces autres technologies qu'on peut trouver sur Internet : la messagerie électronique, le transfert de fichiers, les VPN... Pour tout cela, on ne peut pas se contenter d'un hébergement mutualisé.

Vous trouverez dans ces pages de nombreux tutoriels qui vous permettront d'installer différents logiciels offrant ces services. Nous avons choisi de les présenter sur le système d'exploitation Debian GNU/Linux, qui reste une référence dans le milieu des serveurs sous Linux ; en 20 ans, cette distribution s'est imposée comme une référence, avec plusieurs dizaines de milliers de paquets logiciels officiels maintenus par près d'un millier de bénévoles. La grande majorité des fournisseurs de serveurs dédiés ou de serveurs virtuels proposent cette distribution parmi les choix d'installation que vous pouvez faire lorsque vous commandez ces services.

Sauf indication contraire, ces tutoriels partent sur la base d'une distribution Debian 7 « Wheezy » installée sur une machine physique ou virtuelle, à laquelle on accède au travers d'une connexion SSH, sans interface graphique (comme tout serveur qui se respecte), avec l'utilisateur « root ». Les quelques manipulations en ligne de commandes à effectuer sur un client (connexions FTP par exemple) le sont, dans notre exemple, sur un poste de travail sous Ubuntu. Vous pouvez également, sans trop de difficultés, suivre les mêmes tutoriels avec un serveur basé sur Ubuntu, n'oubliez simplement pas l'habituel « sudo »... Nous estimons aussi que vous savez utiliser un éditeur de texte en ligne de commandes (Vim ou Nano, entre autres) et que vous savez éditer un fichier à distance, en vous connectant en SFTP (grâce à Nautilus par exemple).

La rédaction

Sommaire

GNU/Linux Magazine
Hors-Série
N°68

SERVEURS

1 INTRODUCTION

2 HTTP/WEB

3 MAIL

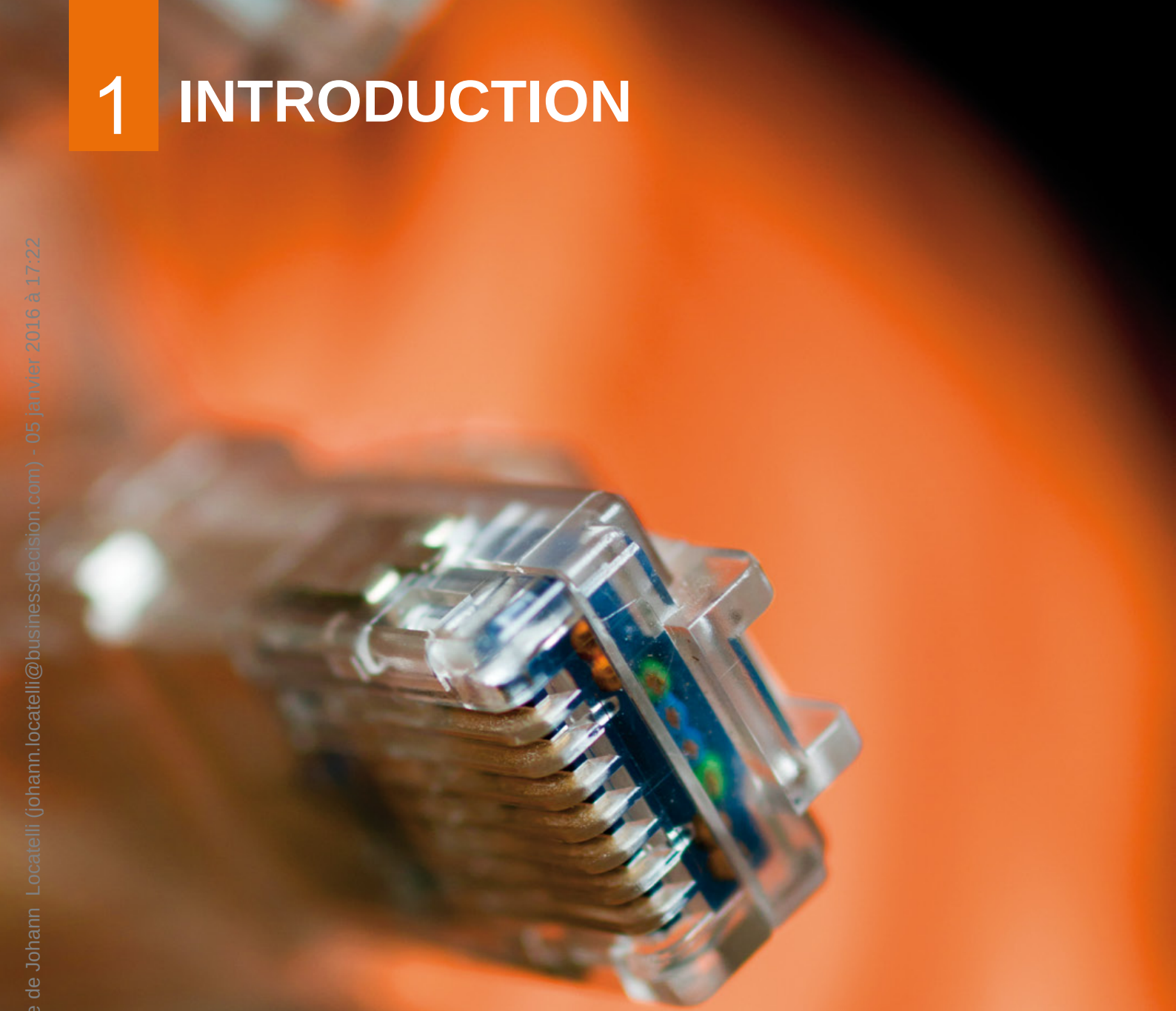
4 LES AUTRES SERVICES UTILES

5 TUTORIELS

1 INTRODUCTION

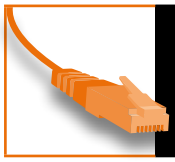
À découvrir dans cette partie...

1 INTRODUCTION



INTERNET : QU'EST-CE ?

On sait tous à peu près ce qu'est Internet : c'est un gros réseau sur lequel on peut connecter son ordinateur pour accéder à des sites et autres services. Approfondissons la question, voyons comment fonctionne Internet.

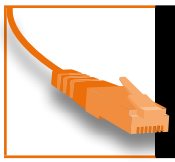


1 Les réseaux d'Internet

Internet est une interconnexion mondiale de réseaux : il ne s'agit en réalité pas d'un seul gros réseau, mais d'un ensemble de réseaux, qui peuvent communiquer entre eux grâce à l'utilisation de **protocoles standards** et d'équipements qui font « passerelle » entre plusieurs réseaux, les **routeurs**.

Cet ensemble de réseaux est complètement décentralisé, il n'y a pas de structure hiérarchique et il n'est pas possible d'« arrêter Internet ». Les interconnexions sont tellement nombreuses qu'il sera toujours possible de trouver un chemin d'une machine à une autre. Notons toutefois que, dans les pays où le gouvernement contrôle les interconnexions des fournisseurs d'accès (voire du seul fournisseur d'accès), celui-ci est tout à fait capable de faire couper le lien entre le pays et le reste d'Internet : ça s'est déjà vu lors de conflits civils ces dernières années...

Sur Internet, toute communication utilise le **protocole IP** (*Internet Protocol*) et tout ordinateur connecté est identifié par son **adresse IP**. Cette adresse IP est attribuée par un **Registre Internet Régional** (RIR) ; ceux-ci sont au nombre de 5 : l'AFRINIC en Afrique, l'APNIC en Asie-Pacifique, l'ARIN au Canada et aux États-Unis, le LACNIC en Amérique Latine et le RIPE pour l'Europe et le Moyen-Orient. Il existe également des intermédiaires, ce sont les LIR (*Local Internet Registry*). Ces LIR sont souvent les opérateurs et les fournisseurs d'accès : en France, le RIPE délègue l'attribution des adresses IP à Orange, Free, SFR ou Bouygues, parmi des centaines d'autres.



2 DNS

Lorsque l'on veut joindre une autre machine ou un service sur Internet (par exemple, un site web), on utilise rarement son adresse IP : ce serait particulièrement pénible de devoir retenir cette information numérique pour chaque site que l'on veut visiter.

C'est pourquoi le DNS (*Domain Name System*) a été conçu en 1983 : celui-ci associe un nom textuel à une adresse IP. Par exemple, le nom `www.editions-diamond.com` pointe vers l'adresse IP `94.23.219.217`.

2.1 Domaine et TLD

Un nom de domaine est composé de deux éléments :

- le domaine de premier niveau (TLD, *Top-Level Domain*), dernière partie du nom ;
- le domaine en lui-même.

Ils se présentent sous la forme suivante : `domaine.tld`.

À savoir

IPV4 ET IPV6

La première version du protocole IP largement répandue est IPv4, créée en 1980 et toujours utilisée. Celle-ci code les adresses IP sur 32 bits, ce qui signifie qu'un maximum d'environ 4,3 milliards d'adresses IP peut être attribué (232). Ces adresses IPv4 sont tout de même officiellement épuisées depuis février 2011 : toute nouvelle attribution d'une adresse est forcément précédée de sa « libération », une machine se retrouve alors sans adresse IP.

C'est pourquoi le protocole IPv6 a été créé et finalisé en 1998, pour être progressivement déployé. Celui-ci code les adresses sur 128 bits, offrant alors plus de 340 milliards de milliards de milliards de milliards d'adresses (2128). Ce déploiement avance toutefois lentement et très peu d'utilisateurs sont aujourd'hui en IPv6.

Les domaines de premier niveau sont gérés par des organismes centralisateurs, par exemple l'AFNIC pour les domaines en `.fr`. Les noms de domaines sont attribués par des **registrars** : ce sont des sociétés chargées de gérer des domaines par délégation de ces organismes centralisateurs.

Toute personne voulant un nom de domaine s'adresse alors à un registrar, qui peut lui attribuer un nom de domaine dans la limite de sa disponibilité. Un tel nom de domaine coûte généralement entre 5 et 40 euros par an : la politique tarifaire dépend du registrar lui-même, mais aussi de l'organisme qui lui délègue la gestion des domaines.

2.2 Noms de machines

Au-delà du nom de domaine en lui-même, c'est son propriétaire qui est chargé de gérer les noms de machines : après avoir acheté le nom `mon-super-domaine.com`, il est nécessaire de mettre en place les enregistrements des différents serveurs que l'on souhaite mettre en place, en faisant pointer `www.mon-super-domaine.com` vers un serveur web et `mail.mon-super-domaine.com` vers un serveur de messagerie.

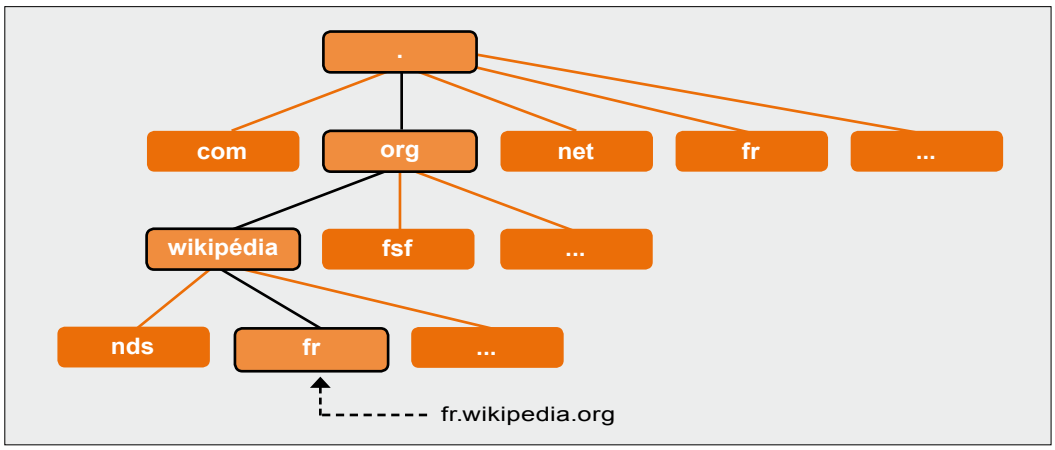
La plupart des registrars proposent directement une interface web permettant de configurer son propre domaine hébergé : la configuration du domaine sera alors hébergée sur les serveurs DNS du registrar. C'est la solution idéale dans la plupart des cas, ces serveurs étant généralement particulièrement robustes et sécurisés.

Il est également possible d'héberger soi-même son serveur DNS, soit parce que les options proposées par le registrar sont insuffisantes, soit par simple volonté de « tout gérer soi-même ». Dans ce cas, il faut utiliser un logiciel serveur DNS tel que **Bind**.

2.3 Interrogation des DNS

Tout ordinateur doit interroger son propre serveur DNS (dans la plupart des foyers français, ce sera une box), qui lui-même va retransmettre la requête à son parent, et ainsi de suite, jusqu'à arriver au « point de rencontre » où un serveur parent lui demandera de s'adresser à tel ou tel autre serveur DNS, afin d'interroger ce domaine particulier.

Les ordinateurs eux-mêmes ne s'adressent normalement jamais directement au serveur DNS qui gère un domaine particulier, sauf dans un cas bien précis : lorsque c'est le serveur DNS local qui gère le domaine concerné. ■



Le DNS est un système hiérarchique

3 Accès à Internet

Réservé à l'origine aux universités et aux chercheurs, l'accès à Internet s'est démocratisé ces dernières années. Dans les pays développés, on peut aisément s'abonner à un fournisseur d'accès à Internet, qui offre une connexion à Internet au travers d'une technologie particulière. Dans ce cadre, on peut être confronté aux connexions point-à-point (PPP) par le réseau téléphonique (RTC), aux réseaux mobiles (3G, 4G, etc.), ainsi que, bien sûr, à l'ADSL ou au câble. D'autres technologies sont en train d'être déployées par les opérateurs : fibre optique, VDSL... Tout cela permettant d'offrir aux utilisateurs des débits de plus en plus importants.

Dans les années 90, la connexion la plus courante en France était le modem RTC, qui offrait généralement un débit entre 28,8 Kbps et 56 Kbps. Au début du XXIème siècle s'est démocratisé l'ADSL, qui a permis des débits de 1 Mbps au début, jusqu'à 20 Mbps aujourd'hui. La fibre optique, quant à elle, offre des débits de 100, 200, voire 300 Mbps. Aujourd'hui, une connexion à Internet est plusieurs centaines de fois plus rapide qu'il y a 15 ans !

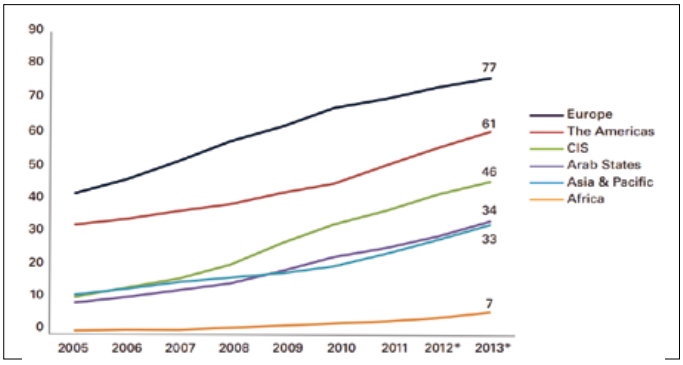
Mais tout le monde n'est pas logé à la même enseigne... Rendons à César ce qui est à César : les chiffres ci-dessous peuvent être consultés sur Wikipédia (page « Internet dans le monde »).

En France, 83 % de la population a accès à Internet en 2012, dont une grande proportion avec des connexions dites « haut débit » : câble, ADSL, fibre, etc. Il faut dire qu'au début des années 90, un trublion a bousculé le monde tranquille des fournisseurs d'accès en offrant de l'ADSL illimitée à 30 euros : tous lui ont maintenant emboîté le pas.

Ce n'est pas le cas dans tous les pays. Prenons par exemple le Viêt Nam : moins de 40 % de la population y a accès à Internet. Et dans beaucoup de pays en voie de développement, les seuls accès Internet utilisent encore des technologies lentes, comme les connexions par RTC.

À l'opposé, on peut rencontrer la Norvège, pays ultra-connecté où 95 % de la population a un accès à Internet.

Dans l'ensemble, nous sommes aujourd'hui plus de 2 milliards d'internautes dans le monde, avec des connexions très disparates : paiement à la durée ou connexion illimitée, RTC bas débit ou fibre très haut débit, connexion fixe ADSL ou connexion mobile 3G, sur des ordinateurs, des smartphones, des tablettes... Internet est un joyeux écosystème où différents environnements hétérogènes se côtoient et échangent des données à la vitesse de la lumière, permettant de rapprocher des gens à deux bouts du monde, équipés d'appareils fondamentalement différents. ■



Pourcentage des foyers équipés d'un accès Internet (= données estimées.)*

À savoir

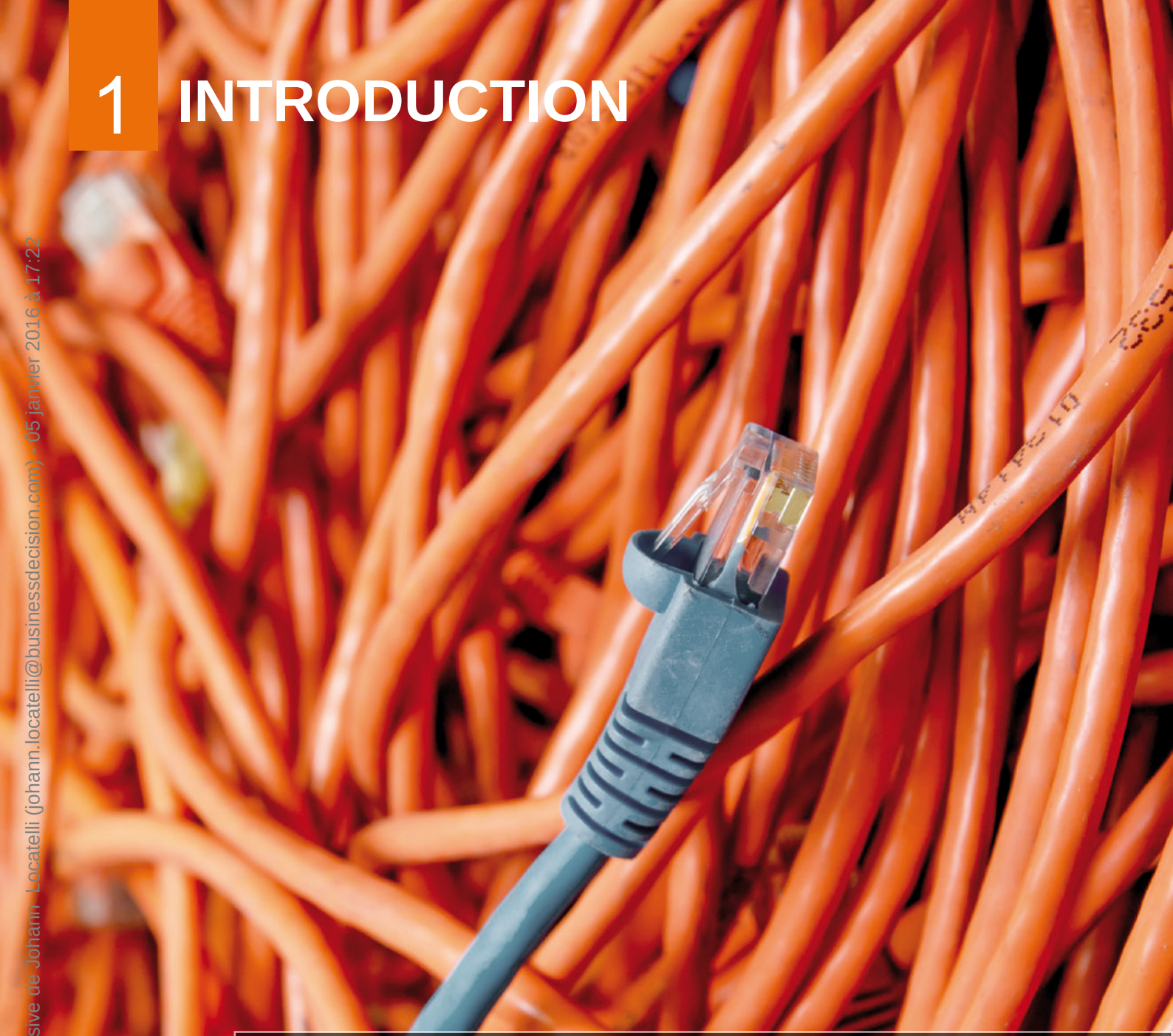
INTERNET EN CHIFFRES...

Selon le dernier rapport de l'Union Internationale des Télécommunications, « The World in 2013 : ICT Facts and Figures », en 2013, plus de 2,7 milliards de gens utilisent Internet, ce qui correspond à 39 % de la population mondiale. Parmi eux, on distingue 37 % de femmes et 41 % d'hommes.

L'Europe est la région la plus « connectée », avec 75 % d'accès pour 100 habitants, suivie de près par les Américains (61 %) ; en Afrique, seuls 16 % de la population utilisent Internet. En 2013, 41 % des foyers dans le monde sont connectés, dont la moitié sont situés dans des zones en voie de développement.

Source : http://www.itu.int/

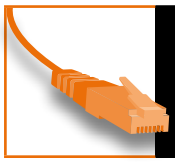
1 INTRODUCTION



HÉBERGER SES SERVICES

Pour proposer un ou plusieurs service(s) sur Internet, ceux-ci doivent être hébergés. Plusieurs approches sont possibles, à chacun de choisir celle qui lui convient le plus : cela va des services « clés en main » à l'auto-hébergement, en passant par l'hébergement mutualisé, les serveurs dédiés et les VPS...

Se faire une place sur Internet, c'est bien, mais encore faut-il arriver à juger ses besoins et l'énergie qu'on compte réellement investir, afin de choisir une solution adaptée parmi les cinq que nous vous présentons ici...



1 Les services « clés en main »

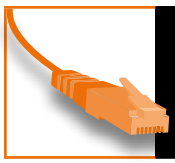
Voilà sans doute la solution la plus rapide et la plus simple pour vous créer une présence sur le web. Il s'agit de services proposés souvent gracieusement par des sites comme **blogger.com**, **over-blog.com**, **wordpress.com** ou de nombreux autres, reposant sur une architecture centralisée, mutualisée et fixe. Vous ne pouvez alors pas choisir l'application utilisée et serez limité en termes de personnalisation.

Bien entendu, chaque fournisseur met en avant l'aspect gratuit du service et la souplesse de configuration grâce à une interface d'administration web ergonomique. Ce n'est qu'ensuite qu'on découvre les limitations, qui peuvent parfois être lourdes et qui sont très variables selon le fournisseur du service :

- le fait de ne pas pouvoir ajouter d'autres services (galerie d'images, webmail, etc.) ;
- la limitation en termes d'espace de stockage pour vos médias ;
- la saturation du site par des publicités ;
- l'impossibilité d'utiliser son propre nom de domaine ;
- l'indisponibilité de l'interface de gestion de contenu en cas de maintenance (fréquent avec certains fournisseurs).

La démarche de ce genre de services consiste généralement à fournir un site gratuit, étayé par l'injection de publicités et reposant sur l'atteinte des limites en termes de besoins. Quand arrive le moment où l'offre gratuite ne suffit plus, il est possible de payer pour accéder à plus de liberté, de fonctionnalités, d'espace disque, de choix en termes de design... en bref, de lever ces limitations des offres gratuites.

C'est simple, rapide et séduisant, mais vous pouvez oublier votre liberté et en particulier celle de récupérer rapidement vos données et de les « poser » ailleurs.



2 L'hébergement mutualisé

La notion de mutualisation parle d'elle-même. C'est une solution consistant, pour un hébergeur, à réduire les coûts de maintenance en mutualisant un seul serveur pour plusieurs clients. Ainsi, personne n'a le contrôle total de la machine (à part l'hébergeur), mais tout le monde dispose d'un espace bien délimité. Des configurations spécifiques sont mises en œuvre, rendant totalement invisible l'espace réservé d'un client aux autres qui partagent la même machine.

En tant que bénéficiaire du service, on dispose ainsi de bien plus de liberté qu'avec un outil « clés en main » gratuit ou payant. Le contenu du site est de notre responsabilité (légale comme technique) et

on peut appliquer l'application web de son choix... ou presque. En effet, bien que le contenu soit librement modifiable, on ne peut guère influer sur la configuration du serveur en elle-même. Ceci implique que le paramétrage du serveur est dépendant des offres proposées : les langages installés et activés (surtout PHP), les bases de données (nombre et taille), et tous les systèmes propres à la configuration d'un serveur HTTP (optimisation, limitation, personnalisation).

L'hébergement mutualisé est généralement très accessible, avec des tarifs démarrant à quelques euros par mois. On peut ainsi disposer d'un espace web avec quelques comptes e-mail et une base de données pour une trentaine d'euros par an. Les hébergeurs proposant ce genre de services ne manquent pas. On pourra citer 1&1, Amen, Nerim, OVH, iKoula, Online, Gandi, etc. Ils se livrent une guerre commerciale sans fin dans un scénario où quelques leaders dominent le marché et les autres se spécialisent en apportant un service bien spécifique.

Les offres varient principalement en fonction de l'espace à disposition et du nombre de bases de données utilisables, ainsi que du trafic autorisé par mois. Le reste des éléments tient dans les services supplémentaires, la qualité du service, la vitesse et la pertinence des réponses du support technique, la performance et la fiabilité de la configuration.

Cette solution est bien plus ouverte que les solutions « clés en main », puisque vous téléchargez vous-même les fichiers de votre site (HTML, PHP, images, vidéos...) directement sur le disque du serveur, généralement par l'intermédiaire d'un client FTP. L'accès à la base de données se fera avec une interface web, qui sera très souvent **PHPMyAdmin**. Il vous sera possible, dans une certaine mesure, de moduler la configuration du serveur web pour votre hébergement par l'intermédiaire de fichiers `.htaccess`.

L'utilisation d'un hébergement mutualisé va généralement de pair avec un dépôt de nom de domaine, les fournisseurs proposent souvent les deux services au sein d'une seule offre. Enfin, comme vous n'avez qu'en partie le contrôle de certains aspects du serveur web, il vous faudra porter toute votre attention sur les langages utilisables et en particulier la configuration de PHP.

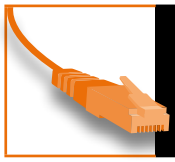
Chaque hébergeur liste les fonctionnalités supportées dans son offre commerciale avec, parfois, la page générée par la commande PHP `phpinfo()`. Ainsi, des fonctionnalités comme la gestion des images (GD), le support multilingue (gettext), la prise en charge de MySQL ou d'un autre SGBD, le chiffrement (SSL) et la gestion du XML doivent être vérifiés afin de ne pas avoir de mauvaise surprise lors de l'installation d'applications web.

Avec un tel hébergement, vous gardez la main sur l'applicatif que vous installez, vous êtes maître de vos données, vous n'avez pas à vous soucier de la gestion des logiciels et du matériel... En contre-partie, vous êtes limité par les fonctionnalités proposées par l'hébergeur et ne pouvez déployer que des applications web, *exit* les autres usages.

Définition

HOUSING

L'étape « ultime », après le serveur dédié, est l'hébergement de machine, appelé « housing ». Là, vous ne louez plus la ou les machine(s), mais les emplacements qu'elle(s) occupe(nt) dans un armoire (baie) installée dans un « datacenter » ; à vous de fournir le matériel qui y sera placé. Cette solution permet une très grande flexibilité (choix des serveurs, etc.) tout en plaçant l'ensemble dans un environnement dédié à l'hébergement de serveurs, avec une alimentation électrique redondante, une climatisation, etc.

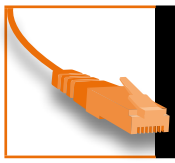


3 Serveur dédié

Nous voici un cran au-dessus. À ce stade, vous avez le contrôle sur la totalité du système d'exploitation du serveur, les services exécutés, les applications installées, web ou autres. Ceci implique que vous devez disposer des compétences techniques pour administrer un système (souvent une distribution GNU/Linux) à distance. En effet, la seule différence entre un serveur dédié et l'une de vos propres machines tient dans le fait que le serveur est loué (souvent au mois) et qu'il n'est pas à portée de main.

Un grand nombre d'hébergeurs proposent ce type d'offres. La gamme de prix est très vaste, elle débute à 10 € par mois (« petits » serveurs d'entrée de gamme) pour atteindre plusieurs centaines d'euros pour les configurations les plus musclées. Le tarif dépend principalement de la configuration matérielle proposée dans l'offre (processeur, mémoire vive, disque dur) : à la date d'écriture de cet article, on retrouve au plus bas de l'échelle (à 10 € par mois) un processeur Intel Atom mono-cœur avec 2 Go de RAM et 160 Go de disque dur et ça peut aller jusqu'à 40 cœurs Intel Xeon E7 4870 avec 1 To de RAM et 7,2 To de disque SAS ultra-rapide (pour environ 2000 € par mois).

Le serveur dédié est généralement à considérer comme une solution d'entreprise ou, du moins, de spécialiste. La configuration du système, du serveur web, du réseau et des différents éléments assurant la bonne marche de l'ensemble est entièrement à la charge du client. Certains hébergeurs fournissent des solutions pré-installées de distributions GNU/Linux, voire de systèmes Windows. Il est également possible de configurer le serveur dédié comme une machine de bureau distante, avec un bureau graphique, comme un PC standard.



4 VPS

Le VPS (*Virtual Private Server*) est une alternative au serveur dédié : plutôt que de fournir un serveur matériel complet, l'hébergeur propose un environnement virtualisé : vous partagez une machine physique avec d'autres utilisateurs, mais vous avez accès à votre propre système et avez votre propre adresse IP.

De cette manière, vous n'avez pas à gérer le matériel, l'adresse IP, etc. L'inconvénient est alors que vous ne pouvez pas mettre en place certaines choses ; selon la technologie utilisée par l'hébergeur, il est possible que vous n'ayez pas accès au noyau Linux, vous ne pouvez pas le changer, charger des modules, configurer le pare-feu...

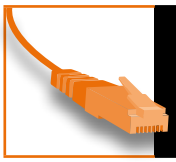
De nombreuses offres de serveurs virtuels privés ont été renommées « cloud » ces derniers temps ; en effet, les VPS – qui existent depuis de nombreuses années – ce n'est rien d'autre que le cloud avant l'heure : on ne gère pas le matériel, l'instance que l'on utilise est stockée dans un data center que nous n'avons pas besoin de connaître ;

tout l'aspect « matériel » devient totalement transparent. Notons que les solutions « cloud » sont souvent facturées avec des grilles très fines : les tarifs annoncés peuvent descendre jusqu'à 1 centime de l'heure.

Les VPS et les serveurs « cloud » présentent deux avantages :

- On peut commencer avec un tout petit VPS, qu'on fera évoluer dynamiquement au fur et à mesure, sans avoir besoin de réinstaller son système ;
- Certains hébergeurs proposent des offres à des prix défiant toute concurrence, commençant à moins de 5€ par mois.

Attention toutefois aux limitations de ces solutions : à performance égale, les serveurs dédiés sont souvent moins onéreux.



5 Auto-hébergement

Avec les débits en téléversement (*upload*) qui s'améliorent – notamment grâce à la fibre optique, au débit symétrique – l'auto-hébergement commence à être une option intéressante. Avec l'auto-hébergement, vous êtes le maître de l'intégralité des éléments, physiques ou logiciels : il s'agit là de dédier une machine personnelle, dans son propre logement, au rôle de serveur Internet. Il y a bien sûr des contraintes :

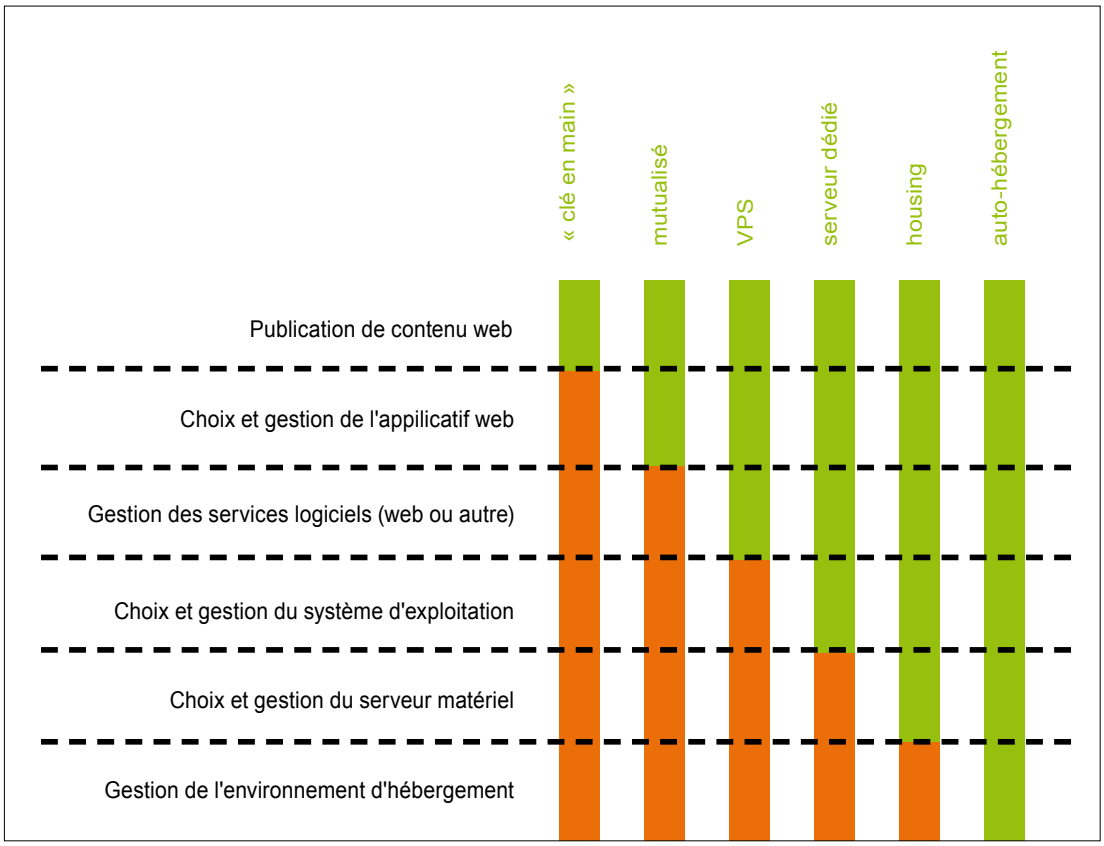
- Chez certains fournisseurs d'accès, l'adresse IP n'est pas fixe, il faut alors utiliser un service tel que DynDNS ou NoIP afin de faire pointer un nom DNS vers cette adresse variable ;
- Si la connexion n'offre pas assez de débit en téléversement ou est instable, le serveur risque d'être souvent indisponible et les difficultés de navigation Internet au sein du foyer risquent de s'accentuer ;
- L'ouverture de certains services est parfois explicitement interdite dans le contrat qui vous lie à votre fournisseur d'accès ;
- Il est nécessaire de mettre en place une infrastructure permettant de garantir la stabilité du système et la sécurité des données (coupure de courant, incendie, surchauffe, permanence du service) ;
- La présence du serveur sur le réseau local du foyer peut poser des problèmes de sécurité informatique, qui ne peuvent être résolues que par la mise en place d'une infrastructure en plusieurs réseaux avec pare-feu de réseau, complexifiant encore l'infrastructure.

Cette solution est en fait valable dans deux cas :

- Soit pour ceux qui veulent « bidouiller », qui veulent se plonger dans ce sujet sans retenue et qui ont envie de gérer une telle infrastructure sur leur temps libre ;
- Soit pour les entreprises qui sont capables de dédier une salle aux serveurs, climatisée et avec alimentation électrique en redondance, avec une connexion Internet correctement dimensionnée.

6 Synthèse

Nous avons vu qu'à chaque type de besoin, une solution adaptée existe... Le schéma ci-dessous offre une synthèse de l'ensemble des solutions :



Synthèse des solutions d'hébergement

Le fait de disposer d'un espace sur Internet s'est généralisé et popularisé. Les solutions ne s'adressent plus aujourd'hui qu'aux professionnels. Les prix très abordables, conjugués à l'évolution des solutions en logiciels libres (comme WordPress par exemple), rendent la création d'un site web ou la fourniture d'un service à la portée de n'importe quel internaute qui décide de « s'y mettre » sérieusement.

Finalement, on s'orientera généralement vers deux solutions :

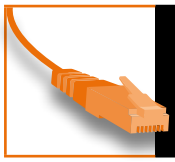
- L'hébergement mutualisé lorsque l'on veut uniquement mettre en place des solutions web et que l'on n'a pas envie de gérer un système ;
- Un serveur dédié lorsque l'on souhaite avoir un maximum de maîtrise pour un coût réduit.

Cela n'empêche pas, bien sûr, de mettre en place un serveur dans son propre réseau privé, pour apprendre à s'en servir avant de se lancer ! ■

1 INTRODUCTION

SERVICES ET PORTS

Une fois une solution d'hébergement choisie, il faut la gérer. Dans le cas des services « clés en main » c'est simple : il suffit de publier son contenu ; dans le cas de l'hébergement mutualisé, ce n'est pas compliqué : il suffit de placer ses fichiers (généralement PHP) au bon endroit et roulez jeunesse ! Mais dans les autres cas (dédié, VPS, housing, auto-hébergement), il faut savoir gérer un serveur et notamment comprendre ce que sont les services et les ports.

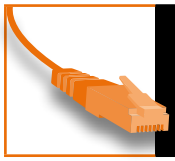


1 Qu'est-ce qu'un service ?

Un service, c'est tout simplement *ce qui tourne sur le serveur*. Beaucoup font la confusion entre « un serveur » et « un serveur web » ; beaucoup croient que tout ce qu'on peut publier sur Internet, ce sont des sites web. Non !

Il faut dire qu'on ne fait pas beaucoup d'efforts pour faciliter la compréhension de tout cela. Essayons alors de résumer les différents termes :

- Un serveur (matériel), c'est une machine allumée 24h/24 qui dessert un ou plusieurs service(s) ;
- Un serveur (logiciel), c'est un logiciel qui fonctionne 24h/24 sur un serveur (matériel) et qui dessert un service ;
- Un service, c'est ce qui est desservi par le serveur (logiciel) – et, par extension, qui est desservi par le serveur (matériel) – mais on parle parfois également de service lorsque l'on évoque le logiciel lui-même ;
- Un serveur (logiciel) est décomposé en un ou plusieurs processus ; ce sont eux qui *écoutent* et qui *répondent* aux requêtes des clients ;
- Un client, c'est une entité distante qui émet des requêtes à destination du serveur.



2 Qu'est-ce qu'un port ?

Les communications en réseau reposent sur différents **protocoles**. Pour chaque communication, un certain nombre de protocoles sont mis en œuvre et s'« empilent » sous forme de couches.

Parmi ces protocoles, on retrouve **IP** (*Internet Protocol*). C'est lui qui permet à des millions de systèmes différents de se comprendre sur la planète ; vous avez certainement déjà entendu parler d' « adresse IP » : chaque ordinateur a une adresse IP, qui permet de le retrouver et de dialoguer avec lui, quelle que soit sa marque, quel que soit le modèle de sa carte réseau, quelle que soit la tension délivrée par son réseau électrique, ...

Mais le protocole IP ne se suffit pas à lui-même. Par-dessus le protocole IP, deux protocoles sont très courants : **TCP** et **UDP**. TCP est un protocole de communication en mode connecté : il s'assure que chaque paquet envoyé est bien reçu par son destinataire. Avec UDP, la bonne réception des informations n'est pas vérifiée ; ce protocole est donc moins sûr quant au transfert de données, mais il est bien plus léger en termes de consommation de bande passante. Chacun de ces protocoles est adapté à certains usages. Le protocole TCP étant le plus utilisé, on parle souvent de **TCP/IP** sur Internet.

Ces deux protocoles ont un point commun, c'est le concept de **port**. Les ports permettent à un ordinateur d'effectuer plusieurs communications en même temps, chaque communication utilisant un port. Les ports sont numérotés de 1 à 65535 (codage sur 16 bits), autorisant théoriquement plus de 65000 communications simultanées. S'il n'y avait pas cette notion de port, une seule communication pourrait avoir lieu en même temps : on ne pourrait pas, par exemple, être simultanément connecté à une messagerie instantanée, lire une page web et écouter de la musique en *streaming*.

Définition

Le modèle OSI est le standard qui définit les différentes couches ; celles-ci sont au nombre de 7 : la couche physique, la couche liaison, la couche réseau, la couche transport, la couche session, la couche présentation, la couche application.

Cependant, il n'y a pas toujours 7 protocoles en œuvre, car certains d'entre eux représentent plusieurs couches. Par exemple, pour une machine connectée en réseau filaire derrière une « box », le protocole Ethernet constitue la couche physique et la couche liaison.

3 Lien entre services et ports

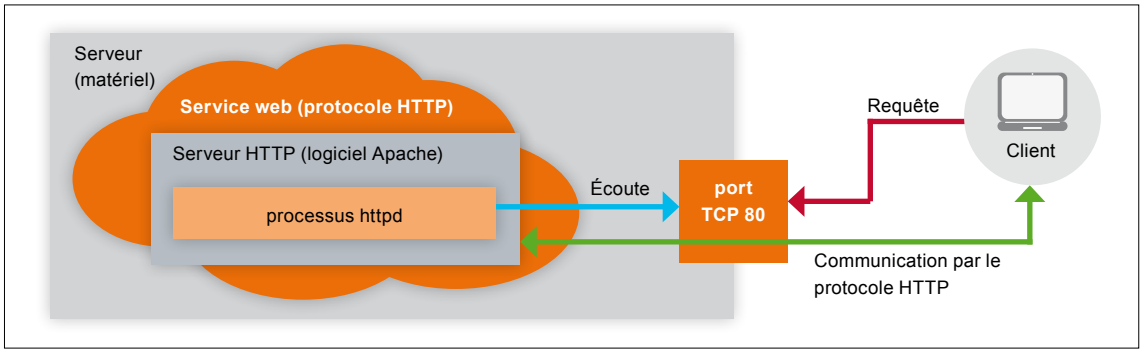
Chaque service réseau est en écoute sur un port donné, pour un protocole donné. C'est ce qui permet à n'importe quel client de communiquer avec un serveur en demandant un service précis.

Par exemple, les sites web sont desservis avec le protocole **HTTP** ; ce dernier utilise le protocole TCP pour communiquer (le serveur web a besoin de savoir que les pages sont bien reçues) et son port par défaut est le port **80**. Lorsque l'on tape l'adresse d'un site web dans la barre d'adresse d'un navigateur, celui-ci va donc se connecter au port 80 du serveur distant.

Les services peuvent être en écoute sur d'autres ports : on peut très bien avoir un serveur HTTP en écoute sur le port TCP 81, auquel cas il faudra l'indiquer au navigateur web avec la syntaxe suivante : **http://www.example.com:81**.

La plupart des protocoles utilisés sur Internet sont associés à un port particulier, afin de faciliter les échanges sur Internet ; par exemple, le protocole FTP (transfert de fichiers) utilise les ports TCP 20 et 21, HTTP (sites web) utilise le port 80, IMAP (messagerie) le port 143, ...

Vous pouvez retrouver l'association entre les services et les numéros de ports dans le fichier `/etc/services` de n'importe quel système UNIX (et donc, dans n'importe quelle distribution Linux).



Résumé des concepts de serveur, service et port

4 Observer les ports ouverts

Vous pouvez à tout moment savoir quels sont les ports en cours d'utilisation sur un ordinateur sous Linux, que ce soit un serveur ou un poste de travail. Cela se fait avec la commande `ss`. Cette commande peut prendre différents arguments, par exemple :

- `-t` : affiche les ports TCP ;
- `-u` : affiche les ports UDP ;
- `-l` : affiche les ports en écoute (par défaut, cette commande n'affiche que les ports en cours d'utilisation) ;
- `-a` : affiche tous les ports (en écoute ou en utilisation) ;
- `-p` : affiche le processus lié à chaque port ;
- `-n` : affiche les adresses IP et les numéros de ports au lieu des noms DNS et des noms de services.

Pour visualiser les ports en écoute, on pourra par exemple utiliser la commande **ss -tulnp**, qui peut donner le résultat suivant :

Terminal

```
# ss -tulnp
Netid State    Recv-Q Send-Q     Local Address:Port  Peer Address:Port
udp   UNCONN   0      0          88.191.185.95:123              *:*
users:(("ntpd",22126,19))
udp   UNCONN   0      0              127.0.0.1:123              *:*
users:(("ntpd",22126,18))
udp   UNCONN   0      0                    *:123              *:*
users:(("ntpd",22126,16))
udp   UNCONN   0      0                   ::1:123             :::*
users:(("ntpd",22126,21))
udp   UNCONN   0      0   fe80::d6ae:52ff:fec7:93ad:123       :::*
users:(("ntpd",22126,20))
udp   UNCONN   0      0                   :::123              :::*
users:(("ntpd",22126,17))
tcp   LISTEN   0      128                 :::22               :::*
users:(("sshd",2162,4))
tcp   LISTEN   0      128                  *:22               *:*
users:(("sshd",2162,3))
tcp   LISTEN   0      100                 :::25               :::*
users:(("master",27470,13),("smtpd",19964,7))
tcp   LISTEN   0      100                  *:25               *:*
users:(("master",27470,12),("smtpd",19964,6))
tcp   LISTEN   0      128                 :::443              :::*
users:(("nginx",7772,61),("nginx",7771,61),("nginx",7770,61),("nginx",
7769,61),("nginx",7768,61))
tcp   LISTEN   0      50              127.0.0.1:3306             *:*
users:(("mysqld",15233,11))
tcp   LISTEN   0      128                 :::80               :::*
users:(("nginx",7772,60),("nginx",7771,60),("nginx",7770,60),("nginx",
7769,60),("nginx",7768,60))
```

Dans cet exemple (tiré d'un serveur réel), on constate un certain nombre de services :

- Le processus **ntpd** (serveur de temps NTP) en écoute sur le port UDP 123, en IPv4 et en IPv6, sur toutes les interfaces ;
- Le processus **sshd** (serveur SSH) en écoute sur le port TCP 22, en IPv4 et en IPv6 ;
- Le processus **master** (serveur SMTP de Postfix) en écoute sur le port TCP 25, en IPv4 et en IPv6 ;
- Le processus **nginx** (serveur HTTP) en écoute sur les ports TCP 80 et 443, en IPv6 ;
- Le processus **mysqld** (SGBDR MySQL) en écoute sur le port TCP 3306, en IPv4 sur l'interface « localhost ».

Pour consulter la liste de l'ensemble des ports ouverts et des communications en cours et les processus liés, on peut utiliser la commande **ss -ap**. Nous ne donnerons pas ici d'exemple pour la bonne et simple raison que cette commande est très verbeuse : 150 lignes sur le serveur utilisé comme exemple !

N'hésitez pas à utiliser cette commande sur votre serveur pour voir par vous-même : elle ne peut rien casser ! ■

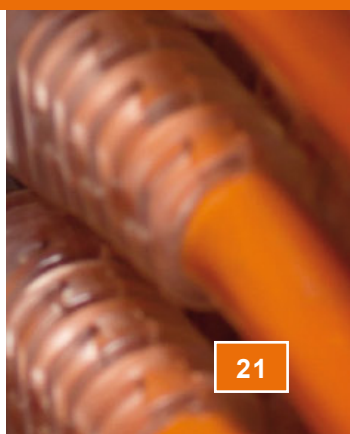
À savoir

ÉCOUTE EN IPV4 ET EN IPV6

Le serveur que nous considérons en exemple est configuré pour écouter en IPv4, ainsi qu'en IPv6. Sous Linux, les services desservis en IPv6 sont automatiquement également desservis en IPv4 ; c'est pourquoi on voit `nginx` n'écouter qu'en IPv6 : il dessert tout de même les requêtes en IPv4 grâce à cette fonctionnalité propre au noyau Linux.

Ce comportement peut être modifié par le pseudo-fichier `/proc/sys/net/ipv6/bindv6only` : si la valeur 1 est écrite dans ce fichier, un port en écoute en IPv6 ne sera pas desservi en IPv4. Pour rendre cela définitif, il faut ajouter la ligne suivante au fichier µ `/etc/sysctl.conf` :

```
net.ipv6.bindv6only = 1
```



1 INTRODUCTION

SÉCURITÉ ET PARE-FEU

Encore une fois, avec les solutions « clés en main » et l'hébergement mutualisé, la sécurité des systèmes est gérée par le fournisseur. Mais dans les autres cas, les plus intéressants, il va vous falloir gérer tout cela vous-même et vous frotter à des notions qui peuvent devenir complexes, à commencer par le filtrage de ports.

1 Utilité d'un pare-feu

Le rôle d'un pare-feu (*firewall* en anglais) est de filtrer ou modifier les données entrant ou sortant de l'ordinateur du ou vers un réseau. Selon la machine sur laquelle il est placé, il sera capable d'effectuer différentes tâches. On distingue notamment :

- d'une part le **pare-feu local**, qui permet de filtrer ou modifier les paquets entrant ou sortant d'un ordinateur (serveur ou poste de travail, par exemple) ;
- d'autre part le **pare-feu de réseau**, qui permet de filtrer les communications entre deux réseaux ou plus ; c'est le rôle que tiennent les « box Internet » que l'on trouve dans la plupart des foyers français aujourd'hui.

Les opérations les plus couramment effectuées sont les suivantes :

- Filtrage de paquets réseau selon certains critères, comme le port de destination ou l'adresse IP source ;
- Redirection de paquets entrants vers un autre serveur (**DNAT**, *Destination Network Address Translation*) ;
- Translation d'une adresse privée vers une adresse publique, pour permettre à une machine n'ayant pas d'adresse publique de communiquer avec une autre machine sur Internet (**SNAT**, *Source Network Address Translation*, voir également **MASQUERADE**).

N'importe quelle distribution Linux peut être utilisée dans le rôle de routeur ou de pare-feu de réseau (on peut cependant préférer des distributions conçues spécifiquement pour cet usage, comme Smoothwall ou IPCop). L'ordinateur en question devient alors un point de passage obligatoire pour toutes les communications du réseau qu'il protège.

Nous allons nous focaliser sur le pare-feu local, qui correspond au besoin d'un serveur dédié ; dans le cas du *housing*, on voudra probablement mettre également en place un pare-feu de réseau, mais ce n'est pas l'objet de cet article.

2 Netfilter : le pare-feu de Linux

Le noyau Linux en version 2.0 (en 1996) comprenait **ipfwadm** ; il a été suivi du noyau 2.2 (en 1999), qui proposait **ipchains**. Pour Linux 2.3 (en 2000), le pare-feu de Linux a encore une fois été complètement réécrit ; cette réécriture se nomme **Netfilter**. Depuis ce moment-là, les évolutions du pare-feu ne passent plus par des réécritures et nouvelles implémentations, mais par une amélioration continue de Netfilter.

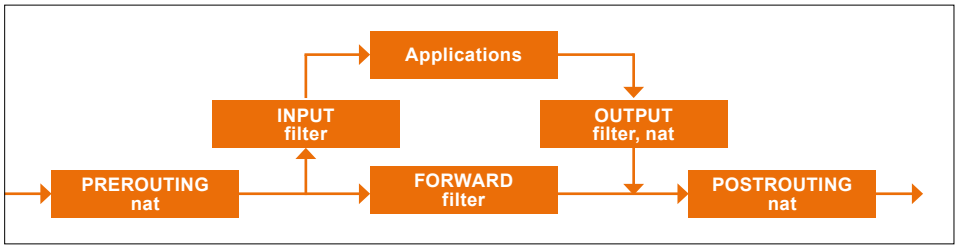
Les différents logiciels que l'on peut utiliser pour gérer le pare-feu de Linux ne sont en fait que des frontaux : ils ne ré-implémentent pas le filtrage en lui-même, qui est toujours géré par Netfilter. Il est toutefois possible d'étendre les capacités de Netfilter grâce à différentes extensions, ainsi qu'à la possibilité de passer le paquet en espace utilisateur, permettant à un autre logiciel de le traiter et de le réinjecter dans le noyau.

Lorsqu'un paquet réseau entre dans le noyau Linux, il parcourt un certain nombre de chaînes. Dans ces chaînes sont définies différentes règles ; si le paquet correspond aux critères d'une règle rencontrée, il est alors traité en conséquence. Par exemple, on peut créer une règle qui bloque tout paquet entrant sur un port donné : cette règle se placerait dans la chaîne appelée **INPUT**. En réalité, ces chaînes sont présentes dans une ou plusieurs table(s), notamment **nat** et **filter**.

À savoir

ROUTEUR OU PARE-FEU ?

Les termes « routeur » et « pare-feu de réseau » désignent le même rôle, ce sont simplement deux approches différentes de ce même rôle. Lorsque l'on parle de routeur, on imagine plusieurs interfaces réseau sur un même boîtier et la gestion de la communication entre les différents réseaux – le routage. Lorsque l'on parle de pare-feu, on imagine plutôt peu d'interfaces, mais un système pseudo-intelligent qui filtre les paquets qui le traversent, afin de protéger le ou les réseau(x) qu'il dessert. Mais on peut très bien imaginer un routeur capable de filtrer les paquets ou un pare-feu capable de gérer de nombreuses interfaces réseau... Le noyau Linux, en tout cas, est capable d'endosser les deux rôles.



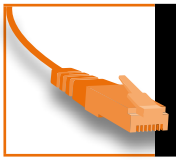
Succession des chaînes Netfilter

Les chaînes les plus importantes sont les suivantes, classées par tables :

- Dans la table **`filter`** sont présentes les chaînes dont l'objectif est de **filtrer** les paquets :
  - **`INPUT`** : cette chaîne voit passer tous les paquets qui sont à destination de l'hôte local – c'est ici que l'on va définir les ports entrants à bloquer ;
  - **`OUTPUT`** : cette chaîne voit passer tous les paquets sortants de l'hôte local – on pourra ici interdire les communications vers l'extérieur ;
  - **`FORWARD`** : cette chaîne voit passer tous les paquets qui transitent par l'hôte local, lorsque celui-ci est placé comme un routeur, comme un pare-feu de réseau ;
- Dans la table **`nat`** sont présentes les chaînes dont l'objectif est de **modifier** les paquets :
  - **`PREROUTING`** : cette chaîne voit passer tous les paquets qui entrent dans le système – c'est dans celle-ci que l'on mettra une règle de redirection (**`DNAT`**), pour renvoyer un paquet vers un autre serveur par exemple (c'est ce que l'on fait lorsque l'on met en place une redirection de port sur une « box »),
  - **`OUTPUT`** : cette chaîne permet de modifier tous les paquets envoyés par le système (par exemple pour rediriger une requête locale vers un autre serveur que celui demandé, par une règle de **`DNAT`** également),
  - **`POSTROUTING`** : cette chaîne voit passer tous les paquets qui sortent du système, qu'ils en émanent ou qu'ils le traversent – elle permet par exemple de faire du **`SNAT`** ou du **`MASQUERADE`**, nécessaire pour permettre à un PC d'un réseau local de communiquer avec un serveur tiers, en utilisant l'adresse IP du pare-feu de réseau.

Sur un pare-feu local (que l'on met en place sur un serveur dédié, par exemple), on se concentrera sur la chaîne **`INPUT`** (pour filtrer les paquets entrants) et éventuellement sur la chaîne **`OUTPUT`** (si on ne fait pas confiance aux logiciels qui fonctionnent localement).

À l'opposé, sur un pare-feu de réseau, on travaillera surtout sur la chaîne **`FORWARD`** (pour filtrer les paquets qui le traversent) et éventuellement **`PREROUTING`** et **`POSTROUTING`** pour les règles de **`DNAT`**, **`SNAT`** et **`MASQUERADE`**.



3 iptables

Les règles de pare-feu du noyau Linux sont gérées grâce à la commande **`iptables`**, qui communique avec Netfilter pour leur manipulation. Cette commande peut prendre de très nombreux arguments différents, un livre entier pourrait y être dédié.

Au début du XXème siècle, on mettait en place un **script iptables**, qui était simplement un enchaînement des commandes **`iptables`** que l'on avait utilisées pour définir les règles du pare-feu ; ce script était mis en place pour être exécuté au démarrage du système, après l'activation des interfaces réseau. On manipulait chaque règle manuellement lorsque l'on avait besoin de nouvelles autorisations ou interdictions, et on devait

s'assurer que le script en question était bien à jour. De plus, les commandes **iptables** manipulant les règles dans un ordre précis, il fallait s'assurer qu'elles soient mises en place dans le bon ordre par le script.

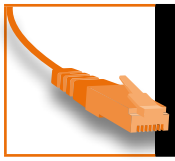
Un script **iptables** pouvait par exemple ressembler à cela :

Fichier

```
iptables -P INPUT DROP
iptables -P OUTPUT ACCEPT
iptables -P FORWARD DROP
iptables --flush
iptables --delete-chain
iptables -A INPUT -i lo -j ACCEPT
iptables -A INPUT -i eth0 -m state --state
ESTABLISHED,RELATED -j ACCEPT
iptables -A INPUT -p tcp --dport 22 -j ACCEPT
iptables -A INPUT -p tcp --dport 80 -j ACCEPT
```

Ce script autorise tout paquet sortant et interdit tout paquet entrant, sauf les connexions en cours ainsi que les ports des protocoles SSH et HTTP.

De nos jours, on ne travaille plus comme ça : on utilise des interfaces – en ligne de commandes ou graphiques – qui permettent de largement simplifier la gestion des règles de pare-feu... Ces logiciels créent des ensembles de règles bien plus complexes et gérant mieux certains cas qu'un tel script ne permet simplement pas.



4 La sécurité, c'est aussi...

La sécurité d'un serveur, sur Internet ou sur un autre réseau, ne passe pas seulement par le pare-feu. D'ailleurs, un pare-feu n'est pas toujours nécessaire...

Un aspect bien plus important que le pare-feu pour la sécurité d'un serveur, c'est le maintien à jour de tout ce qui fonctionne dessus. C'est pourquoi il est important d'appliquer aussitôt que possible toute mise à jour de sécurité qui serait proposée par la distribution : c'est l'avantage des distributions Linux d'ailleurs, l'immense majorité des mises à jour se mettent en place facilement et rapidement.

Pensez toutefois également à maintenir à jour vos applicatifs web : si vous avez téléchargé un tel logiciel (par exemple le moteur de blog WordPress), installez toutes ses mises à jour pour éviter qu'un pirate exploite une faille qui aurait été corrigée entre-temps. Gérer un serveur, ce n'est pas que l'installer, c'est également **s'assurer qu'il reste à jour** !

Un autre point extrêmement important est **l'authentification** : si vous optez pour une identification par mot de passe, utilisez des mots de passe solides, compliqués à deviner et à « casser ». Pour vos connexions à des fins d'administration du serveur, vous pouvez aussi préférer l'authentification par clés SSH, qui offre une bien plus grande sécurité : pour corrompre l'accès au serveur, il faut alors réussir à vous voler votre clé privée (que vous conservez bien sûr précieusement) et la phrase de passe de cette clé. ■

À retenir

PARE-FEU : INUTILE ?

Sur un serveur parfaitement bien géré, les seuls services en écoute sont ceux qui sont nécessaires. Le rôle d'un pare-feu étant de bloquer l'accès aux ports qu'on ne veut pas desservir publiquement, si aucun autre port que le strict nécessaire n'est ouvert, alors un pare-feu est théoriquement inutile.

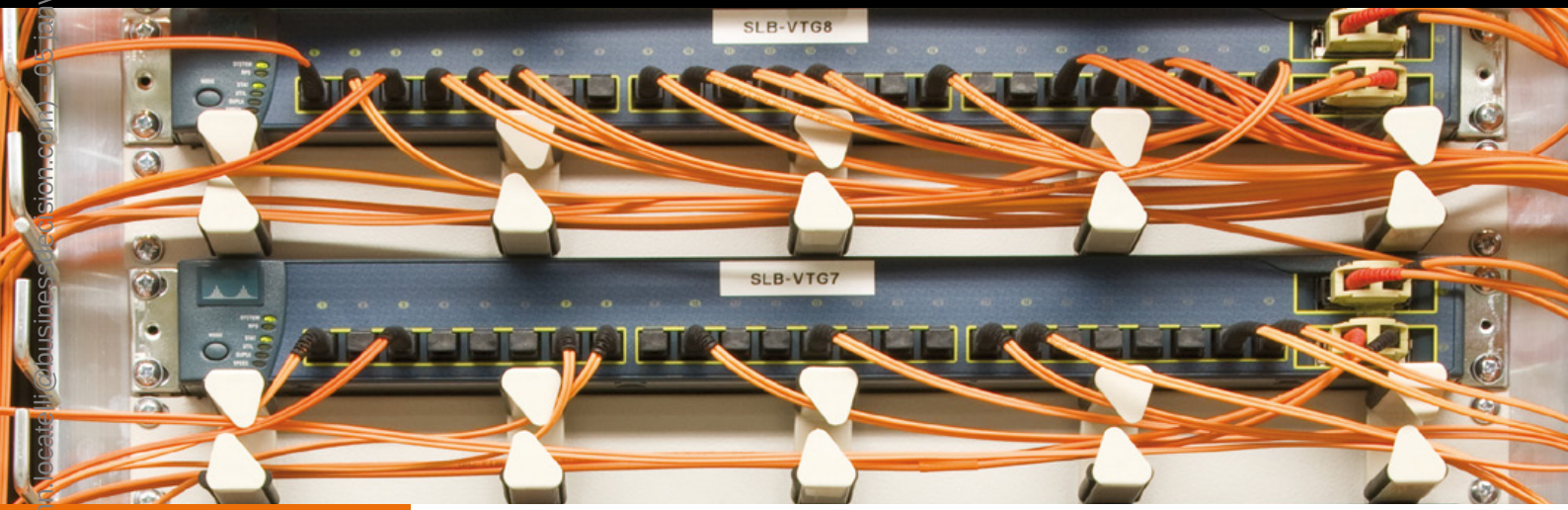
Certains affirmeront qu'il faut obligatoirement bloquer certains protocoles, comme ICMP, « par mesure de sécurité ». Il s'agit surtout d'empêcher un pirate de deviner quel est le système d'exploitation en fonctionnement sur un serveur pour mieux cibler ses attaques. Ce n'est rien d'autre que de la *sécurité par l'obscurité* : on cache ce qu'on a pour qu'un tiers ne puisse pas l'exploiter ; cette approche a prouvé ses failles à de nombreuses reprises.

On peut toutefois envisager la mise en place d'un pare-feu, même sur un serveur qui n'a que le strict nécessaire, pour se prémunir par exemple d'une ouverture de port accidentelle, dans le cadre d'une installation d'un nouveau service sur un serveur en production par exemple... Un pare-feu peut également être nécessaire lorsque l'on souhaite n'autoriser que certains équipements distants à accéder à des applications desservies en réseau, auquel cas on met généralement en place un filtrage par adresse IP source.

1 INTRODUCTION

UFW, LE PARE-FEU SIMPLIFIÉ

Dans l'article précédent, nous avons évoqué des logiciels permettant de simplifier la gestion de Netfilter, le pare-feu de Linux. Ufw (pour « Uncomplicated Firewall ») est l'un d'entre eux, créé pour Ubuntu mais également disponible pour d'autres distributions...



Comme son nom l'indique (*Uncomplicated Firewall*), ce logiciel a pour but de simplifier la gestion du pare-feu de Linux. Développé par Canonical pour Ubuntu à l'origine, celui-ci est maintenant proposé dans d'autres distributions Linux, par exemple Debian. C'est un logiciel en ligne de commandes, ce qui est particulièrement utile sur un serveur où, par définition, il n'y a pas d'interface graphique.

Par défaut, Ufw interdit tous les paquets entrants et autorise tous les paquets sortants ; il autorise également toutes les connexions en cours. C'est le comportement générique le plus habituel. Il ne reste alors qu'à autoriser les paquets entrants selon nos besoins.

1 Installer Ufw

Ufw n'est pas installé par défaut sous Debian, il faut donc l'installer :

Terminal

```
~# apt-get install ufw
Lecture des listes de paquets... Fait
Construction de l'arbre des dépendances
Lecture des informations d'état... Fait
Les paquets supplémentaires suivants seront installés :
  file libmagic1 mime-support python python-minimal python2.7
  python2.7-minimal
Paquets suggérés :
  python-doc python-tk python2.7-doc binfmt-support
```

À savoir

PARE-FEU DE RÉSEAU

Pour gérer un pare-feu de réseau, le logiciel Shorewall peut être préférable : paramétré par des fichiers de configuration, celui-ci permet de représenter les différents réseaux de manière logique et génère les règles Netfilter sous forme de commandes `iptables`, qu'il applique automatiquement à chaque redémarrage du système ou du service `shorewall` lui-même.

Il va bien plus loin qu'Ufw dans la définition des règles `iptables`, mais est bien sûr plus compliqué à prendre en main.

Terminal

```
Les NOUVEAUX paquets suivants seront installés :
  file libmagic1 mime-support python python-minimal python2.7
  python2.7-minimal ufw
Ø mis à jour, 8 nouvellement installés, Ø à enlever et Ø non mis à jour.
Il est nécessaire de prendre 5 113 ko dans les archives.
Après cette opération, 18,6 Mo d'espace disque supplémentaires seront utilisés.
Souhaitez-vous continuer [O/n] ?  o
[...]
```

2 Activer Ufw

Il suffit d'une commande pour mettre en place le pare-feu immédiatement :

Terminal

```
~# ufw enable
Command may disrupt existing ssh connections. Proceed with operation (y|n)? y
Firewall is active and enabled on system startup
```

Malgré l'alerte qui s'affiche, la connexion SSH en cours n'est pas coupée. Ouf ! Par défaut, Ufw met en place une règle qui autorise les connexions déjà ouvertes ; si vous avez touché aux fichiers dans **`/etc/ufw`**, assurez-vous que vous n'avez pas supprimé cette règle (voir la ligne **`-A [...] RELATED,ESTABLISHED -j ACCEPT`** dans **`/etc/ufw/before.rules`**).

3 Autoriser ou interdire un flux entrant (simple)

L'autorisation d'un flux entrant de manière simple se fait de la manière suivante :

Fichier

```
ufw <allow|reject|deny> <port>
```

Par exemple, pour autoriser les connexions SSH :

Terminal

```
~# ufw allow 22
Rule added
Rule added (v6)
```

4 Autoriser ou interdire un flux entrant (application connue)

Ufw a des ensembles de règles prédéfinies permettant d'autoriser des applications sans avoir à entrer manuellement les numéros de ports, par exemple.

Pour obtenir la liste des applications disponibles, utilisez la commande suivante :

Terminal

```
~# ufw app list
```

Ensuite, pour autoriser une application, c'est la même commande que plus haut :

Fichier

```
ufw <allow|reject|deny> <application>
```

Par exemple, pour autoriser l'application SSH (c'est une alternative à la règle `allow 22`) :

Terminal

```
~# ufw allow SSH
Rule added
Rule added (v6)
```

5 Autoriser ou interdire un flux entrant (règle complexe)

Il est possible de créer des règles plus complexes, permettant par exemple de filtrer par adresse IP. La syntaxe est alors la suivante :

Fichier

```
ufw <allow|reject|deny> [proto <protocole>] [from <adresse IP source> [port
<port source>]] [to <adresse IP destination> [port <port destination>]]
```

Par exemple, pour bloquer les requêtes SSH provenant de l'adresse IP 1.2.3.4 :

Terminal

```
~# ufw deny from 1.2.3.4 to any port 22
Rule added
```

6 Lister les règles en place

Ufw permet de lister les règles grâce à l'argument `status` :

Terminal

```
~# ufw status
Status: active
To                         Action      From
--                         ------      ----
22                         ALLOW       Anywhere
SSH                        ALLOW       Anywhere
22                         DENY        1.2.3.4
22                         ALLOW       Anywhere (v6)
SSH (v6)                   ALLOW       Anywhere (v6)
```

7 Supprimer une règle

Pour supprimer une règle, il suffit de mettre le mot-clé `delete` devant les arguments qui ont servi à la créer. Par exemple, pour supprimer la première règle mise plus haut (redondante avec la règle liée à l'application « SSH ») :

Terminal

```
~# ufw delete allow 22
Rule deleted
Rule deleted (v6)
```

8 Supprimer une règle selon son rang

Lorsque l'on veut supprimer une règle, on n'a souvent pas envie de rechercher la commande qu'on avait utilisée pour la mettre en place. On peut alors la supprimer avec son rang.

Commençons par afficher les règles avec leur numéro :

Terminal

```
~# ufw status numbered
Status: active
     To                         Action      From
     --                         ------      ----
[ 1] SSH                        ALLOW IN    Anywhere
[ 2] 22                         DENY IN     1.2.3.4
[ 3] SSH (v6)                   ALLOW IN    Anywhere (v6)
```

Ensuite, effaçons par exemple la règle bloquant les requêtes provenant de l'adresse 1.2.3.4 :

Terminal

```
~# ufw delete 2
Deleting:
 deny from 1.2.3.4 to any port 22
Proceed with operation (y|n)? y
Rule deleted
```

9 Lister les règles de manière détaillée

Avec l'option **verbose**, on peut lister les règles de manière détaillée, ainsi que les politiques par défaut :

Terminal

```
~# ufw status verbose
Status: active
Logging: on (low)
Default: deny (incoming), allow (outgoing)
New profiles: skip
To                         Action      From
--                         ------      ----
22/tcp (SSH)               ALLOW IN    Anywhere
22/tcp (SSH (v6))          ALLOW IN    Anywhere (v6)
```

10 Aller plus loin…

Ufw permet bien sûr des manipulations plus complètes que ces exemples.
Utilisez la commande suivante pour obtenir la liste des commandes disponibles :

Terminal

```
~# ufw help
```

Parmi les fonctions non abordées ici, on trouve par exemple :

- le changement des politiques par défaut ;
- la modification du niveau de journalisation (logs) ;
- la visualisation de rapports.

Voyez également la page de manuel d'Ufw pour tous les détails croustillants :

Terminal

```
man ufw
```

2

2 HTTP/WEB

À découvrir dans cette partie...

METTRE EN PLACE UNE ARCHITECTURE LAMP

Linux, Apache, MySQL, PHP : les quatre technologies que l'on associe le plus couramment dans le cadre d'un serveur web. C'est le point de départ de nombreux outils en ligne, il est donc nécessaire de savoir installer cet ensemble avant de mettre en place ce genre d'outils.

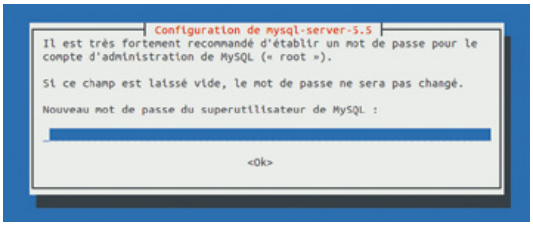
1 Installer les logiciels

Commençons par installer les paquets qui constituent cet ensemble :

Terminal

```
~# apt-get install apache2 mysql-server php5 php5-mysql libapache2-mod-php5
```

La procédure d'installation demande le mot de passe du super-utilisateur de MySQL. Cette procédure demande également la confirmation de ce même mot de passe.



Chacun renseignera le mot de passe de son choix, il est toutefois préférable de ne pas utiliser le même mot de passe que l'utilisateur **root**, pour renforcer la sécurité du système. Ce mot de passe sera à utiliser pour vous connecter au serveur MySQL en ligne de commandes.

Si le pare-feu Ufw a été activé, il faut ajouter une règle pour autoriser les requêtes HTTP :

Terminal

```
~# ufw allow WWW
```

2 Page de test PHP

Pour valider le bon fonctionnement de PHP sur ce serveur, créez le fichier **/var/www/test.php** avec la commande suivante :

Terminal

```
~# echo '<? phpinfo() ?>' > /var/www/test.php
```

Accédez ensuite à l'adresse **http://<adresse_IP_du_serveur>/test.php** à partir d'un navigateur web ; la page de confirmation suivante s'affiche :

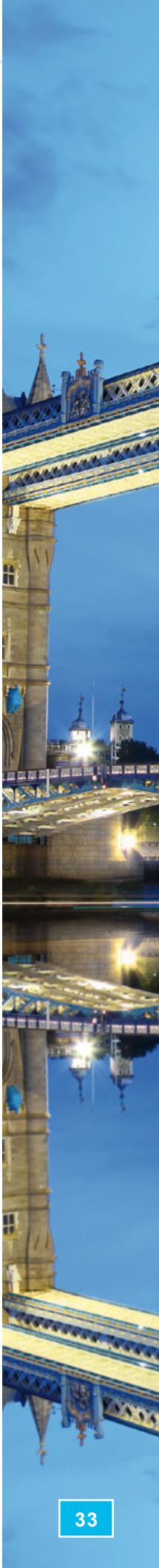
System	Linux gmlf 3.2.0-4-686-pae #1 SMP Debian 3.2.41-2+deb7u2 i686
Build Date	Mar 4 2013 15:55:09
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php5/apache2
Loaded Configuration File	/etc/php5/apache2/php.ini
Scan this dir for additional .ini files	/etc/php5/apache2/conf.d
Additional .ini files parsed	/etc/php5/apache2/conf.d/10-pdo.ini, /etc/php5/apache2/conf.d/20-mysql.ini, /etc/php5/apache2/conf.d/20-mysqli.ini, /etc/php5/apache2/conf.d/20-pdo_mysql.ini
PHP API	20100412
PHP Extension	20100525

On retrouve également plus bas, dans cette même page, la confirmation que le module MySQL pour PHP est bien installé. Cette page contient aussi des sections « mysqli », « PDO » et « pdo_mysql ».

3 HTTPS

3.1 Créer un certificat SSL

Pour desservir un site sécurisé par le protocole SSL, il faut tout d'abord créer un certificat SSL ; nous allons la placer dans le répertoire **/etc/ssl**.



Terminal
```
~# make-ssl-cert /usr/share/ssl-cert/ssleay.cnf /etc/ssl/certs/www.monsite.fr.crt
```

Cette commande demande tout d'abord le nom d'hôte du serveur – pour notre exemple, nous allons indiquer **www.monsite.fr**, chacun y indiquera le nom de son site.
Un second écran explique qu'il est possible de chiffrer plusieurs sites avec un même certificat SSL, grâce au champ **subjectAltName**. Enfin, les noms supplémentaires sont demandés – dans notre cas, nous ne nous en servons pas, ce champ reste vide.

Notons que cette méthode génère un certificat **auto-signé** : le navigateur web estimera que cela est peu sûr et affichera une alerte au visiteur. Une alternative est d'acheter un certificat auprès d'une autorité de certification, cela permettra de ne pas avoir de message d'erreur lors de l'accès au site.

3.2 Activer le module SSL sur Apache

Par défaut, Apache est configuré pour ne pas desservir le protocole HTTPS. Il faut donc activer le module idoine, avec la commande suivante :

Terminal
```
~# a2enmod ssl
```

3.3 Desservir un site en HTTPS

Nous pourrions activer simplement l'hôte virtuel en SSL par défaut d'Apache, celui-ci utilise un certificat auto-signé généré lors de l'installation d'Apache. Cependant, nous voulons utiliser notre propre certificat ; cette étape servira également de premier exemple de création d'un hôte virtuel pour Apache.

Créez un nouveau fichier **/etc/apache2/sites-available/www.monsite.fr-ssl** dont le contenu sera le suivant (il faut bien sûr remplacer **www.monsite.fr** par l'adresse du site qui sera desservi) :

Fichier
```
<VirtualHost _default_:443>
    ServerAdmin webmaster@monsite.fr
    DocumentRoot /srv/www.monsite.fr-ssl
    ErrorLog ${APACHE_LOG_DIR}/www.monsite.fr-ssl/error.log
    LogLevel warn
    CustomLog ${APACHE_LOG_DIR}/www.monsite.fr-ssl/access.log combined

    SSLEngine on
    SSLCertificateFile  /etc/ssl/certs/www.monsite.fr.crt
    BrowserMatch "MSIE [2-6]" \
      nokeepalive ssl-unclean-shutdown \
      downgrade-1.0 force-response-1.0
    BrowserMatch "MSIE [17-9]" ssl-unclean-shutdown
</VirtualHost>
```

Créez ensuite les répertoires évoqués dans ce fichier de configuration :

Terminal
```
~# mkdir /srv/www.monsite.fr-ssl
~# mkdir /var/log/apache2/www.monsite.fr-ssl
```

Puis, activez ce site web et redémarrez Apache :

Terminal

```
~# a2ensite www.monsite.fr-ssl
[...]
~# service apache2 restart
[...]
```

À partir de là, Apache desservira le contenu de `/srv/www.monsite.fr-ssl` si on se connecte au serveur en HTTPS : il ne reste plus qu'à y mettre le contenu du site.

Si le pare-feu Ufw a été activé, il faut ajouter une règle pour autoriser les requêtes HTTPS :

Terminal

```
~# ufw allow "WWW Secure"
```

3.4 Desservir plusieurs sites en HTTPS

Le point précédent indique une méthode pour desservir un unique site en HTTPS. Les navigateurs et serveurs web récents (sortis ces dernières années) permettent de desservir plusieurs sites en HTTPS sur un même serveur, dépassant ainsi une limitation historique du protocole SSL grâce à l'extension SNI (*Server Name Indication*).

Tout d'abord, activez les hôtes virtuels nommés sur le port 443 en ajoutant la ligne suivante au fichier `/etc/apache2/ports.conf` :

Fichier

```
NameVirtualHost *:443
```

Ensuite, modifiez le fichier `/etc/apache2/sites-available/www.monsite.fr-ssl` :

Fichier

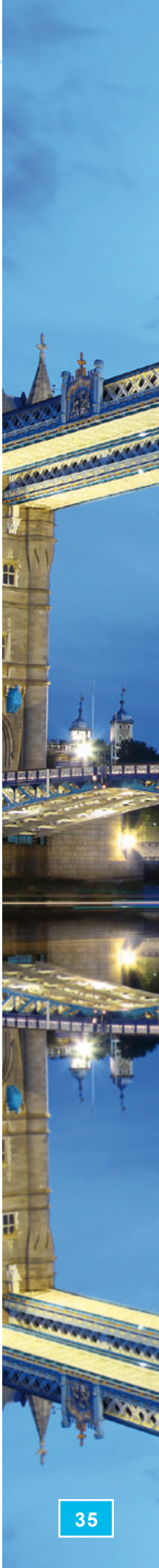
```
<VirtualHost *:443>
    ServerName www.monsite.fr
    ServerAdmin webmaster@monsite.fr
    DocumentRoot /srv/www.monsite.fr-ssl
    ErrorLog ${APACHE_LOG_DIR}/www.monsite.fr-ssl/error.log
    LogLevel warn
    CustomLog ${APACHE_LOG_DIR}/www.monsite.fr-ssl/access.log combined

    SSLEngine on
    SSLCertificateFile  /etc/ssl/certs/www.monsite.fr.crt
    BrowserMatch "MSIE [2-6]" \
      nokeepalive ssl-unclean-shutdown \
      downgrade-1.0 force-response-1.0
    BrowserMatch "MSIE [17-9]" ssl-unclean-shutdown
</VirtualHost>
```

Rechargez ensuite la configuration :

Terminal

```
~# service apache2 reload
```

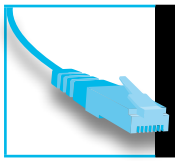
Il est maintenant possible d'ajouter d'autres hôtes virtuels par l'intermédiaire d'autres fichiers dans `/etc/apache2/sites-available`, d'autres commandes `a2ensite` et d'autres certificats SSL. ■



OPTEZ POUR UN SERVEUR WEB PLUS LÉGER : LIGHTTPD

Apache est peut-être le serveur HTTP le plus courant, mais il n'est pas le seul. Intéressons-nous à Lighttpd (que l'on nomme communément « Lighty »), un serveur web très léger...

Lighttpd étant un serveur web à part entière, il ne faut pas qu'Apache soit installé pour que celui-ci puisse fonctionner avec sa configuration par défaut : les deux logiciels voudront accéder au port 80 en même temps, ce qui créera un conflit. Soit il faut modifier la configuration de l'un des deux pour écouter sur un autre port, soit il ne faut pas installer Apache.



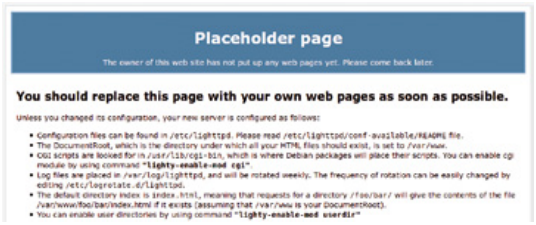
1 Installer Lighttpd

L'installation de Lighttpd se fait avec une simple commande :

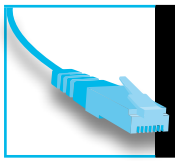
Terminal
```
~# apt-get install lighttpd
```

Lorsque l'on a choisi de protéger son serveur par un pare-feu avec Ufw, une règle doit être ajoutée afin d'autoriser les requêtes HTTP :

Terminal
```
~# ufw allow WWW
```



On peut vérifier le bon fonctionnement de Lighttpd en accédant simplement à l'adresse **http://<adresse_IP_du_serveur>/** à partir d'un navigateur web ; la page par défaut ci-contre s'affiche :



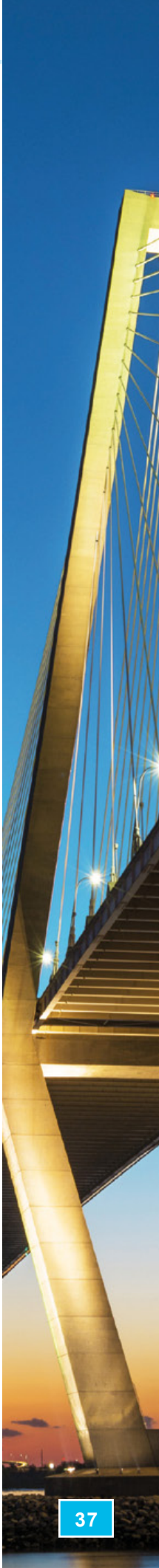
2 PHP avec Lighttpd

Lighttpd ne propose pas de module permettant d'interpréter du code PHP ; il faut passer par un intermédiaire, qui sera **FastCGI**. Le paquet Debian de Lighttpd propose un fichier de configuration spécifique pour simplifier la mise en place de FastCGI avec PHP. Nous allons donc installer l'interpréteur CGI de PHP et activer le fichier de configuration en question :

Terminal
```
~# apt-get install php5-cgi
[...]
~# lighty-enable-mod fastcgi-php
Met dependency: fastcgi
Enabling fastcgi-php: ok
Enabling fastcgi: ok
Run /etc/init.d/lighttpd force-reload to enable changes
~# service lighttpd force-reload
```

Pour valider le bon fonctionnement de PHP sur ce serveur, créez le fichier **/var/www/test.php** avec la commande suivante :

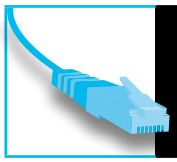
Terminal
```
~# echo '<? phpinfo() ?>' > /var/www/test.php
```



Accédez ensuite à l'adresse **http://<adresse_IP_du_serveur>/test.php** à partir d'un navigateur web ; la page de confirmation suivante s'affiche :

System	Linux glmf 3.2.0-4-686-pae #1 SMP Debian 3.2.41-2+deb7u2 i686
Build Date	Mar 4 2013 15:58:48
Server API	CGI/FastCGI
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php5/cgi
Loaded Configuration File	/etc/php5/cgi/php.ini
Scan this dir for additional .ini files	/etc/php5/cgi/conf.d
Additional .ini files parsed	/etc/php5/cgi/conf.d/10-pdo.ini
PHP API	20100412

Les plus observateurs auront remarqué que cette section est strictement identique à ce que l'on trouve dans le cas d'une architecture LAMP : en effet, dans les deux cas, PHP fonctionne de la même manière. Notons toutefois qu'on ne trouve pas les sections « mysqli », « PDO » et « pdo_mysql », vu que le paquet idoine n'a pas été installé.



3 Installer MySQL

MySQL peut très bien être utilisé avec un serveur Lighttpd : son utilisation se fait par PHP, il n'y a aucun lien direct avec le serveur web :

Terminal
```
~# apt-get install mysql-server php5-mysql
```

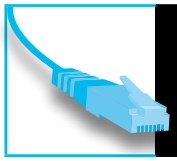
La procédure d'installation étant la même que celle utilisée lors de l'installation de MySQL avec Apache, elle demande également le mot de passe du super-utilisateur de MySQL. Pour des raisons de sécurité, ce mot de passe ne devrait pas être le même que celui de l'utilisateur **root**. Il servira à se connecter à MySQL en ligne de commandes, avec la commande **mysql**.

Après rechargement de Lighttpd (pour relancer l'interpréteur PHP lancé par FastCGI)...

Terminal
```
~# service lighttpd force-reload
```

... on peut valider le fonctionnement de ce module à l'adresse **http://<adresse_IP_du_serveur>/test.php**. On retrouve la confirmation que le module MySQL pour PHP est bien installé :

MySQL Support	enabled
Active Persistent Links	0
Active Links	0
Client API version	5.5.31
MYSQL_MODULE_TYPE	external



4 HTTPS

4.1 Créer un certificat SSL

Comme pour Apache, la création d'un certificat SSL auto-signé se fera par l'outil **make-ssl-cert**, disponible dans le paquet **ssl-cert** :

Terminal
```
~# apt-get install ssl-cert
```

Créons un certificat SSL dans **/etc/lighttpd/server.pem** :

Terminal
```
~# make-ssl-cert /usr/share/ssl-cert/ssleay.cnf /etc/lighttpd/server.pem
```

Comme vu précédemment, deux questions vous sont posées ici : tout d'abord le nom du serveur, on gardera ici l'exemple de **www.monsite.fr**. Ensuite, les autres sites à protéger avec ce certificat – nous ne voulons protéger qu'un seul site, alors on n'en déclare aucun.

4.2 Activer SSL sur Lighttpd

Comme pour PHP, le paquet Debian de Lighttpd propose une configuration toute faite pour activer SSL :

Terminal

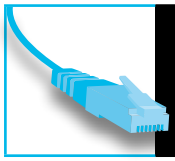
```
~# lighty-enable-mod ssl
Enabling ssl: ok
Run /etc/init.d/lighttpd force-reload to enable changes
~# service lighttpd force-reload
```

À partir de là, Lighttpd desservira le contenu de **/var/www** si on se connecte au serveur en HTTPS.

Et comme pour le protocole HTTP, il faut autoriser le flux HTTPS si le pare-feu est activé avec Ufw :

Terminal

```
~# ufw allow "WWW Secure"
```



5 Desservir plusieurs sites

Contrairement à Apache qui propose par défaut une configuration basée sur des hôtes virtuels afin de desservir plusieurs sites, Lighttpd ne sait par défaut desservir qu'un seul site.

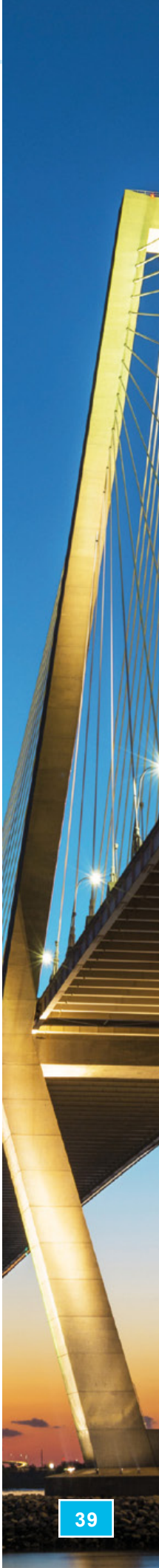
La méthode la plus simple pour desservir des sites consiste à placer tous les sites sous un même répertoire et à activer le module **simple-vhost** ; par défaut, le paquet de Lighttpd pour Debian propose une configuration où le répertoire en question est **/srv** ; celle-ci est activée avec la commande suivante :

Terminal

```
~# lighty-enable-mod simple-vhost
Enabling simple-vhost: ok
Run /etc/init.d/lighttpd force-reload to enable changes
~# service lighttpd force-reload
```

Comme indiqué dans le fichier **/etc/lighttpd/conf-available/10-simple-vhost.conf**, les fichiers des sites seront recherchés dans les répertoires **/srv/<nom d'hôte>/htdocs/**, le nom d'hôte par défaut (retourné lorsqu'une adresse inconnue est demandée, ou lorsque le serveur est accédé par son adresse IP directement) étant **www.example.com**. Chacun adaptera ce fichier à ses besoins, en modifiant les valeurs suivantes :

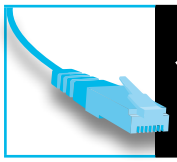
- **simple-vhost.server-root** : répertoire parent dans lequel chercher les données ;
- **simple-vhost.document-root** : sous-répertoire dans lequel est stockée l'arborescence du site ;
- **simple-vhost.default-host** : nom de l'hôte (répertoire) à desservir par défaut. ■



CHOISISSEZ LA HAUTE PERFORMANCE AVEC NGINX

Autre challenger intéressant, Nginx (prononcé – à l'anglaise – « Engine X ») est réputé pour ses performances qui crèvent le plafond, surtout avec des fichiers statiques. Développé à l'origine par une équipe russe, celui-ci est utilisé par de nombreux sites importants (plus de 15% des serveurs web), comme WordPress.com par exemple.

Nginx étant un serveur web à part entière, ni Apache, ni Lighttpd ne doivent être installés pour que celui-ci puisse fonctionner avec sa configuration par défaut : les deux logiciels accéderaient au port 80 en même temps, ce qui créerait un conflit. Soit il faut modifier la configuration de l'un d'entre eux pour écouter sur un autre port, soit il ne faut installer ni Apache, ni Lighttpd.



1 Installer Nginx

Pour offrir ces performances, Nginx a une particularité qu'on ne retrouve généralement plus dans les logiciels récents : il n'est pas modulaire. C'est à la compilation que l'on choisit les fonctionnalités à activer. C'est pourquoi Debian propose plusieurs paquets pour Nginx : **`nginx-light`**, **`nginx-full`**, **`nginx-extras`** et **`nginx-naxsi`**. À chacun de choisir la version qui lui convient, selon les modules activés. La liste complète des modules activés selon les paquets est disponible à l'adresse **http://wiki.debian.org/Nginx**.

Il est tout à fait possible de commencer avec l'un des paquets et d'en installer un plus complet si on se rend compte qu'on a besoin d'un module qui n'est pas présent dans le paquet actuellement installé. C'est pour cela que l'on va commencer par installer la version la plus légère :

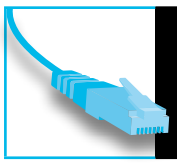
Terminal

```
~# apt-get install nginx-light
```

Après installation de ce paquet, Nginx ne démarre pas seul (nous laissant le configurer avant son premier démarrage). Démarrons-le toutefois immédiatement, afin de valider son fonctionnement :

Terminal

```
~# service nginx start
Starting nginx: nginx.
```

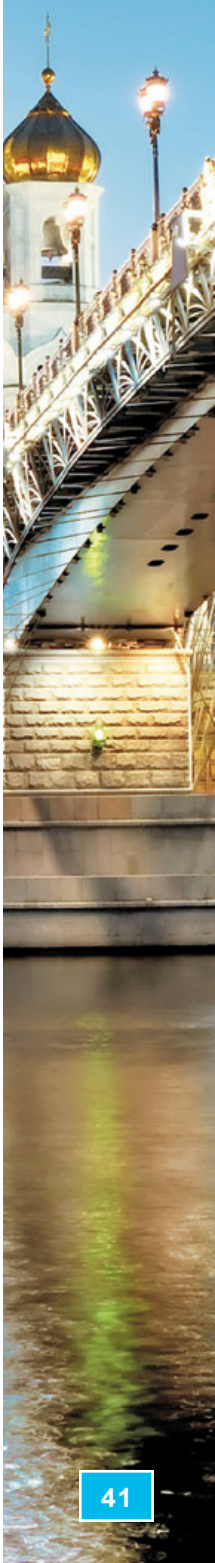


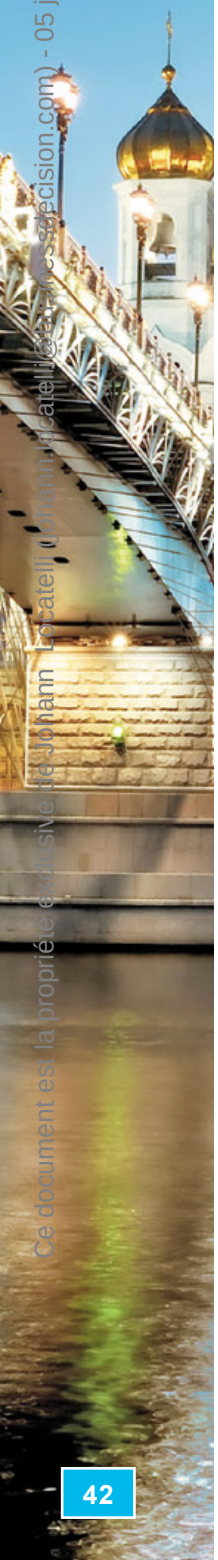
2 Règle de pare-feu

De même que pour Apache et Lighttpd, si on a choisi de filtrer les paquets entrants par Ufw, il faut alors autoriser les paquets sur le port HTTP, 80. Le paquet Nginx proposé par Debian ajoute des applications spécifiques à Ufw, utilisons celle qui est dédiée au protocole HTTP, même si le résultat sera strictement le même qu'en utilisant l'application « WWW » :

Terminal

```
~# ufw allow "Nginx HTTP"
Rule added
Rule added (v6)
```





3 Vérifier le fonctionnement

On peut vérifier le bon fonctionnement de Nginx en accédant à l'adresse **http://<adresse_IP_du_serveur>/** avec un navigateur web. Un simple message « Welcome to nginx! » est alors affiché : la configuration de Nginx par le paquet Debian n'est pas très verbeuse...

4 Desservir un site

Par défaut, Nginx dessert le contenu du répertoire `/usr/share/nginx/www`, par le biais de l'hôte virtuel par défaut configuré dans `/etc/nginx/sites-available/default`. Configurons un hôte virtuel plus adapté au besoin, dans le fichier `/etc/nginx/sites-available/monsite` :

Fichier

```
server {
   root /srv/monsite;
   index index.html;
   server_name www.monsite.fr;
   location / {
      try_files $uri $uri/ /index.html;
   }
}
```

Créons ensuite le répertoire destiné à contenir ce site, activons ce fichier et rechargeons la configuration de Nginx :

Terminal

```
~# mkdir /srv/monsite
~# cd /etc/nginx/sites-enabled/
/etc/nginx/sites-enabled# ln -s ../sites-available/monsite
/etc/nginx/sites-enabled# service nginx reload
Reloading nginx configuration: nginx.
```

Si on essaie maintenant d'accéder au site en question (à condition que le nom `www.monsite.fr` corresponde bien à ce serveur – à chacun de configurer cela selon ses besoins), on tombe sur une erreur « 403 Forbidden » : en effet, il n'y a pas de fichier `index.html` dans le répertoire `/srv/monsite` et nous n'avons pas configuré Nginx pour afficher le contenu du répertoire. Tout va bien.

Cette procédure peut être suivie autant de fois qu'il y a de sites à desservir.

5 PHP avec Nginx

Comme pour Lighttpd, il n'y a pas de fonctionnalité intégrée à Nginx pour traiter directement du PHP. On se basera alors encore une fois sur FastCGI. Nous utiliserons toutefois une méthode différente de ce qui a été fait avec Lighttpd : PHP propose un logiciel qui s'appelle

« Fast Process Manager » (FPM), dont le rôle est de traiter des requêtes FastCGI ; il se lance indépendamment du serveur web, ce qui rend sa gestion plus flexible.

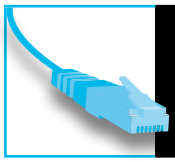
Le module FastCGI de Nginx est inclus dès la version « light », il n'y a donc pas besoin de passer à la version « full » ou « extras ».

Installons d'abord FPM :

Terminal
```
~# apt-get install php5-fpm
```

Le fichier **/etc/nginx/sites-available/monsite** est ensuite à modifier pour desservir des fichiers PHP :

Fichier
```
server {
   root /srv/monsite;
   index index.php index.html;
   server_name www.monsite.fr;
   location / {
      try_files $uri $uri/ /index.html;
   }
   location ~ \.php$ {
      fastcgi_pass unix:/var/run/php5-fpm.sock;
      fastcgi_index index.php;
      fastcgi_param SCRIPT_FILENAME /srv/monsite$fastcgi_script_name;
      include fastcgi_params;
   }
}
```



6 Page de test PHP

Pour valider le bon fonctionnement de PHP sur ce serveur, créez le fichier **/srv/monsite/test.php** comme suit :

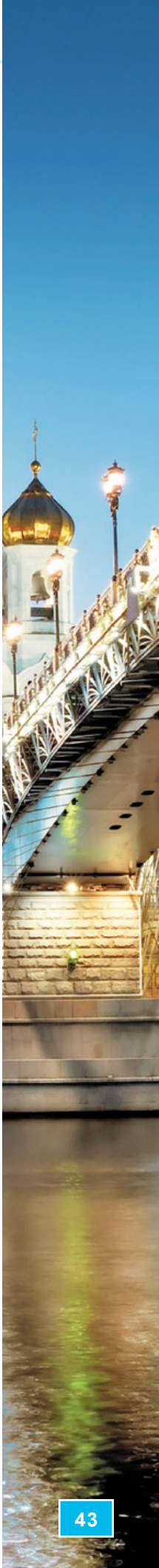
Terminal
```
~# echo '<? phpinfo() ?>' > /srv/monsite/test.php
```

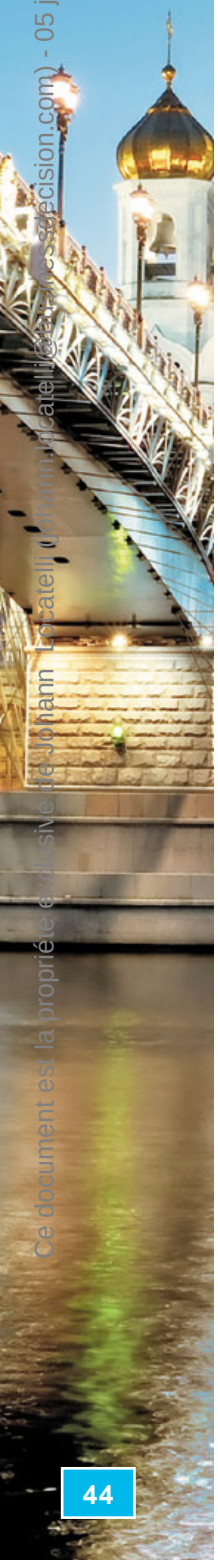
Accédez ensuite à l'adresse **http://<adresse_IP_du_serveur>/test.php** à partir d'un navigateur web ; la page de confirmation suivante s'affiche :

PHP Version 5.4.4-14

System	Linux gmlf 3.2.0-4-686-pae #1 SMP Debian 3.2.41-2+deb7u2 i686
Build Date	Mar 4 2013 15:55:09
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php5/apache2
Loaded Configuration File	/etc/php5/apache2/php.ini
Scan this dir for additional .ini files	/etc/php5/apache2/conf.d
Additional .ini files parsed	/etc/php5/apache2/conf.d/10-pdo.ini, /etc/php5/apache2/conf.d/20-mysql.ini, /etc/php5/apache2/conf.d/20-mysqli.ini, /etc/php5/apache2/conf.d/20-pdo_mysql.ini
PHP API	20100412
PHP Extension	20100525

Encore une fois, cette opération et son résultat sont identiques à ce que l'on trouve pour Lighttpd et pour Apache. Quel que soit le serveur HTTP utilisé, PHP fonctionne de la même manière.





7 Installer MySQL

MySQL peut être utilisé avec Nginx aussi bien qu'avec Apache ou Lighttpd : en réalité, ce n'est pas du tout le serveur HTTP qui dialogue avec la base de données, mais bien le programme en PHP. Dans Apache, c'est intégré grâce au module idoine ; pour Lighttpd, c'est le programme `php5-cgi` exécuté par le serveur web et dans le cas présent, c'est FPM qui s'en charge.

Terminal

```
~# apt-get install mysql-server php5-mysql
```

8 Mot de passe MySQL

Encore une fois, à procédure identique, déroulement identique. Le mot de passe du super-utilisateur de MySQL est demandé lors de son installation.

9 Valider l'activation de MySQL

FPM est automatiquement redémarré lors de l'installation de `php5-mysql`, on peut directement valider le fonctionnement de ce module à l'adresse **http://<adresse_IP_du_serveur>/test.php**. On retrouve encore une fois les sections confirmant que le module MySQL pour PHP est installé.

10 HTTPS : créer un certificat SSL

Créons un certificat SSL auto-signé, encore une fois avec `make-ssl-cert` qu'il faut d'abord installer :

Terminal

```
~# apt-get install ssl-cert
```

Le certificat sera créé dans le fichier `/etc/ssl/certs/monsite.crt` :

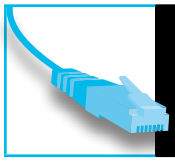
Terminal

```
~# make-ssl-cert /usr/share/ssl-cert/ssleay.cnf /etc/ssl/certs/monsite.crt
```

Les deux mêmes questions sont posées :

➥ Tout d'abord le nom du serveur (`www.monsite.fr` dans notre cas).

➥ Puis, les autres sites à protéger par ce certificat ; on n'en indique aucun, car on ne configure le certificat que pour un site :



11 HTTPS : activer SSL sur Nginx

Comme sur Apache, le chiffrement par SSL s'active de manière indépendante pour chaque hôte virtuel ; on peut également desservir plusieurs sites avec des certificats différents.

Pour activer SSL sur un site (on présente ici l'exemple de **www.monsite.fr** traité jusqu'ici), adaptez la configuration de la manière suivante :

Fichier

```
server {
   listen 443;
   root /srv/monsite;
   index index.html;
   server_name www.monsite.fr;
   location / {
      try_files $uri $uri/ /index.html;
   }
   ssl on;
   ssl_certificate /etc/ssl/certs/monsite.crt;
   ssl_certificate_key /etc/ssl/certs/monsite.crt;
}
```

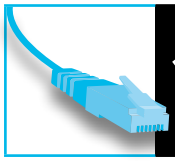
Il est bien sûr possible de combiner PHP et SSL sur un même hôte virtuel.

Il suffit ensuite de recharger la configuration de Nginx :

Terminal

```
~# service nginx reload
Reloading nginx configuration: nginx.
```

À partir de là, Nginx desservira le contenu de **/serv/monsite** si on se connecte au serveur en HTTPS.

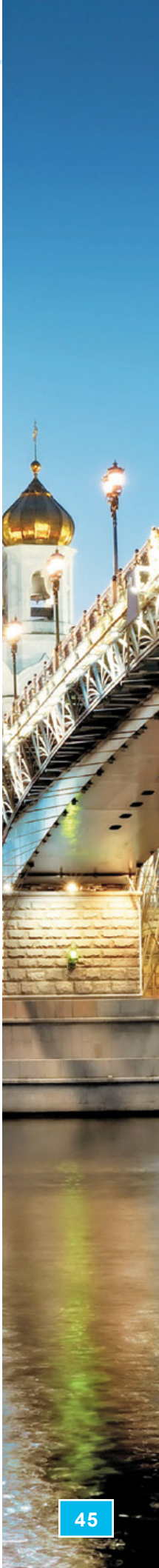


12 Autoriser le flux HTTPS

Avec Ufw, c'est toujours la même punition : si les ports sont filtrés, alors il faut autoriser le protocole HTTPS ; par souci de cohérence, on utilise encore une fois la règle applicative spécifique à Nginx, même si la règle classique **WWW Secure** offre le même résultat ;

Terminal

```
~# ufw allow "Nginx HTTPS"
Rule added
Rule added (v6)
```



3

3 MAIL

À découvrir dans cette partie...

GÉRER SA MESSAGERIE AVEC EXIM

Exim est le MTA (Mail Transfer Agent) par défaut de Debian. Ce logiciel est une valeur sûre, utilisée sur de nombreuses infrastructures de par le monde, que ce soit pour des besoins modestes ou pour des milliers de boîtes e-mail. Nous verrons ici comment le mettre en œuvre dans le cadre d'un serveur de messagerie complet.

La première version d'Exim date de 1995 ; elle fut écrite par Philip Hazel pour le service informatique de l'université de Cambridge.

Exim est un MTA monolithique, il est constitué d'un seul processus binaire qui se charge de l'ensemble des traitements des e-mails. Certains pensent qu'Exim se comporte mal dans des environnements à forte charge, cependant il est déployé sur des plateformes où il doit traiter plusieurs milliers de messages à l'heure sans le moindre problème.

1 Installer Exim

Le projet Debian propose deux « versions » d'Exim, avec plus ou moins de fonctionnalités compilées :

- **`exim4-daemon-light`** inclut uniquement les fonctionnalités de base ;
- **`exim4-daemon-heavy`** inclut des fonctionnalités étendues : la recherche de données LDAP, SQLite, PostgreSQL et MySQL, l'authentification SASL et SPA, un interpréteur Perl embarqué, l'extension de balayage de contenu.

Il est possible, après installation, de passer d'une version à l'autre de manière quasi-transparente. Commençons alors avec le paquet le plus simple...

Terminal
```
~# apt-get install exim4-daemon-light
```

2 Configuration de base

Le paramétrage mis en place par défaut est très limité. Le paquet Debian propose une méthode permettant de mettre en œuvre une configuration plus « réaliste ». Pour utiliser cette méthode, exécutez la commande suivante :

Terminal
```
~# dpkg-reconfigure exim4-config
```

Cette commande affiche alors un message indiquant ce qu'il sera possible de sélectionner comme configuration ; validez avec « Ok » (pour se placer sur ce « bouton », il faut utiliser la touche [Tab]).

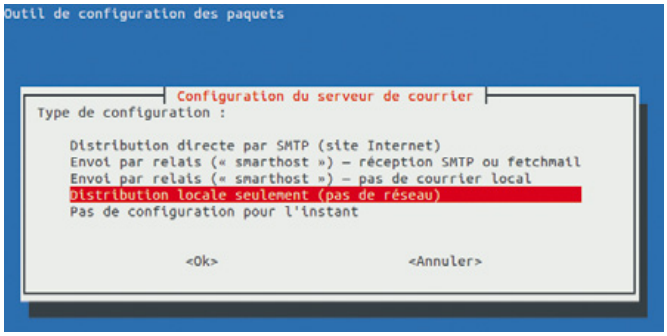
Définition

Dans une infrastructure de messagerie, les logiciels se classent en 5 catégories :

- **les MTA (*Mail Transfer Agent*) ont pour rôle de transporter les e-mails d'un serveur A à un serveur B, généralement par le protocole SMTP ;**
- **les MDA (*Mail Delivery Agent*) se chargent de stocker les e-mails sur le serveur de destination en les filtrant si nécessaire ;**
- **les MSA (*Mail Submission Agent*) reçoivent les nouveaux e-mails et les envoient aux MTA – on notera que ce sont souvent les mêmes logiciels qui ont le rôle de MSA et le rôle de MTA ;**
- **les MRA (*Mail Retrieval Agent*) récupèrent les e-mails dans les boîtes des utilisateurs pour qu'ils puissent être lus ;**
- **les MUA (*Mail User Agent*) sont les clients de messagerie, qui envoient des e-mails par l'intermédiaire des MSA et qui interrogent les MRA pour recevoir les nouveaux e-mails arrivés.**

3 Type de configuration

La procédure de configuration demande tout d'abord quel est le type de serveur à mettre en place.

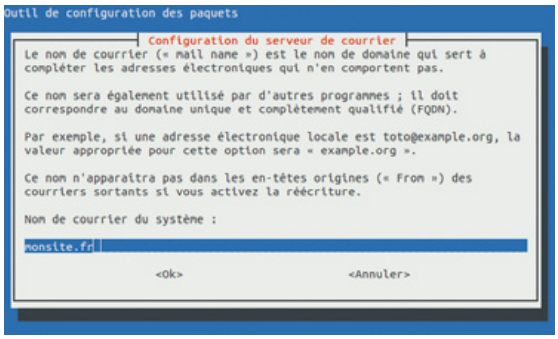


Deux options nous intéressent :

- ***Distribution directe par SMTP (site Internet)*** : cette option permet de mettre en place un serveur SMTP autonome, qui sera capable d'envoyer ses e-mails directement à ses destinataires – elle est à choisir si votre hébergeur ne propose pas de *relais SMTP* ;
- ***Envoi par relais (« smarthost ») — réception SMTP ou fetchmail*** : cette option permet de mettre en place un serveur SMTP qui transmettra ses e-mails sortants à un relais – c'est le cas si l'on s'auto-héberge (on utilise alors le serveur SMTP de son fournisseur d'accès), ou si l'hébergeur propose un service de relais SMTP.

4 Nom de courrier

L'étape suivante est la définition du nom de domaine par défaut du serveur : lorsqu'un e-mail sortant ne précise pas l'adresse e-mail de l'expéditeur, alors ce domaine est ajouté au nom d'utilisateur.

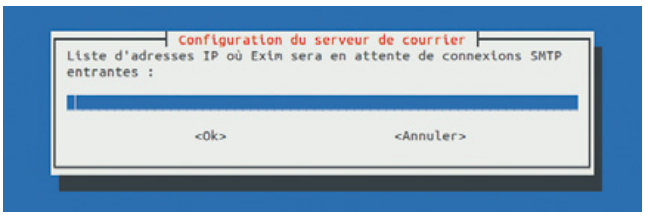


Pour notre exemple, nous indiquons `monsite.fr` (le nom `root` sera donc changé en `root@monsite.fr` dans les e-mails sortants) : chacun y indiquera son propre domaine.

5 Adresses d'écoute

Ensuite, l'assistant demande les adresses IP sur lesquelles le serveur doit être à l'écoute.

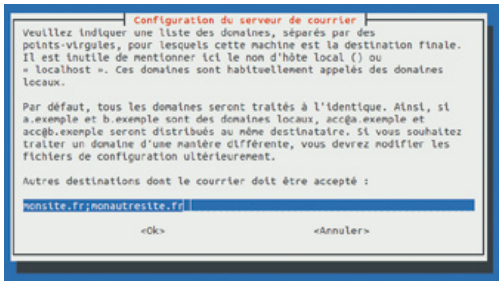
Pour que le service soit desservi sur toutes les interfaces, on ne renseigne rien dans ce champ (ce paramètre est utile si le serveur est connecté sur plusieurs réseaux par plusieurs cartes réseau et ne doit desservir le protocole SMTP que sur certains d'entre eux).



6 Destinations à accepter

L'assistant demande alors la liste des destinations pour lesquelles les e-mails doivent être acceptés ; les différents domaines doivent être séparés par des points-virgules.

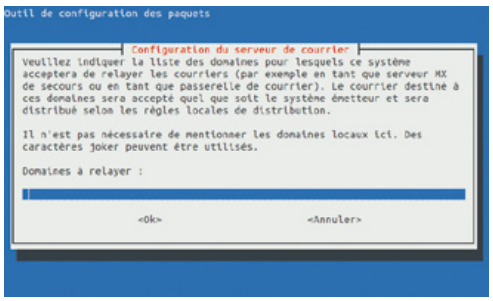
Dans notre exemple, nous indiquons **`monsite.fr;monautresite.fr`** pour accepter tous les e-mails à destination de ces deux domaines.



7 Domaines à relayer

Exim peut également être configuré pour accepter des e-mails à relayer pour d'autres serveurs : ces e-mails seront acceptés, puis retransmis au bon destinataire. C'est utile pour définir un serveur secondaire, qui servira de point de passage de secours en cas de panne du serveur e-mail principal.

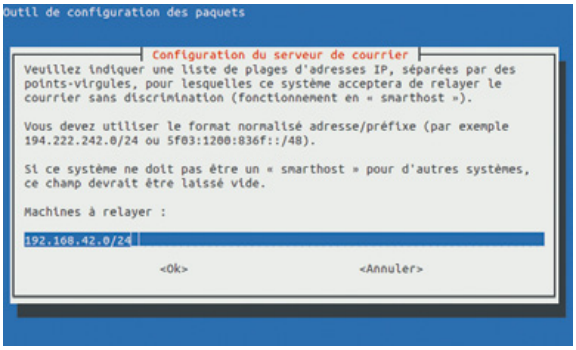
Notre exemple ne prend en compte aucun domaine à relayer, ce champ reste donc vide.



8 Être un smarthost

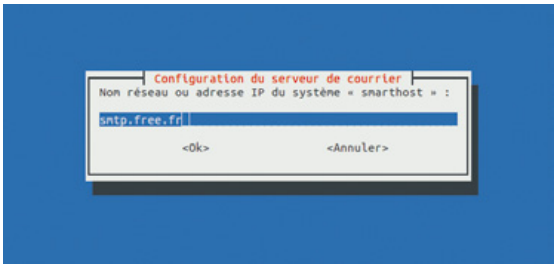
Si ce serveur doit relayer des e-mails pour d'autres serveurs ou pour des postes de travail, il faut préciser dans ce champ le ou les réseau(x) à desservir.

Dans le cas d'un serveur dédié et sans autre serveur à relayer, ce champ doit rester vide. Dans le cas d'un serveur auto-hébergé, on peut imaginer qu'il servira de point de passage pour les e-mails envoyés par les PC présents dans le réseau, on peut alors définir la plage réseau ; dans notre exemple, ce sera **`192.168.42.0/24`**.



Avec une telle configuration, Exim n'est alors pas seulement un MTA, il prend aussi le rôle de MSA (*Mail Submission Agent*).

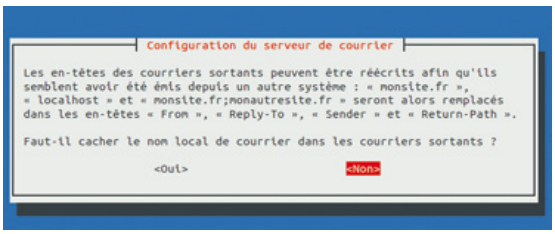
9 Relayer vers un smarthost



Dans le cas de l'envoi par relais, il faut indiquer à l'assistant l'adresse vers laquelle relayer tous les e-mails sortants. **Cette étape ne s'affiche pas si on a choisi la distribution directe**.

Dans notre exemple, nous imaginons un serveur auto-hébergé chez le fournisseur d'accès Free, nous indiquons donc `smtp.free.fr` comme relais.

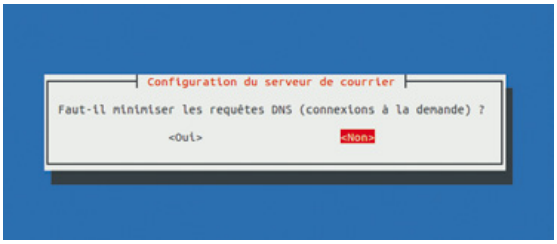
10 Cacher le nom local



Dans le cas de l'envoi par relais, il est également possible de réécrire tous les e-mails sortants, afin qu'ils semblent avoir été envoyés par d'autres adresses.

Cette fonctionnalité ne correspondant pas à notre besoin, on répond « Non ».

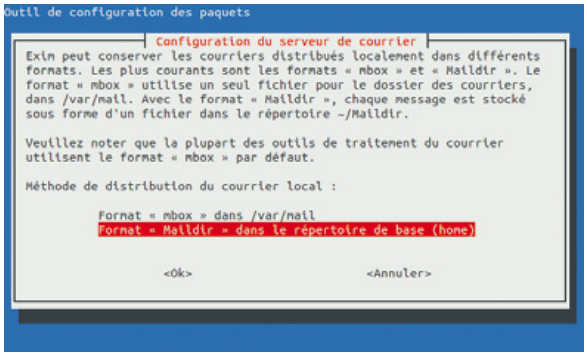
11 Minimisation des requêtes DNS



Exim effectue habituellement des requêtes DNS pendant son fonctionnement, dans différents buts. Si le serveur est placé derrière une connexion discontinue, alors cela peut poser un problème. Dans le cas d'un serveur dédié ou auto-hébergé, qui est connecté à Internet en continu, cela ne pose pas de problème. On demande alors de ne pas minimiser les requêtes DNS.

12 Méthode de distribution

Lorsqu'un courrier arrive sur le serveur, Exim le dépose dans le système de fichiers. Deux méthodes sont proposées :

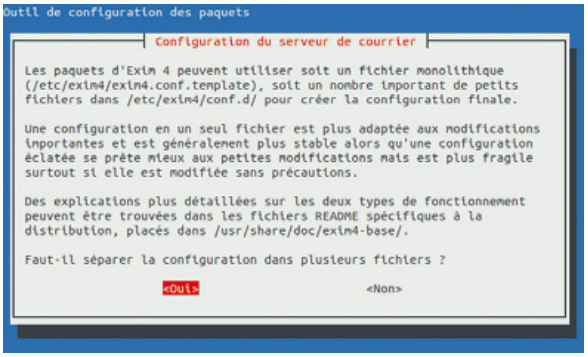


- **mbox** : un seul fichier par utilisateur stocke tous ses e-mails, dans **/var/mail** ;
- **Maildir** : un fichier par e-mail, stocké dans le répertoire **Maildir** du répertoire personnel de chaque utilisateur.

Nous choisissons le format « Maildir », car celui-ci est plus efficace dans certains cas ; notamment lorsqu'un utilisateur a beaucoup d'e-mails dans son arborescence, la gestion de l'ensemble en un seul gros fichier peut poser des problèmes de performance. L'impact risque d'être très important si l'on choisit ensuite d'utiliser le protocole IMAP pour lire les e-mails, en les laissant sur le serveur.

On voit ici qu'Exim prend également le rôle de MDA (*Mail Distribution Agent*).

13 Séparer la configuration



Enfin, l'assistant demande si la configuration doit être compilée dans un seul fichier, ou si elle doit être éclatée en plusieurs fichiers.

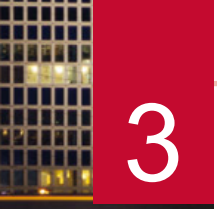
On choisit la séparation en plusieurs fichiers, car cela permet une gestion plus souple : un seul fichier « monolithique » est compliqué à gérer sur le long terme.

14 Règle de pare-feu

Avec le pare-feu activé, il faut ajouter une règle pour autoriser le protocole SMTP :

Terminal

```
~# ufw allow SMTP
Rule added
Rule added (v6)
```



15 Création d'un utilisateur

Exim se base sur les utilisateurs classiques du système ; lorsque l'on veut créer un nouvel utilisateur de messagerie, il faut alors créer un nouvel utilisateur sur le serveur :

Terminal

```
~# adduser toto
Ajout de l'utilisateur " toto " ...
Ajout du nouveau groupe " toto " (1001) ...
Ajout du nouvel utilisateur " toto " (1001) avec le groupe
" toto " ...
Création du répertoire personnel " /home/toto "...
Copie des fichiers depuis " /etc/skel "...
Entrez le nouveau mot de passe UNIX :
Retapez le nouveau mot de passe UNIX :
passwd : le mot de passe a été mis à jour avec succès
Modification des informations relatives à l'utilisateur toto
Entrez la nouvelle valeur ou " Entrée " pour conserver la valeur
proposée
       Nom complet []:
       N° de bureau []:
       Téléphone professionnel []:
       Téléphone personnel []:
       Autre []:
Cette information est-elle correcte ? [O/n]o
```

16 Test des e-mails entrants

Une fois un utilisateur local créé, il est possible de valider le bon fonctionnement d'Exim en dialoguant directement avec le protocole SMTP. On utilisera le logiciel **telnet** pour faire cette manipulation manuellement, il faut donc l'installer :

Terminal

```
~# apt-get install telnet
```

Une communication SMTP simplifiée se déroule de la manière suivante :

- Le serveur nous accueille avec le code **220** et une chaîne textuelle ;
- On envoie la commande **HELO nom-d-hote** pour le saluer ;
- Le serveur nous répond avec le code **250** et une chaîne textuelle ;
- On indique l'adresse e-mail de l'expéditeur avec la commande **MAIL FROM: <adresse>** ;

- Le serveur accepte cette adresse avec le code **250** ;
- On indique l'adresse e-mail du destinataire avec la commande **RCPT TO: <adresse>** ;
- Le serveur accepte, encore une fois avec le code **250** ;
- On indique que l'on va envoyer le contenu du mail avec la commande **DATA** ;
- Le serveur indique qu'il est prêt avec le code **354** ;
- On entre les entêtes du mail, puis une ligne vide, puis le contenu du mail, puis un point seul sur une ligne ;
- Le serveur indique que le mail est accepté avec le code **250** ;
- On se déconnecte avec la commande **QUIT** ;
- Le serveur confirme que la communication est finie avec le code **221**.

Un exemple vaut mieux que des pages d'explications, le voici :

Terminal

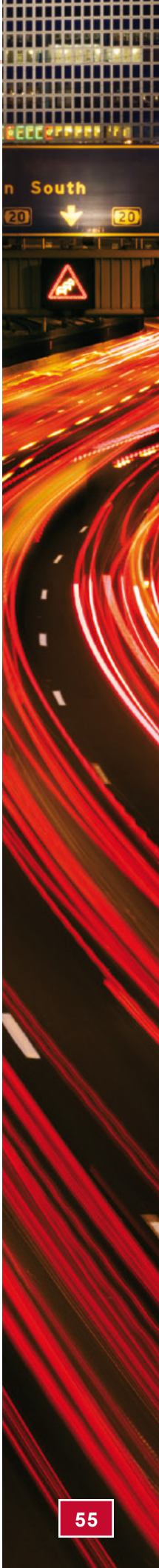
```
~# telnet localhost 25
Trying ::1...
Connected to localhost.
Escape character is '^]'.
220 glmf ESMTP Exim 4.80 Thu, 30 May 2013 12:30:37 +0200
HELO test.moi
250 glmf Hello localhost [::1]
MAIL FROM: <test@test.moi>
250 OK
RCPT TO: <toto@monsite.fr>
250 Accepted
DATA
354 Enter message, ending with "." on a line by itself
Subject: Essai

Contenu de test !
.
250 OK id=1Ui08V-0001gl-M3
QUIT
221 glmf closing connection
Connection closed by foreign host.
```

Ici, le serveur d'exemple s'appelle « glmf », c'est pourquoi ses réponses sont toujours préfixées par ces quatre lettres.

On peut alors constater que cet e-mail est bien arrivé, on le retrouve dans le répertoire **Maildir/new** de l'utilisateur concerné :

Terminal

```
~# ls /home/toto/Maildir/new/
1369309879.H356696P6498.glmf
```

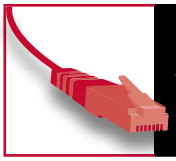


On peut même lire son contenu :

Terminal

```
~# cat /home/toto/Maildir/new/1369909879.H356696P6498.glmf
Return-path: <test@test.moi>
Envelope-to: toto@monsite.fr
Delivery-date: Thu, 30 May 2013 12:31:19 +0200
Received: from localhost ([::1] helo=test.moi)
      by glmf with smtp (Exim 4.80)
      (envelope-from <test@test.moi>)
      id 1Ui08V-0001gl-M3
      for toto@monsite.fr; Thu, 30 May 2013 12:31:19 +0200
Subject: Essai
Message-Id: <E1Ui08V-0001gl-M3@glmf>
From: test@test.moi
Date: Thu, 30 May 2013 12:31:14 +0200

Contenu de test !
```

Une fois un e-mail stocké sur le serveur, ce sera le rôle du MDA de le délivrer à l'utilisateur, par l'intermédiaire de son client de messagerie (MUA).



17 Test des e-mails sortants

On peut ensuite valider le bon fonctionnement de l'envoi des e-mails vers l'extérieur, de la même manière mais en pointant vers une adresse qui n'est pas gérée par le serveur. Dans notre exemple, nous avons activé le relais pour le réseau **192.168.42.0/24**, on va donc tenter cette opération à partir d'un PC de ce réseau :

Terminal

```
~$ telnet 192.168.42.37 25
Trying 192.168.42.37...
Connected to 192.168.42.37.
Escape character is '^]'.
220 glmf ESMTP Exim 4.80 Thu, 30 May 2013 12:46:32 +0200
HELO test.moi
250 glmf Hello test.moi [192.168.42.43]
MAIL FROM: <n-existe-pas@gmail.com>
250 OK
RCPT TO: <destinataire@gmail.com>
250 Accepted
DATA
354 Enter message, ending with "." on a line by itself
Subject: Essai 2

Second contenu de test...
.
250 OK id=1Ui0Nq-0001hg-2M
QUIT
221 glmf closing connection
Connection closed by foreign host.
```

Le destinataire reçoit alors cet e-mail.

18 Vérification dans les journaux

Les journaux système d'Exim sont placés dans le répertoire **/var/log/exim4**, plus particulièrement dans le fichier **mainlog**. On peut retrouver nos deux e-mails dans ce fichier, on a alors tous les détails de la façon dont ils ont été traités :

Fichier

```
2013-05-30 12:31:19 1Ui08V-0001gl-M3 <= test@test.moi H=localhost
(test.moi) [::1] P=smtp S=315
2013-05-30 12:31:19 1Ui08V-0001gl-M3 => toto <toto@monsite.fr>
R=local_user T=maildir_home
2013-05-30 12:31:19 1Ui08V-0001gl-M3 Completed
2013-05-30 12:47:11 1Ui0Nq-0001hg-2M <= n-existe-pas@gmail.com
H=(test.moi) [192.168.42.43] P=smtp S=350
2013-05-30 12:47:14 1Ui0Nq-0001hg-2M => destinataire@gmail.com
R=smarthost T=remote_smtp_smarthost H=smtp.free.fr [212.27.48.4]
2013-05-30 12:47:14 1Ui0Nq-0001hg-2M Completed
```

On remarque notamment que chaque ligne de ces logs indique l'identifiant unique du mail, qui a également été donné par Exim lorsqu'il a été envoyé (juste avant que l'on envoie la commande **QUIT**). Cela permet de tracer le parcours d'un e-mail s'il y a des problèmes d'envoi ou de réception.

19 Alias

Généralement, on souhaite fournir à nos correspondants une adresse e-mail qui ne se résume pas à un simple nom d'utilisateur ; l'usage veut qu'on se tourne vers quelque chose comme **prenom.nom@domaine.fr**.

Les MTA permettent de traduire une adresse en une autre, c'est ce qu'on appelle des **alias**. Ceux-ci se configurent dans le fichier **/etc/aliases**, la syntaxe étant la suivante :

Fichier

```
jean.bon: toto
```

Une fois ce fichier modifié, il est nécessaire d'utiliser la commande suivante pour prendre les modifications en compte :

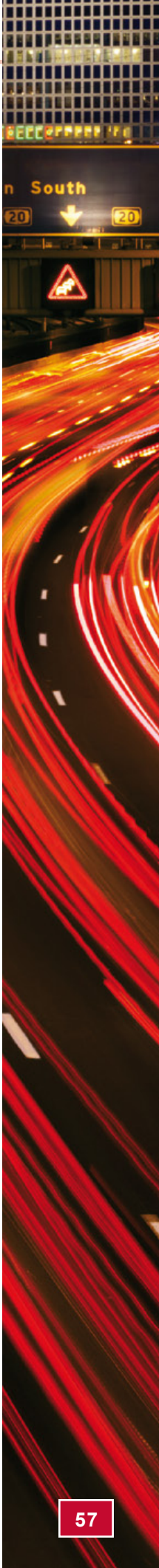
Terminal

```
~# newaliases
```

Dans notre exemple, un e-mail envoyé à **jean.bon@monsite.fr** arrivera dans le compte **toto** :

Fichier

```
2013-05-30 12:55:20 1Ui0Vm-0001hp-9g <= test@test.moi H=(test.
moi) [192.168.42.43] P=smtp S=299
2013-05-30 12:55:20 1Ui0Vm-0001hp-9g => toto <jean.bon@monsite.
fr> R=local_user T=maildir_home
2013-05-30 12:55:20 1Ui0Vm-0001hp-9g Completed
```



POSTFIX, UN SERVEUR DE MESSAGERIE FACILE À ADMINISTRER

Postfix est parmi les MTA libres les plus réputés. Il est utilisé par près d'un quart des serveurs de messagerie sur Internet. Il est notamment le serveur de messagerie par défaut de la distribution Red Hat Enterprise Linux ; il est également utilisé par le logiciel de messagerie collaborative Zimbra.

Postfix a été publié pour la première fois en 1998, écrit par Wietse Venema au centre de recherche IBM T. J. Watson. Il a d'abord été connu sous les noms *Vmailer* et *IBM Secure Mailer*. Il repose sur une architecture très modulaire, chaque sous-service étant géré par un processus différent ; cela lui permet d'être très performant sur des systèmes multi-tâches, notamment lorsqu'un nombre important de processeurs ou de cœurs est disponible.

1 Installer Postfix

Postfix s'installe tout simplement par le paquet **postfix** :

Terminal

```
~# apt-get install postfix
```

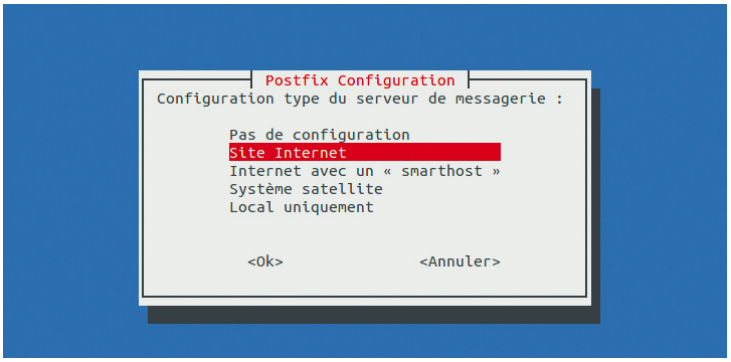
2 Type de serveur

Lors de l'installation du paquet **postfix**, l'assistant de configuration s'affiche automatiquement. Il demande tout d'abord quelle est la configuration type la plus adaptée au besoin. Parmi les choix proposés, les deux entrées qui nous intéressent sont :

- ***Site Internet*** : serveur SMTP autonome, qui enverra directement ses e-mails à ses destinataires et qui est capable d'en recevoir ;
- ***Internet avec un « smarthost »*** : serveur SMTP qui retransmet ses e-mails sortants à un relais.

Les autres possibilités sont :

- ***Système satellite*** : serveur SMTP qui ne fait qu'envoyer des e-mails (pas configuré pour en recevoir), par l'intermédiaire d'un relais ;
- ***Local uniquement*** : serveur SMTP qui ne communique avec aucun autre serveur.



À savoir

De nombreux MTA libres existent, nous ne pouvons bien sûr pas les traiter tous, ou alors il faudrait une centaine de pages rien que sur ce sujet. Évoquons toutefois les autres possibilités...

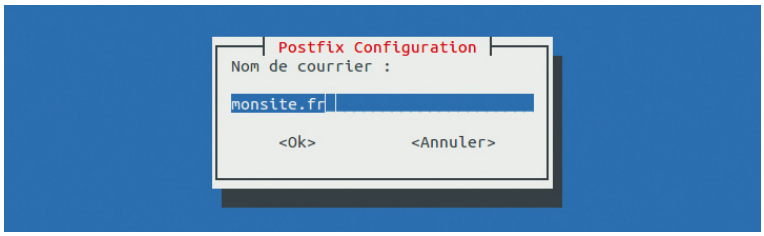
Sendmail est le MTA historique, créé au début des années 80 ; son plus grand inconvénient est le format des fichiers de configuration, plutôt indigeste.

On peut également rencontrer qmail, mais le développement de celui-ci était très ralenti, car sa licence interdisait de le distribuer sous forme compilée ; sa dernière version (1.03), sortie en 1998, est passée dans le domaine public. Malgré la faible activité de développement, ce logiciel a toujours du succès auprès de certains administrateurs.

On peut également trouver des mini-MTA, qui sont généralement utilisés comme « smarthosts », par exemple dans des réseaux de taille très réduite : nullmailer, msmtp...

3 Nom de courrier

L'assistant demande ensuite le nom sous lequel les e-mails sortants doivent être connus. Ce nom sera ajouté comme nom de domaine pour tous les e-mails sortants dont l'expéditeur n'est qu'un simple utilisateur, sans adresse e-mail complète.

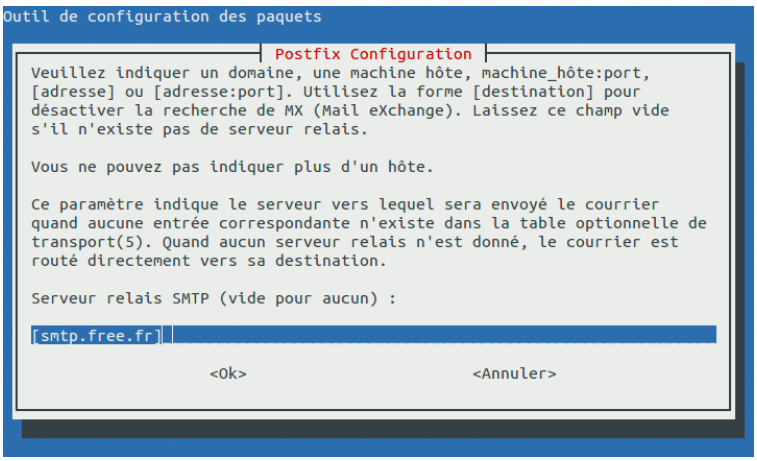


On garde toujours le même exemple, **monsite.fr** : un e-mail émanant de **root** sera vu comme envoyé par **root@monsite.fr**.

4 Relayer vers un smarthost

Dans le cas où on a choisi ***Internet avec un « smarthost »***, l'assistant demande l'adresse vers laquelle envoyer les e-mails. Celle-ci peut prendre différentes formes :

- **adr.ess.e.ip** : envoi vers l'adresse IP spécifiée ;
- **nom_de_domaine** : résolution de l'entrée MX du nom de domaine, envoi vers le serveur identifié pour ce nom ;
- **[nom_de_machine]** : envoi à la machine qui correspond au nom indiqué.



Nous prenons ici l'exemple où le serveur est auto-hébergé sur une connexion chez le fournisseur d'accès Free, le serveur relais étant donc **smtp.free.fr**.

5 Autres questions...

En réalité, cet assistant est capable de poser d'autres questions. Celles-ci sont les principales, elles sont posées lors de l'installation. Ensuite, la configuration se fait dans les fichiers que nous allons aborder ci-après. Il est toutefois possible de forcer l'assistant à nous poser toutes les questions en procédant comme avec Exim :

Terminal

```
~# dpkg-reconfigure postfix
```

Nous n'utiliserons pas cette commande et allons, cette fois-ci, paramétrer Postfix par ses fichiers de configuration.

6 (Dés)activation des sous-services

Les sous-services activés ou désactivés sont configurés dans le fichier **/etc/postfix/master.cf**. Celui-ci indique chaque nom de service et ses options sur une ligne, terminant par le nom de la commande exécutée pour ce service.

Pour désactiver un service, il suffit de commenter la ligne qui y correspond (en la faisant commencer par un dièse **#**) ; et inversement pour activer un service.

On retrouve par exemple la ligne suivante, qui permet à Postfix d'« écouter » sur le port SMTP :

Fichier

```
smtp      inet  n       -       -       -       -       smtpd
```

Si on commente cette ligne, Postfix n'« écoutera » plus sur le port SMTP et ne pourra plus recevoir de courrier à partir d'autres serveurs ; il pourra toutefois envoyer des e-mails vers d'autres serveurs, le protocole SMTP n'étant pas le seul moyen de faire « rentrer » un e-mail dans Postfix.

Ce chapitre n'est là qu'à titre de référence, on ne modifiera pas ce fichier : sa configuration par défaut est convenable pour les cas les plus courants.

7 Fichier de configuration

Le fichier de configuration principal de Postfix est **/etc/postfix/main.cf**. C'est généralement ce fichier que l'on modifie lorsqu'il y a des paramètres à changer.

Après avoir changé un élément de ce fichier, il est nécessaire de recharger la configuration avec la commande suivante :

Terminal
```
~# service postfix reload
```

8 Réception en Maildir

Postfix peut également avoir le rôle de MDA (*Mail Delivery Agent*) : il est capable de stocker les e-mails reçus dans différents formats, dont les populaires **mbox** et **Maildir**. Étant donné que, de nos jours, les accès aux e-mails ne devraient se faire qu'avec le protocole IMAP, c'est le format **Maildir** qui est le plus adapté.

Pour stocker les e-mails entrants dans un répertoire **Maildir** dans le dossier personnel de chaque utilisateur, la ligne suivante doit être ajoutée au fichier **/etc/postfix/main.cf** :

Fichier
```
home_mailbox = Maildir/
```

La configuration est ensuite rechargée :

Terminal
```
~# service postfix reload
```

9 Destinations du serveur

Pour demander à Postfix d'accepter localement d'autres destinations que celle qui a été définie avec l'assistant, il faut modifier la ligne **mydestination** du fichier **main.cf** ; par exemple, pour accepter les e-mails à destination de **monsite.fr** et de **monautresite.fr**, on ajoutera la ligne suivante :

Fichier
```
mydestination = monsite.fr, monautresite.fr
```

Après cette modification, comme après toute modification, la configuration de Postfix doit être rechargée.

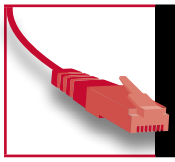
10 Être un smarthost

Postfix peut bien sûr être configuré pour relayer les e-mails provenant d'un ensemble d'autres machines. Ce paramètre correspond à la directive **mynetworks** du fichier **main.cf**. Par défaut, il est relais de lui-même ; ajoutons par exemple le réseau **192.168.42.0/24** :

Fichier

```
mynetworks = 192.168.42.0/24 127.0.0.0/8 [::ffff:127.0.0.0]/104 [::1]/128
```

Avec ce paramétrage, Postfix servira de MSA (*Mail Submission Agent*) pour l'ensemble des machines du réseau **192.168.42.0/24**.

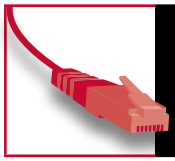


11 Règle de pare-feu

Encore une fois, si le pare-feu est activé, il faut ouvrir le port SMTP. Comme Nginx pour les serveurs HTTP, le paquet Postfix distribué par Debian ajoute une application à Ufw ; celle-ci est parfaitement équivalente à SMTP que l'on a utilisé avec Exim (ouverture du port TCP 25), mais elle permet d'afficher explicitement que l'on parle de Postfix dans le retour de la commande **ufw status** :

Terminal

```
~# ufw allow Postfix
Rule added
Rule added (v6)
```

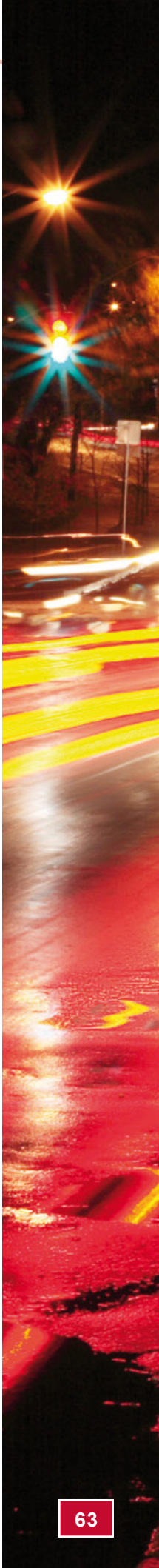


12 Utilisateur

Par défaut, Postfix délivre les e-mails pour les utilisateurs existant sur le serveur. Créons donc un nouvel utilisateur :

Terminal

```
~# adduser pdupont
Ajout de l'utilisateur " pdupont " ...
Ajout du nouveau groupe " pdupont " (1001) ...
Ajout du nouvel utilisateur " pdupont " (1001) avec le groupe
" pdupont " ...
Création du répertoire personnel " /home/pdupont "...
Copie des fichiers depuis " /etc/skel "...
Entrez le nouveau mot de passe UNIX :
Retapez le nouveau mot de passe UNIX :
passwd : le mot de passe a été mis à jour avec succès
Modification des informations relatives à l'utilisateur pdupont
Entrez la nouvelle valeur ou " Entrée " pour conserver la valeur
proposée
       Nom complet []:
       N° de bureau []:
       Téléphone professionnel []:
       Téléphone personnel []:
       Autre []:
Cette information est-elle correcte ? [O/n]o
```



À savoir

POURQUOI TELNET ?

Telnet est principalement connu comme logiciel de connexion distante, permettant de prendre la main sur un serveur, ceci de manière absolument pas sécurisée : il ne propose aucun chiffrement ! En réalité, Telnet est simplement un client qui permet d'envoyer et de recevoir des informations par le protocole TCP : il ne propose aucune gestion spécifique, aucune négociation automatique, etc. Ce n'est ni plus ni moins qu'une interface d'entrée/sortie au travers du protocole TCP, particulièrement simple et « bête ».

C'est pourquoi il est utilisé pour tester manuellement certains protocoles, dont le protocole SMTP : il permet de voir précisément comment le serveur réagit à telle ou telle commande, sans qu'aucune négociation ne se fasse automatiquement… On se connecte alors simplement avec la commande `telnet` au port TCP à tester, en l'occurrence le port 25.

13 Test des e-mails entrants

Validons ensuite que Postfix accepte bien les e-mails à destination de cet utilisateur, en nous connectant manuellement avec **`telnet`** sur le port en écoute de Postfix. Commençons par installer **`telnet`** :

Terminal

```
~# apt-get install telnet
```

Ensuite, avec **`telnet`**, simulons une conversation SMTP classique :

Terminal

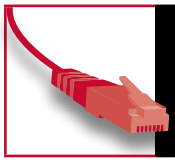
```
~# telnet localhost 25
Trying ::1...
Connected to localhost.
Escape character is '^]'.
220 glmf ESMTP Postfix (Debian/GNU)
HELO test.moi
250 glmf
MAIL FROM: <test@test.moi>
250 2.1.0 Ok
RCPT TO: <pdupont@monsite.fr>
250 2.1.5 Ok
DATA
354 End data with <CR><LF>.<CR><LF>
Subject: Essai Postfix

Contenu de test pour Postfix...
.
250 2.0.0 Ok: queued as 400C961449
QUIT
221 2.0.0 Bye
Connection closed by foreign host.
```

On retrouve cet e-mail dans le répertoire **`Maildir`** de l'utilisateur **`pdupont`** :

Terminal

```
~# ls /home/pdupont/Maildir/new/
1369985712.V801I6c5M458268.glmf
```

La distribution de cet e-mail à l'utilisateur sera ensuite le rôle du MDA (*Mail Distribution Agent*) – ce sujet sera traité plus tard.

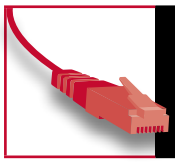


14 Test d'utilisateur inexistant

Au passage, testons comment réagit Postfix si on lui transmet un e-mail pour un utilisateur inexistant :

Terminal

```
~# telnet localhost 25
Trying ::1...
Connected to localhost.
Escape character is '^]'.
220 glmf ESMTP Postfix (Debian/GNU)
HELO test.moi
250 glmf
MAIL FROM: <test@test.moi>
250 2.1.0 Ok
RCPT TO: <inconnu@monsite.fr>
550 5.1.1 <inconnu@monsite.fr>: Recipient address rejected: User
unknown in local recipient table
QUIT
221 2.0.0 Bye
Connection closed by foreign host.
```



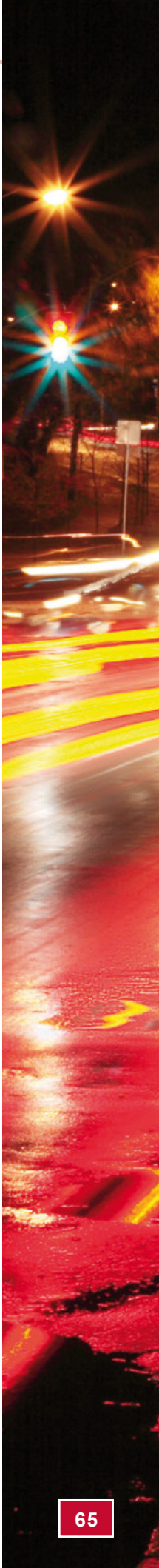
15 Test des e-mails sortants

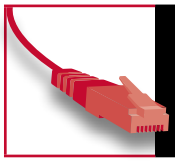
Les e-mails entrants fonctionnent bien, testons maintenant l'envoi d'un e-mail sortant :

Terminal

```
~# telnet localhost 25
Trying ::1...
Connected to localhost.
Escape character is '^]'.
220 glmf ESMTP Postfix (Debian/GNU)
HELO test.moi
250 glmf
MAIL FROM: <n-existe-pas@gmail.com>
250 2.1.0 Ok
RCPT TO: <destinataire@gmail.com>
250 2.1.5 Ok
DATA
354 End data with <CR><LF>.<CR><LF>
Subject: Essai vers l'exterieur

Contenu vers l'exterieur
.
250 2.0.0 Ok: queued as 0FFB361449
QUIT
221 2.0.0 Bye
Connection closed by foreign host.
```

Cet e-mail arrive bien à destination.





16 Vérification dans les journaux

Postfix envoie ses journaux à **syslog**, qui les stocke dans le fichier **/var/log/mail.log**. Dans celui-ci, on retrouve la trace des trois e-mails de test vus ci-dessus :

Fichier

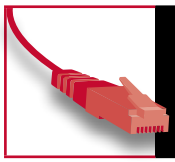
```
May 31 09:34:32 glmf postfix/smtpd[2488]: connect from
localhost[::1]
May 31 09:34:58 glmf postfix/smtpd[2488]: 400C961449:
client=localhost[::1]
May 31 09:35:12 glmf postfix/cleanup[2492]: 400C961449: message-
id=<20130531073458.400C961449@glmf>
May 31 09:35:12 glmf postfix/qmgr[2206]: 400C961449: from=<test@
test.moi>, size=331, nrcpt=1 (queue active)
May 31 09:35:12 glmf postfix/local[2493]: 400C961449: to=<pdupont@
monsite.fr>, relay=local, delay=23, delays=23/0.04/0/0.01, dsn=2.0.0,
status=sent (delivered to maildir)
May 31 09:35:12 glmf postfix/qmgr[2206]: 400C961449: removed
May 31 09:35:14 glmf postfix/smtpd[2488]: disconnect from
localhost[::1]
[...]
May 31 09:41:04 glmf postfix/smtpd[2502]: connect from
localhost[::1]
May 31 09:41:20 glmf postfix/smtpd[2502]: NOQUEUE: reject: RCPT from
localhost[::1]: 550 5.1.1 <inconnu@monsite.fr>: Recipient address
rejected: User unknown in local recipient table; from=<test@test.
moi> to=<inconnu@monsite.fr> proto=SMTP helo=<test.moi>
May 31 09:41:24 glmf postfix/smtpd[2502]: disconnect from
localhost[::1]
[...]
May 31 09:42:19 glmf postfix/smtpd[2502]: connect from
localhost[::1]
May 31 09:42:39 glmf postfix/smtpd[2502]: 0FFB361449:
client=localhost[::1]
May 31 09:42:54 glmf postfix/cleanup[2510]: 0FFB361449: message-
id=<20130531074239.0FFB361449@glmf>
May 31 09:42:54 glmf postfix/qmgr[2206]: 0FFB361449: from=<n-existe-
pas@gmail.com>, size=346, nrcpt=1 (queue active)
May 31 09:42:55 glmf postfix/smtp[2511]: connect to smtp.free.
fr[2a01:e0c:1:1599::10]:25: Network is unreachable
May 31 09:42:56 glmf postfix/smtpd[2502]: disconnect from
localhost[::1]
May 31 09:42:58 glmf postfix/smtp[2511]: 0FFB361449:
to=<destinataire@gmail.com>, relay=smtp.free.fr[212.27.48.4]:25,
delay=26, delays=23/0.04/0.78/2.3, dsn=2.0.0, status=sent (250 2.0.0
Ok: queued as 0F622D48204)
May 31 09:42:58 glmf postfix/qmgr[2206]: 0FFB361449: removed
```

On constate que les deux e-mails acceptés ont suivi un cheminement à travers les différents processus de Postfix, chacun enregistrant sa trace :

- **smtpd** accepte les connexions et les e-mails ;
- **cleanup** ajoute les entêtes manquants et met les e-mails en file d'attente ;
- **qmgr** gère la file d'attente des e-mails ;
- **local** délivre les e-mails localement ;
- **smtp** délivre les e-mails à distance.

Le courrier électronique vers une adresse inconnue a, quant à lui, été rejeté par **smtpd** directement.

Chaque e-mail a un identifiant unique dans l'ensemble de la procédure, on peut alors suivre le cheminement de chaque e-mail dans ces processus.



17 Alias

Une adresse en **pdupont@monsite.fr** c'est sympa, mais **pierre.dupont@monsite.fr** serait encore plus pratique ! Postfix est configuré pour utiliser le fichier **/etc/aliases** dans ce but (voir les directives **alias_maps** et **alias_databases** du fichier **main.cf**).

Ajoutons alors une ligne à **/etc/aliases** :

Fichier

```
pierre.dupont: pdupont
```

Puis, exécutons :

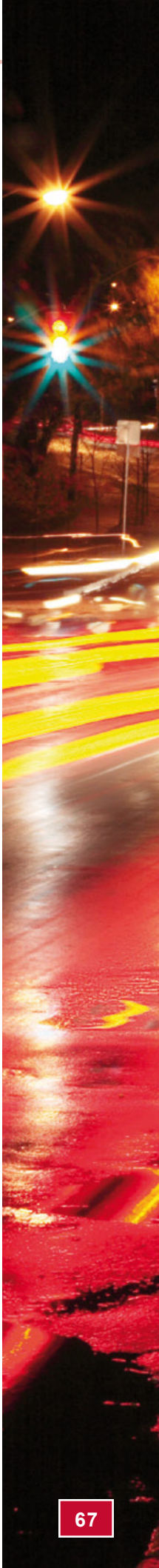
Terminal

```
~# newaliases
```

On retrouve alors ce courrier électronique dans les logs, dirigé vers le bon utilisateur :

Fichier

```
[...]
May 31 09:53:32 glmf postfix/qmgr[2206]: D23FA61449: from=<test@
test.moi>, size=335, nrcpt=1 (queue active)
May 31 09:53:32 glmf postfix/local[2525]: D23FA61449:
to=<pdupont@monsite.fr>, orig_to=<pierre.dupont@monsite.fr>,
relay=local, delay=26, delays=26/0.02/0/0, dsn=2.0.0, status=sent
(delivered to maildir)
[...]
```

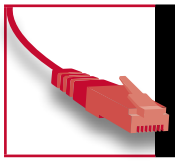


LE SERVEUR IMAP DE LA SUITE COURIER

Parmi les nombreux serveurs IMAP et POP disponibles, on peut distinguer The Courier Mail Server, qui est en réalité une suite complète de logiciels. Son serveur IMAP, simplement nommé Courier-IMAP, est relativement populaire pour une utilisation avec un MTA/MDA existant, comme ce que nous abordons dans ce numéro.

The Courier Mail Server, dont la première version date de 1999, est un serveur de messagerie partiellement modulaire : certaines parties peuvent être installées de manière totalement indépendante du reste de la suite, c'est le cas du serveur IMAP.

En réalité, c'est bien le serveur IMAP de Courier qui est la partie la plus populaire de l'ensemble de la suite, même si le projet dans son ensemble peut être utilisé de manière totalement autonome. C'est cet élément que nous allons utiliser dans ce tutoriel, afin de desservir les e-mails stockés au format **Maildir** stockés par un MTA/MDA tel que Postfix ou Exim.



1 Installer Courier-IMAP

Comme nous l'avons vu, nous n'allons installer que le serveur IMAP de la suite Courier :

Terminal
```
~# apt-get install courier-imap
```

Cet élément dépend toutefois des éléments de base de la suite The Courier Mail Server...

La suite Courier, lorsqu'elle est utilisée dans son ensemble, permet une administration web. Cependant, nous n'allons utiliser que le serveur IMAP et cette interface ne sera pas utile. C'est pourquoi nous indiquons qu'il ne faut pas créer les répertoires nécessaires à l'administration web.

Lorsque le pare-feu est en place avec Ufw, il faut autoriser le protocole IMAP :

Terminal
```
~# ufw allow IMAP
```

Courier dessert par défaut les e-mails stockés au format **Maildir**. On peut confirmer cela en regardant la dernière ligne du fichier **/etc/courier/imapd** :

Fichier
```
MAILDIRPATH=Maildir
```

Courier place des fichiers dans le répertoire **Maildir** des utilisateurs, permettant de meilleures performances :

Terminal
```
~# ls -d /home/toto/Maildir/courier*
/home/toto/Maildir/courierimapkeywords
/home/toto/Maildir/courierimapuiddb
```

2 FAM

Pour offrir de meilleures performances dans le cadre de connexions IDLE, le logiciel FAM est nécessaire : il permet à Courier de surveiller en temps réel les changements effectués dans les boîtes de messagerie des utilisateurs connectés.

Terminal

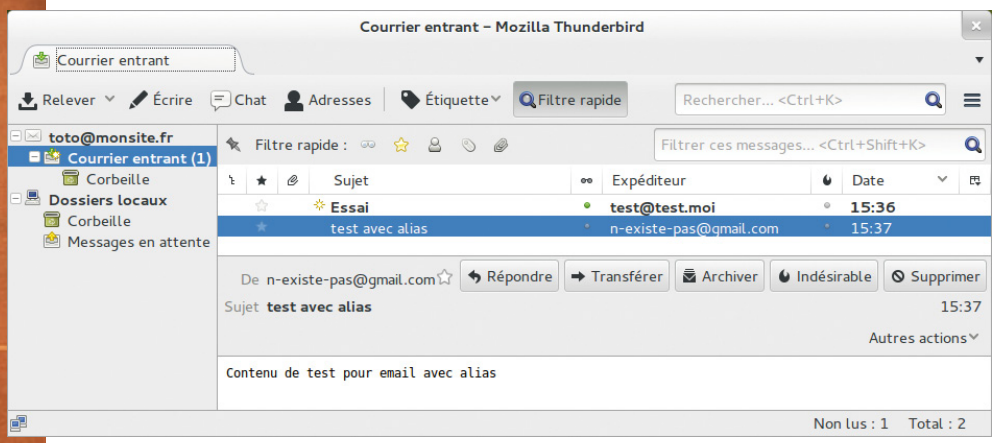
```
~# apt-get install fam
```

À savoir

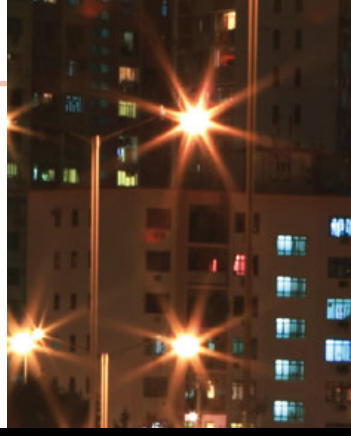
La fonctionnalité IDLE permet de ne pas fermer la connexion au serveur ; au lieu de cela, le client IMAP reste connecté et attend des informations de la part du serveur : lorsqu'un courrier arrive, le client est immédiatement prévenu.

3 Client de messagerie

Lorsque l'on se connecte alors à ce serveur avec un client de messagerie utilisant le protocole IMAP, on voit les e-mails que l'on a envoyés lors des tests effectués pendant la mise en place du MTA :



Par défaut, Courier-IMAP ne chiffre pas la communication : la connexion est totalement non sécurisée. D'ailleurs, lorsque l'on configure ce compte avec Thunderbird, on est accueilli par une alerte assez parlante...



4 Chiffrement de la connexion

Pour pouvoir chiffrer la connexion, il faut installer la version SSL de Courier-IMAP :

Terminal
```
~# apt-get install courier-imap-ssl
```

L'installeur de **courier-ssl** (dont dépend **courier-imap-ssl**) indique qu'un certificat SSL est requis. Il indique qu'un certificat auto-signé est automatiquement généré.

Afin d'arrêter le serveur IMAPS sur le port 993 et de forcer l'utilisation de STARTTLS sur le port classique, les lignes **IMAPDSSLSTART** et **IMAP_TLS_REQUIRED** sont modifiées dans le fichier **/etc/courier/imapd-ssl** :

Fichier
```
IMAPDSSLSTART=NO
IMAP_TLS_REQUIRED=1
```

Le service est ensuite redémarré :

Terminal
```
~# service courier-imap-ssl restart
[ ok ] Stopping Courier IMAP-SSL server: imapd-ssl.
```

On remarque que le serveur **imapd-ssl** (dont le rôle est d'écouter sur le port 993) ne se relance pas : il est devenu inutile.

Pour spécifier un certificat SSL différent, il faut modifier la directive **TLS_CERTFILE** du fichier **/etc/courier/imapd-ssl**, sa valeur par défaut étant la suivante :

Fichier
```
TLS_CERTFILE=/etc/courier/imapd.pem
```

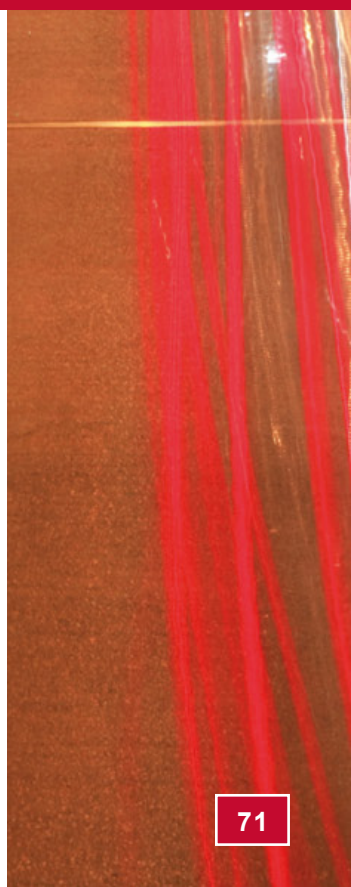
Il faut bien sûr redémarrer **courier-imap** après avoir modifié cette directive :

Terminal
```
~# service courier-imap restart
```

À savoir

SSL OU STARTTLS

Historiquement, les communications IMAP ou POP à protéger étaient entièrement encapsulées dans un flux SSL, on parlait alors d'IMAPS ou de POPS (au même titre que HTTPS). Cependant, cette technique tend à être abandonnée au profit de STARTTLS, qui permet d'avoir un seul démon en écoute, sur le port IMAP ou POP classique : c'est immédiatement après la connexion que le serveur annonce que STARTTLS est utilisé, la communication devient alors chiffrée à ce moment-là, avant même la communication de l'identifiant auquel on se connecte.



4

4 LES AUTRES SERVICES UTILES

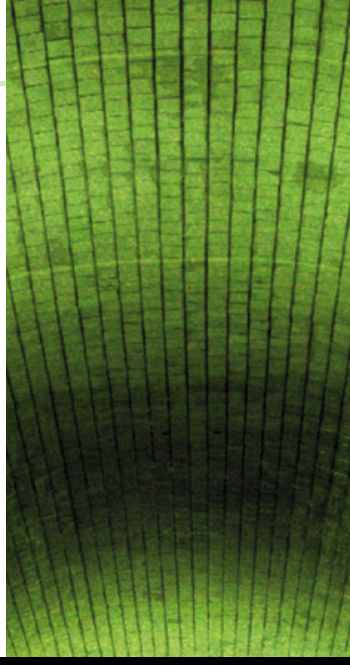
À découvrir dans cette partie...

4 LES AUTRES SERVICES UTILES

L'ÉCHANGE DE FICHIERS SÉCURISÉ AVEC VSFPTD

Le protocole FTP est très utilisé lorsqu'il s'agit de mettre en ligne un site web ; on l'utilise également lorsque l'on veut télécharger des fichiers volumineux. De nombreux serveurs FTP existent sous Linux, intéressons-nous à vsFTPd, l'un des plus populaires.

VsFTPd (*very secure FTP daemon*), créé en 2000, est un serveur FTP pensé pour être sécurisé, performant et stable. C'est notamment l'un des premiers logiciels serveurs à mettre en œuvre la séparation des privilèges, permettant de réduire les risques de piratage en ne donnant aux exécutables que les droits dont ils ont besoin (pas de démon en écoute fonctionnant *en tant que root* par exemple) – c'est aujourd'hui considéré comme une nécessité lorsque l'on développe un serveur. Il est développé par Chris Evans, chargé de la sécurité de Chrome chez Google.



1 Installer VsFTPd

VsFTPd est proposé par Debian en paquet :

Terminal
```
~# apt-get install vsftpd
```

Si le pare-feu est activé avec Ufw, il faut ajouter une règle pour autoriser le protocole FTP ; la configuration d'Ufw sous Debian ne propose pas d'application spécifique à ce protocole, il faut donc donner le numéro de port :

Terminal
```
~# ufw allow 21
```

2 Configuration par défaut

La configuration par défaut de VsFTPd n'autorise que les connexions anonymes. Il dessert alors le contenu du répertoire de base de l'utilisateur **ftp**, qui est configuré sur **/srv/ftp** si cet utilisateur n'existait pas avant l'installation de VsFTPd. Créons un fichier dans ce répertoire et testons l'accès...

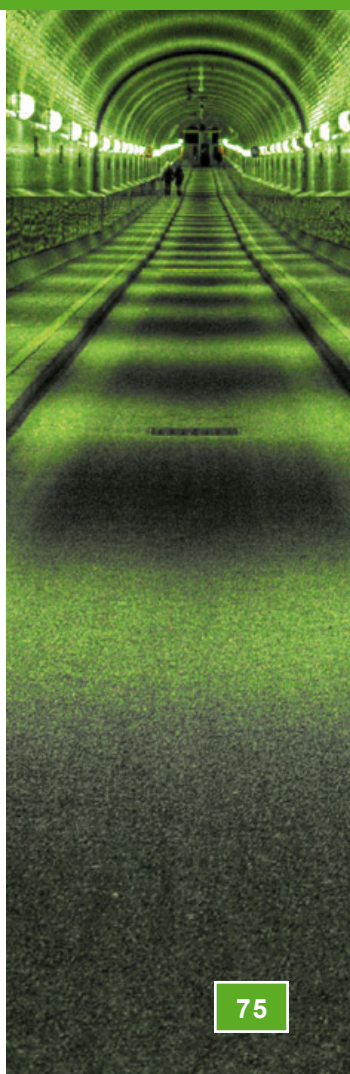
Terminal
```
~# echo "Hello world" > /srv/ftp/README.Test
```

Si l'on essaie de s'y connecter à partir d'une autre machine, seul le nom d'utilisation **anonymous** est autorisé :

Terminal
```
~$ ftp 192.168.42.37
Connected to 192.168.42.37.
220 (vsFTPd 2.3.5)
Name (192.168.42.37:toto): toto
```

À savoir

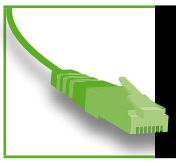
Par défaut, le protocole FTP n'est pas du tout sécurisé : autant le mot de passe que le contenu des fichiers transférés sont lisibles « en clair » sur le réseau. Pour assurer une sécurité des transferts, il faut activer le chiffrement SSL.



Terminal

```
530 This FTP server is anonymous only.
Login failed.
ftp> user anonymous
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> ls
200 PORT command successful. Consider using PASV.
150 Here comes the directory listing.
-rw-r--r--    1 0        0              12 Jun 04 11:58 README.Test
226 Directory send OK.
ftp> exit
221 Goodbye.
```

Si l'objectif est de proposer des fichiers en téléchargement public sans aucune possibilité de téléversement, on peut s'arrêter là !



3 Fichier de configuration

Toute la configuration de VsFTPd s'effectue dans le fichier **/etc/vsftpd.conf**.
Ce fichier accepte des directives avec la syntaxe suivante :

Fichier

```
nom=valeur
```

Ce fichier peut également contenir des lignes commentées, commençant par un dièse :

Fichier

```
# Ceci est un commentaire
```

Après toute modification à ce fichier, la configuration de VsFTPd doit être rechargée :

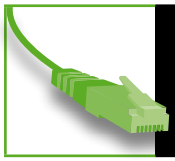
Terminal

```
~# service vsftpd reload
```

VsFTPd enregistre toutes les opérations dans son journal qui correspond au fichier **/var/log/vsftpd.conf**. Ce fichier peut par exemple contenir les entrées suivantes, qui correspondent à une connexion puis un téléversement :

Fichier

```
Tue Jun  4 12:53:15 2013 [pid 2] CONNECT: Client "192.168.42.43"
Tue Jun  4 12:53:15 2013 [pid 1] [toto] OK LOGIN: Client "192.168.42.43"
Tue Jun  4 12:53:18 2013 [pid 3] [toto] OK UPLOAD: Client
"192.168.42.43", "/home/toto/test.txt", 143 bytes, 846.61Kbyte/sec
```



4 Définir les accès

Il est possible de changer le répertoire auquel les utilisateurs anonymes se connectent grâce à la directive **anon_root** à ajouter au fichier **/etc/vsftpd.conf** ; elle peut par exemple être configurée de la manière suivante :

Fichier

```
anon_root=/srv/racine_anonyme
```

L'utilisateur **ftp** n'a pas nécessairement besoin d'être propriétaire de ce répertoire : il a uniquement besoin d'en lire le contenu (droits de lecture).

Pour permettre aux utilisateurs locaux du serveur de se connecter, la directive à modifier dans le fichier **/etc/vsftpd.conf** est **local_enable** :

Fichier

```
local_enable=YES
```

Par défaut, VsFTPd interdit le téléversement (l'écriture) de manière globale. Pour l'autoriser, la directive est la suivante :

Fichier

```
write_enable=YES
```

Pour interdire l'accès anonyme, il faut configurer la directive **anonymous_enable** à **NO** :

Fichier

```
anonymous_enable=NO
```

Attention, il ne suffit pas de commenter cette ligne : si elle n'est pas définie, l'authentification anonyme est autorisée.

Par défaut, l'utilisateur **anonymous** n'a que le droit de télécharger le contenu du répertoire qui lui est présenté. Il est possible de l'autoriser à créer des fichiers, des répertoires, etc.

Pour autoriser **anonymous** à téléverser des fichiers sur le serveur, la directive est la suivante :

Fichier

```
anon_upload_enable=YES
```

Pour autoriser **anonymous** à créer des répertoires sur le serveur, la directive est la suivante :

Fichier

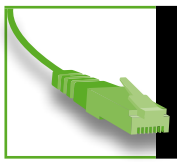
```
anon_mkdir_write_enable=YES
```

VsFTPd permet de changer automatiquement le propriétaire des fichiers téléversés de manière anonyme ; de cette façon, une personne peut téléverser un fichier et être sûre

qu'aucune autre ne pourra modifier son fichier. Pour cela, il faut configurer la directive **chown_uploads** à **YES** et **chown_username** au nom de l'utilisateur de destination (l'utilisateur **root** est déconseillé) :

Fichier

```
chown_uploads=YES
chown_username=toto
```



5 Chiffrement

Pour activer le chiffrement des communications, la directive **ssl_enable** est à ajouter :

Fichier

```
ssl_enable=YES
```

Il faut également créer un certificat SSL ; la configuration par défaut de VsFTPd va chercher ce certificat dans le fichier **/etc/ssl/private/vsftpd.pem**, créons donc ce fichier :

Terminal

```
~# apt-get install ssl-cert
[...]
~# make-ssl-cert /usr/share/ssl-cert/ssleay.cnf /etc/ssl/private/
vsftpd.pem
```

Comme lorsque nous avons créé des certificats SSL pour les serveurs web, il faut renseigner le nom d'hôte pour lequel ce certificat est créé. De cette manière, les utilisateurs locaux n'auront pas le droit de se connecter sans chiffrement SSL...

Le client en ligne de commandes **ftp** ne peut d'ailleurs plus être utilisé, il ne supporte pas le chiffrement SSL :

Terminal

```
~$ ftp 192.168.42.37
Connected to 192.168.42.37.
220 (vsFTPd 2.3.5)
Name (192.168.42.37:toto): toto
530 Non-anonymous sessions must use encryption.
Login failed.
ftp> 421 Service not available, remote server has closed
connection
```

Pour se connecter en FTP sécurisé en SSL à partir d'une ligne de commandes, on peut utiliser **lftp** :

Terminal

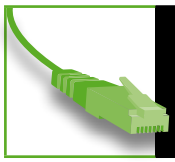
```
~$ sudo apt-get install lftp
```

Pour autoriser les certificats auto-signés, il faut modifier la configuration globale de **lftp** en ajoutant la ligne suivante dans le fichier **/etc/lftp.conf** (sur le poste de travail bien sûr et non sur le serveur) :

Fichier

```
set ssl:verify-certificate off
```

Cette étape est bien sûr valable quel que soit le serveur FTP, vu qu'il ne concerne que le client.



6 SSL et pare-feu

Le protocole FTP est relativement complexe : toutes les communications ne passent pas par un seul port. Dès qu'un transfert de données est demandé par le client (liste des fichiers d'un répertoire, contenu d'un fichier, etc.), le serveur indique un port auquel se connecter : le client se connecte alors sur le port indiqué pour la transmission des données, chaque port étant dédié à un transfert.

Netfilter, le pare-feu de Linux, est capable de « suivre » ces transferts : c'est un pare-feu à états. Un tel pare-feu est capable de lire le contenu de certaines communications et d'autoriser les communications liées ; c'est le cas du protocole FTP, avec lequel Netfilter autorise dynamiquement les flux sur les ports annoncés par le serveur FTP.

Cependant, ce mécanisme est impossible lorsque le flux est chiffré par SSL : Netfilter, comme tout autre logiciel tiers, est incapable de lire le contenu d'un message chiffré. Il ne peut alors plus ouvrir les ports dynamiquement, les transferts ne passent pas.

Pour autoriser les transferts lorsque les communications sont chiffrées, il faut alors explicitement définir une plage de ports parmi lesquels vsFTPd pourra piocher et les autoriser inconditionnellement.

Nous allons d'abord indiquer une plage précise à vsFTPd – choisissons les ports 30000 à 30500 – dans le fichier **/etc/vsftpd.conf** :

Fichier

```
pasv_min_port=30000
pasv_max_port=30500
```

La configuration doit bien sûr être rechargée :

Terminal

```
~# service vsftpd reload
```

Ensuite, la plage de ports est explicitement autorisée avec Ufw :

Terminal

```
~# ufw allow proto tcp to any port 30000:30500
```

Notons que cela n'autorisera que 501 transferts simultanés ; un client pouvant initier plusieurs transferts simultanés (notamment les logiciels graphiques, qui permettent de lancer plusieurs téléchargements en même temps tout en explorant le contenu du serveur), cela veut dire qu'on ne permet alors que 501 clients en même temps au maximum, probablement beaucoup moins.

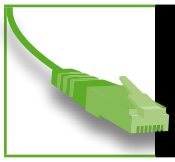
Pour en savoir plus sur la configuration de vsFTPd, n'hésitez pas à consulter le manuel du fichier de configuration (**man vsftpd.conf**) ! ■

4 LES AUTRES SERVICES UTILES

PURE-FTPD, UN SERVEUR FTP PLUS SIMPLE À CONFIGURER

Parmi les nombreux serveurs FTP libres existants, l'un des plus populaires – outre vsFTPd – est Pure-FTPd. Celui-ci se focalise sur l'efficacité et la facilité d'utilisation. Découvrons ce second « poids lourd » parmi les serveurs FTP...

Basé à l'origine sur Troll-FTPd (qui fut développé par Arnt Gulbrandsen lorsqu'il travaillait chez Trolltech entre 1995 et 1999), Pure-FTPd est d'abord un ensemble de patches appliqués à celui-ci, à partir de 2001. Il évolue bien sûr maintenant de manière totalement indépendante, géré par son équipe propre et notamment son créateur, Frank Denis. Sa caractéristique la plus visible est de pouvoir être lancé facilement en ligne de commandes, sans s'appuyer sur des fichiers de configuration.



1 Installer Pure-FTPd

Pure-FTPd est disponible sous quatre formes différentes, chacune dans un paquet :

- **pure-ftpd** est le serveur FTP classique ;
- **pure-ftpd-ldap** est le serveur avec authentification LDAP ;
- **pure-ftpd-mysql** permet l'authentification à partir d'une base MySQL ;
- **pure-ftpd-postgresql** offre la possibilité d'authentifier les utilisateurs par une base PostgreSQL.

Nous n'allons pas nous intéresser aux possibilités d'authentification avancées et allons donc installer la version de base :

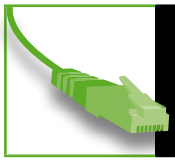
Terminal

```
~# apt-get install pure-ftpd
```

Avec un pare-feu activé, il faut autoriser les connexions sur le port FTP 21 :

Terminal

```
~# ufw allow 21
```



2 Configuration et paramètres

En réalité, Pure-FTPd ne se base sur aucun fichier de configuration : son paramétrage se fait entièrement par l'intermédiaire d'arguments qui lui sont donnés lors de son démarrage.

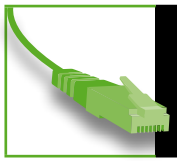
Le paquet Debian de Pure-FTPd propose une approche originale à sa configuration : chaque fichier dans le répertoire **/etc/pure-ftpd/conf** correspond à un paramètre, il contient alors une seule ligne qui indique la valeur du paramètre. Une fois un fichier ajouté ou modifié dans **/etc/pure-ftpd/conf**, il faut redémarrer Pure-FTPd :

Terminal

```
~# service pure-ftpd restart
Restarting ftp server: Running: /usr/sbin/pure-ftpd -l pam -8
UTF-8 -u 1000 -O clf:/var/log/pure-ftpd/transfer.log -E -B
```

Lors du redémarrage de Pure-FTPd, on voit les arguments qui lui sont donnés, traductions des paramétrages effectués dans ces fichiers. Par défaut, le paramétrage de Pure-FTPd selon Debian interdit les connexions anonymes et autorise toutes les manipulations (téléchargement, téléversement, création de répertoires...) pour les utilisateurs définis dans le système.

Les informations de connexion sont envoyées à Syslog (stockées par défaut dans **/var/log/syslog**), tandis que les détails sur les transferts sont enregistrés dans le fichier dont le chemin est dans le paramètre **AltLog**, sa valeur par défaut étant **/var/log/pure-ftpd/transfer.log**.



3 Accès anonyme

Pour autoriser les accès anonymes, il faut changer la valeur dans **/etc/pure-ftpd/conf/NoAnonymous** à **no**. Il faut également créer soi-même un utilisateur s'appelant **ftp** :

Terminal
```
~# adduser ftp --disabled-login
```

Une connexion anonyme sera alors confinée dans le répertoire de base de cet utilisateur (**/home/ftp** par défaut).

Pour n'autoriser que l'accès anonyme et interdire tout accès avec des utilisateurs existant sur le système, c'est le fichier **/etc/pure-ftpd/conf/AnonymousOnly** qui doit être créé, avec la valeur **yes**. Cela peut se faire avec la commande suivante :

Terminal
```
~# echo "yes" > /etc/pure-ftpd/conf/AnonymousOnly
```

Toute connexion sera alors anonyme, quel que soit le nom d'utilisateur qui aura été fourni.

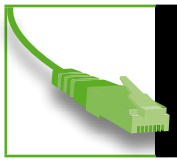
Par défaut, l'utilisateur anonyme peut téléverser des fichiers mais ne peut pas créer de répertoire. Pour autoriser la création de répertoire, le fichier est **AnonymousCanCreateDirs** :

Terminal
```
~# echo "yes" > /etc/pure-ftpd/conf/AnonymousCanCreateDirs
```

Pour interdire le téléversement, le fichier est **AnonymousCantUpload** :

Terminal
```
~# echo "yes" > /etc/pure-ftpd/conf/AnonymousCantUpload
```

Tout cela restant bien sûr suivi d'un redémarrage de Pure-FTPd.



4 Chiffrement

Pour activer le chiffrement des communications, le paramétrage se fait dans le fichier **/etc/pure-ftpd/conf/TLS**. La valeur à y indiquer peut être :

- **0** : pas de chiffrement ;

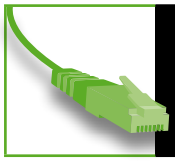
- **1** : autoriser les connexions chiffrées et les connexions non chiffrées ;
- **2** : refuser les connexions où l'authentification n'est pas chiffrée ;
- **3** : n'autoriser que les connexions intégralement chiffrées.

Pour un maximum de sécurité, utilisons la valeur **3** :

Terminal
```
~# echo "3" > /etc/pure-ftpd/conf/TLS
```

Le certificat SSL doit obligatoirement être situé dans le fichier **/etc/ssl/private/pure-ftpd.pem** ; ce paramètre ne peut pas être modifié. Créons un certificat auto-signé :

Terminal
```
~# apt-get install ssl-cert
[...]
~# make-ssl-cert /usr/share/ssl-cert/ssleay.cnf /etc/ssl/private/pure-ftpd.pem
```



5 SSL et pare-feu

Les explications du point 6 du tutoriel sur vsFTPd sont valables pour Pure-FTPd également : le pare-feu n'est pas capable de connaître le port dynamique qui est annoncé par le serveur au client pour les transferts. Pour spécifier une plage de ports à utiliser à Pure-FTPd, c'est le fichier **PassivePortRange** qui doit être utilisé ; celui-ci prend deux valeurs, séparées par un espace. Pour les ports 30000 à 30500, le paramétrage sera donc le suivant :

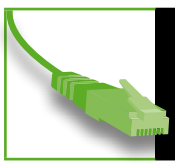
Terminal
```
~# echo "30000 30500" > /etc/pure-ftpd/conf/PassivePortRange
```

Pure-FTPd est ensuite redémarré :

Terminal
```
~# service pure-ftpd restart
Restarting ftp server: Running: /usr/sbin/pure-ftpd -l pam -p 30000:30500
-8 UTF-8 -u 1000 -Y 3 -O clf:/var/log/pure-ftpd/transfer.log -B
```

Ensuite, il reste à autoriser cette plage :

Terminal
```
~# ufw allow proto tcp to any port 30000:30500
```



6 En savoir plus…

Pour connaître la liste complète des noms des paramètres offerts par la « version Debian » de Pure-FTPd, consultez le manuel de **pure-ftpd-wrapper** (**man pure-ftpd-wrapper**) ; pour connaître la liste des paramètres que Pure-FTPd accepte, voyez plutôt sa page de manuel (**man pure-ftpd**). Les noms des fichiers de paramètres de Debian correspondent bien sûr aux versions longues des options de Pure-FTPd, que l'on voit au début de son manuel. ■

4 LES AUTRES SERVICES UTILES

VOTRE RÉSEAU PRIVÉ AVEC OPENVPN

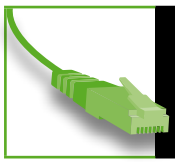
L'intérêt d'un réseau privé virtuel (VPN), c'est d'offrir une communication sécurisée entre deux sites distants, au travers d'Internet. OpenVPN est l'un des serveurs de VPN libres les plus populaires en raison de sa simplicité de configuration et d'utilisation.

OpenVPN a été imaginé par James Yonan au début du XXIème siècle, lorsqu'il a été autorisé par son employeur à travailler à distance ; il a alors beaucoup voyagé, se connectant aux systèmes de son employeur à travers différents réseaux, auxquels il ne faisait pas nécessairement confiance, notamment lorsque sa connexion traversait l'Asie, la Russie, etc.

À ce moment, on rencontrait deux écoles en matière de connexion distante : d'un côté les adeptes de la sécurité, qui mettaient en place des systèmes complexes mais bien protégés et de l'autre côté, les adeptes de la facilité d'utilisation, qui faisaient des concessions sur la sécurité. OpenVPN a été créé avec à l'esprit les deux aspects : facilité d'utilisation et sécurité.

Lorsque l'on se connecte à un serveur OpenVPN, le système crée une nouvelle interface réseau virtuelle, qui fonctionne comme n'importe quelle autre interface et qui a une adresse IP privée, connectée à un réseau virtuel géré par le serveur OpenVPN, les communications étant chiffrées par SSL.

OpenVPN supporte de nombreux modes de fonctionnement différents (tunnel IP, tunnel Ethernet, bridge Ethernet...), nous ne pouvons bien sûr aborder ici que l'un d'entre eux, nous nous concentrerons sur la mise en place d'un tunnel IP.



1 Installer OpenVPN

Avec OpenVPN, le même exécutable est utilisé pour le serveur et les clients. Installons OpenVPN sur le serveur :

Terminal
```
~# apt-get install openvpn
```

Installons ensuite OpenVPN sur le client (ici, sur un PC utilisant Ubuntu) :

Terminal
```
~$ sudo apt-get install openvpn
```

Par défaut, aucun serveur OpenVPN ne se lance et aucune connexion ne se fait, que ce soit sur le serveur ou sur le client.

Les réseaux virtuels d'OpenVPN sont paramétrés au travers de fichiers de configuration. Lors de la manipulation du service **`openvpn`** (par exemple avec **`service openvpn start`**), le script de démarrage va rechercher les fichiers **`.conf`** dans **`/etc/openvpn`** et exécuter une instance d'OpenVPN pour chaque fichier correspondant à ces critères.

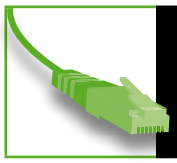
Au cours de ce tutoriel, nous allons créer un fichier de configuration de test, que nous placerons dans **`/etc/openvpn/monvpn.conf`**.

À savoir

IPsec est un ensemble de standards ouverts, fortement lié aux protocoles IPv6 et IPv4, dont l'objectif est de chiffrer les données transférées. La mise en œuvre d'IPsec est relativement complexe, ce protocole permet cependant parfois d'interconnecter des équipements de réseau de marques différentes, tant que ceux-ci respectent correctement ces standards.

OpenVPN n'est pas compatible avec IPsec. Il n'utilise pas ces standards, au profit d'une encapsulation : les communications d'OpenVPN se font au travers d'un tunnel qui utilise UDP ou TCP ; c'est par-dessus cette communication classique qu'un réseau virtuel est créé.

Sous Linux, pour le support d'IPsec, on peut notamment utiliser les logiciels OpenSWAN et StrongSWAN.



2 Configuration de base

Généralement, la configuration d'OpenVPN doit contenir les lignes suivantes (à placer dans le fichier **/etc/openvpn/monvpn.conf**) :

Fichier

```
port 1194
proto udp
dev tun
ca /chemin/vers/le/ca.crt
cert /chemin/vers/le/certificat.crt
key /chemin/vers/la/cle.key
dh /chemin/vers/les/parametres-diffie-hellman.pem
comp-lzo
persist-tun
persist-key
verb 3
mute 20
```

- Le **port** peut être modifié (il doit d'ailleurs l'être si plusieurs configurations sont en place sur une même machine, un seul démon à la fois pouvant écouter sur un port), 1194 étant le port officiellement attribué à OpenVPN ;
- Le **proto**cole peut être TCP dans les situations où UDP ne peut pas fonctionner, auquel cas la valeur doit être **tcp-server** (dans la configuration du client, cette valeur sera alors **tcp-client**) ;
- **dev** peut prendre deux valeurs : **tun** crée un tunnel IP (lien point-à-point, une sorte de mini-réseau reliant les deux parties du VPN, auto-configuré par OpenVPN), **tap** crée un tunnel Ethernet (une interface réseau qu'il faudra ensuite configurer, comme n'importe quelle interface réseau) ;
- Les paramètres **ca**, **cert**, **key** et **dh** sont liés au certificat SSL à utiliser, les détails de ces paramètres seront abordés plus bas ;
- Avec la directive **comp-lzo**, OpenVPN compresse les paquets traversant le lien VPN, améliorant ainsi légèrement le débit ;
- **persist-tun** et **persist-key** permettent de conserver certains éléments lors d'un redémarrage d'OpenVPN ;
- **verb** définit le niveau de verbosité d'OpenVPN dans les logs, de 0 (uniquement les erreurs fatales) à 9 (enregistrer un maximum d'informations) ;
- La directive **mute** permet de ne pas inscrire dans les logs un même message qui se répète ; sa valeur correspond au nombre maximum de répétitions d'un même message dans les logs.

Ces paramètres doivent être présents sur le client et sur le serveur.

Du côté du serveur, il faut indiquer quelle configuration réseau doit être mise en place, avec la directive **server** :

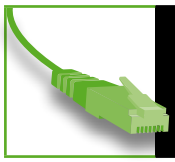
Fichier

```
server 10.8.0.0 255.255.255.0
```

Ici, le serveur aura l'adresse **10.8.0.1** et les clients auront des adresses dans le réseau **10.8.0.0/24**.

Il est possible de limiter le nombre de clients autorisés sur le serveur OpenVPN avec la directive **max-clients**, par exemple pour limiter ce nombre à 100 :

Fichier
```
max-clients 100
```



3 Routes et passerelle

Si le VPN est utilisé pour joindre des machines situées sur des réseaux internes, il est nécessaire de « pousser » les routes aux clients VPN ; par exemple :

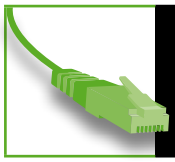
Fichier
```
push "route 192.168.10.0 255.255.255.0"
push "route 192.168.20.0 255.255.255.0"
```

Avec ces deux lignes, le serveur OpenVPN indiquera à ses clients que les réseaux **192.168.10.0/24** et **192.168.20.0/24** seront joignables au travers de cette connexion VPN.

Notons que, dans ce cas, les réseaux concernés doivent être capables de répondre à ces requêtes ; ils doivent donc avoir une route vers le réseau **10.8.0.0** qui pointe vers le serveur VPN si celui-ci n'est pas joignable au travers de leur passerelle par défaut.

Si tous les flux doivent passer par le VPN, alors il faut indiquer au client que le serveur VPN doit être configuré comme passerelle par défaut, avec la directive suivante :

Fichier
```
push "redirect-gateway def1 bypass-dhcp"
```



4 Communication entre les clients

Par défaut, un client OpenVPN ne voit que le serveur OpenVPN. Toute communication avec un autre équipement passe alors obligatoirement par l'interface TUN/TAP. Avec la directive suivante, OpenVPN route directement les communications entre deux clients VPN pour réduire la charge :

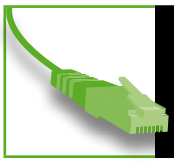
Fichier
```
client-to-client
```

Ce paramètre induit cependant obligatoirement que les clients se voient, se trouvent dans le même réseau. C'est pratique pour faire communiquer plusieurs télé-travailleurs, c'est à éviter si on souhaite uniquement permettre aux clients d'accéder à des ressources serveurs.

Par défaut, lorsqu'un client se connecte avec un certificat déjà utilisé, la connexion en cours est interrompue au profit du nouveau client. Cela permet de s'assurer qu'un seul utilisateur se connecte à la fois avec un même certificat.

Il est cependant possible d'autoriser plusieurs connexions avec le même certificat (ce qui permet à un utilisateur de se connecter plusieurs fois, par exemple s'il est équipé de plusieurs ordinateurs). Pour cela, la directive à utiliser est la suivante :

Fichier
```
duplicate-cn
```



5 Easy-RSA

Un certificat SSL doit être signé par une autorité de certification. OpenVPN est livré avec un ensemble de scripts permettant de gérer ces certificats, nommé Easy-RSA ; cet outil s'appuie sur la commande **openssl**. Copions cet ensemble dans le répertoire de configuration d'OpenVPN et installons OpenSSL :

Terminal
```
~# cp -a /usr/share/doc/openvpn/examples/easy-rsa/2.0
/etc/openvpn/easy-rsa
~# apt-get install openssl
```

Pour faciliter l'utilisation ultérieure d'Easy-RSA, les lignes de la fin du fichier **/etc/openvpn/easy-rsa/vars** peuvent être modifiées, par exemple :

Fichier
```
export KEY_COUNTRY="FR"
export KEY_PROVINCE="Alsace"
export KEY_CITY="Selestat"
export KEY_ORG="Ma pomme"
export KEY_EMAIL="toto@monsite.fr"
export KEY_OU="Ma pomme"
```

Easy-RSA est ensuite initialisé :

Terminal
```
~# cd /etc/openvpn/easy-rsa
/etc/openvpn/easy-rsa# . vars
NOTE: If you run ./clean-all, I will be doing a rm -rf on
/etc/openvpn/easy-rsa/keys
/etc/openvpn/easy-rsa# ./clean-all
```

À partir de là, tous les certificats et toutes les clés créés seront dans **/etc/openvpn/easy-rsa/keys**. Bien sûr, si vous avez acheté des certificats auprès d'une vraie autorité de certification, alors vous pouvez les utiliser directement et ne pas vous servir de Easy-RSA.

Créons ensuite un certificat d'autorité de certification :

Terminal
```
/etc/openvpn/easy-rsa# ./build-ca
[...]
Common Name (eg, your name or your server's hostname)
[changeme]:vpn.monsite.fr
Name [changeme]:VPN de mon site
Email Address [toto@monsite.fr]:
```

Une fois l'autorité de certification existante, créons un certificat pour le serveur, qui sera alors signé par l'autorité de certification :

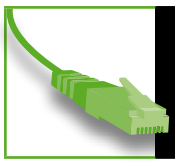
Terminal

```
/etc/openvpn/easy-rsa# ./build-key-server monvpn
[...]
Common Name (eg, your name or your server's hostname) [monvpn]:vpn.monsite.fr
Name [changeme]:Mon VPN
[...]
Certificate is to be certified until Jun  3 16:56:25 2023 GMT (3650 days)
Sign the certificate? [y/n]:y

1 out of 1 certificate requests certified, commit? [y/n]y
Write out database with 1 new entries
Data Base Updated
```

Générons ensuite les paramètres Diffie-Hellman :

Terminal

```
/etc/openvpn/easy-rsa# ./build-dh
Generating DH parameters, 1024 bit long safe prime, generator 2
This is going to take a long time
.................................................++*++*++*
```



6 Certificats clients

Une fois le certificat de l'autorité de certification et le certificat serveur créés, il faut créer un certificat pour chaque client. Encore une fois, Easy-RSA propose un script permettant de créer ce certificat :

Terminal

```
/etc/openvpn/easy-rsa# ./build-key client-toto
[...]
Common Name (eg, your name or your server's hostname) [client-toto]:
Name [changeme]:Toto
Email Address [toto@monsite.fr]:
[...]
Certificate is to be certified until Jun  3 17:39:35 2023 GMT (3650 days)
Sign the certificate? [y/n]:y

1 out of 1 certificate requests certified, commit? [y/n]y
Write out database with 1 new entries
Data Base Updated
```

Le nom de fichier ne doit bien sûr pas obligatoirement commencer par **client** : on met ici ce préfixe pour l'identifier précisément, c'est une habitude que chacun choisira de prendre ou non.

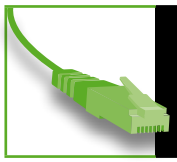
Notons que plusieurs commandes peuvent être utilisées pour générer le certificat client :

- **build-key**, comme dans cet exemple, qui crée un certificat SSL classique ;
- **build-key-pass**, qui permet de mettre en place un mot de passe qui sera demandé à l'utilisateur pour se connecter avec ce certificat ;

- **build-key-pkcs12**, qui permet de créer un fichier PKCS #12, contenant à lui seul l'ensemble des fichiers nécessaires pour le certificat client.

Une fois le certificat client créé, il faut copier sur l'ordinateur client (ou transmettre à l'utilisateur, de manière sécurisée bien sûr) :

- dans le cas d'un certificat classique, les fichiers **ca.crt**, **<nom du client>.crt** et **<nom du client>.key** ;
- dans le cas d'un fichier PKCS #12, le fichier **<nom du client>.p12**.



7 Récapitulatif

Nous pourrons retrouver la configuration suivante pour le serveur (fichier **/etc/openvpn/monvpn.conf**) :

Fichier

```
port 1194
proto udp
dev tun
comp-lzo
persist-tun
persist-key
ca /etc/openvpn/easy-rsa/keys/ca.crt
cert /etc/openvpn/easy-rsa/keys/monvpn.crt
key /etc/openvpn/easy-rsa/keys/monvpn.key
dh /etc/openvpn/easy-rsa/keys/dh1024.pem
server 10.8.0.0 255.255.255.0
push "route 192.168.10.0 255.255.255.0"
verb 3
mute 20
```

... ainsi que la configuration suivante pour le client :

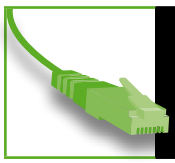
Fichier

```
proto udp
dev tun
comp-lzo
client
ca ca.crt
cert client-toto.crt
key client-toto.key
remote 12.34.56.78
```

Dans le cas d'un fichier PKCS #12, la configuration pour le client sera plutôt :

Fichier

```
proto udp
dev tun
comp-lzo
tls-client
pkcs12 vpn/client-toto.p12
remote 12.34.56.78
```



8 Démarrage du serveur

Une fois le serveur configuré, il peut être démarré :

Terminal

```
~# service openvpn start
[ ok ] Starting virtual private network daemon: monvpn.
```

Une fois le serveur OpenVPN en fonction, on peut constater que la route pour le réseau virtuel est en place ; il utilise l'interface virtuelle **tun0** :

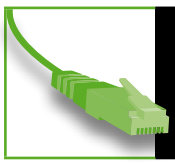
Terminal

```
~# ip route
default via 192.168.42.254 dev eth0
10.8.0.0/24 via 10.8.0.2 dev tun0
10.8.0.2 dev tun0  proto kernel  scope link  src 10.8.0.1
192.168.42.0/24 dev eth0  proto kernel  scope link  src
192.168.42.37
```

Si le pare-feu est activé avec Ufw, il faut autoriser la connexion au port défini dans la configuration d'OpenVPN :

Terminal

```
~# ufw allow 1194
Rule added
Rule added (v6)
```



9 Connexion du client

Du côté du client, les fichiers de certificat, ainsi que le fichier de configuration, sont placés dans un répertoire utilisateur, nommé par exemple **vpn** ; le client est lancé en lui donnant le fichier de configuration comme argument :

Terminal

```
~$ cd vpn
~/vpn$ ls
ca.crt  client-toto.crt  client-toto.key  clientvpn.conf
~/vpn$ sudo openvpn clientvpn.conf
Wed Jun  5 20:42:45 2013 OpenVPN 2.2.1 x86_64-linux-gnu [SSL]
[LZO2] [EPOLL] [PKCS11] [eurephia] [MH] [PF_INET6] [IPv6 payload
20110424-2 (2.2RC2)] built on Feb 13 2013
[...]
Wed Jun  5 20:42:47 2013 /sbin/ifconfig tun0 10.8.0.6 pointopoint
10.8.0.5 mtu 1500
Wed Jun  5 20:42:47 2013 Initialization Sequence Completed
```

Une fois la connexion effectuée, on peut vérifier que les routes configurées sur le serveur sont bien en place et qu'on peut bien effectuer une requête « ping » vers celui-ci (à effectuer dans un autre terminal, vu que le premier est occupé par le client OpenVPN) :

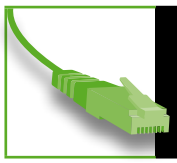
Terminal

```
~$ ip route
default via 192.168.42.254 dev eth0  proto static
10.8.0.1 via 10.8.0.5 dev tun0
10.8.0.5 dev tun0  proto kernel  scope link  src 10.8.0.6
169.254.0.0/16 dev eth0  scope link  metric 1000
192.168.10.0/24 via 10.8.0.5 dev tun0
192.168.42.0/24 dev eth0  proto kernel  scope link  src
192.168.42.43  metric 1
~$ ping -c 1 10.8.0.1
PING 10.8.0.1 (10.8.0.1) 56(84) bytes of data.
64 bytes from 10.8.0.1: icmp_req=1 ttl=64 time=0.688 ms

--- 10.8.0.1 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.688/0.688/0.688/0.000 ms
```

De même, à partir du serveur une requête « ping » fonctionne vers le client :

Terminal

```
~# ping -c 1 10.8.0.6
PING 10.8.0.6 (10.8.0.6) 56(84) bytes of data.
64 bytes from 10.8.0.6: icmp_req=1 ttl=64 time=0.486 ms

--- 10.8.0.6 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.486/0.486/0.486/0.000 ms
```



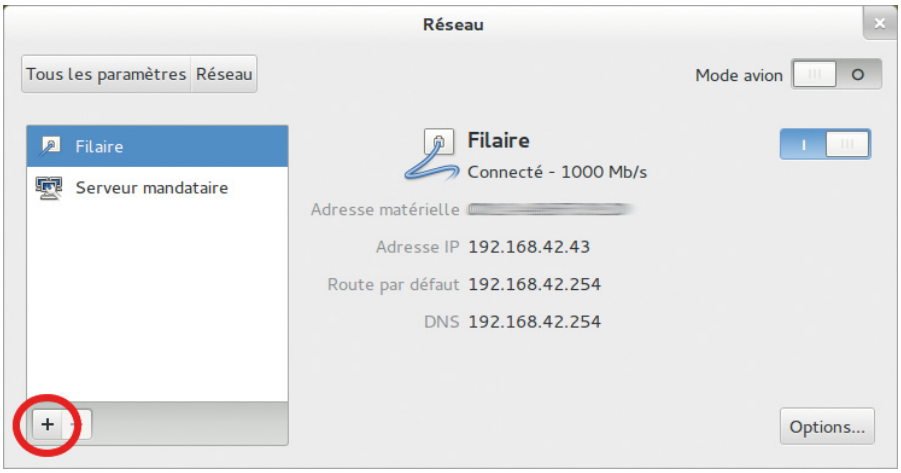
10 Configuration de NetworkManager

NetworkManager est l'outil installé dans différentes distributions pour gérer la configuration réseau de manière dynamique ; c'est notamment lui qui est utilisé sur Ubuntu. Installons sur notre PC Ubuntu le support OpenVPN de NetworkManager :

Terminal

```
~$ sudo apt-get install network-manager-openvpn
```

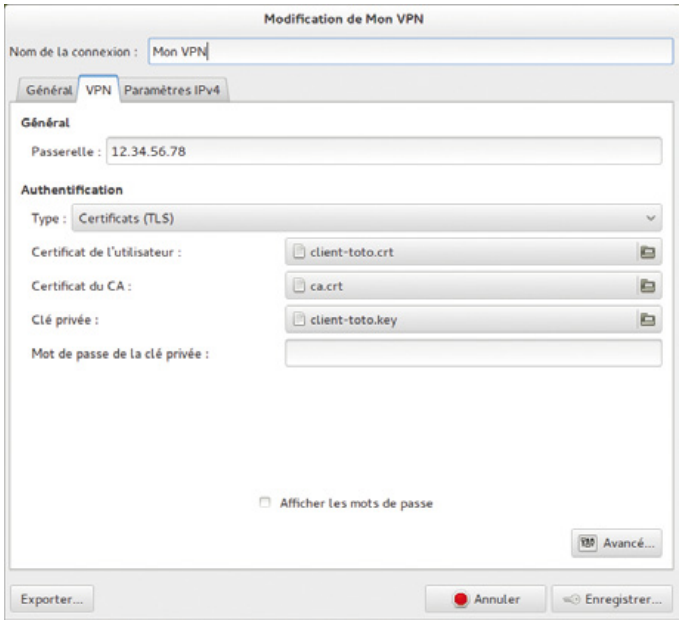
Ceci fait, dans les paramètres du réseau, cliquez sur le bouton « + » pour ajouter une connexion.



Le type d'interface à configurer est ***VPN*** : à cette étape, c'est généralement le seul choix possible. L'écran suivant permet de choisir le type de VPN à mettre en place : ici, on choisit ***OpenVPN***. L'écran suivant permet de configurer le VPN :

- Indiquez dans ***Nom de la connexion*** un nom explicite (pour affichage dans l'interface) ;
- Précisez dans ***Passerelle*** l'adresse IP ou le nom du serveur VPN ;
- Choisissez le type d'authentification ***Certificats (TLS)*** ;
- Dans ***Certificat de l'utilisateur***, ***Certificat du CA*** et ***Clé privée***, pointez vers les fichiers récupérés du serveur.

Validez l'ensemble en cliquant sur le bouton ***Enregistrer***. Il suffit ensuite de cliquer sur l'icône du réseau et de sélectionner la connexion ainsi créée. Une fois le VPN connecté, l'icône du réseau change afin de l'indiquer. Notons que, dans les captures d'écran d'exemple, l'environnement GNOME Shell est utilisé ; l'utilisation est similaire avec Unity. ■



4 LES AUTRES SERVICES UTILES

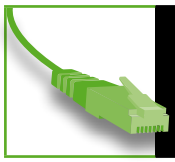
LA GESTION DE VERSIONS AVEC GIT

Toute équipe de développement efficace utilise un outil de gestion de versions afin d'effectuer le suivi des développements, de développer plusieurs aspects en parallèle, etc. De nombreux logiciels existent pour servir cet objectif, le plus populaire aujourd'hui étant indéniablement Git.

Git a été créé en 2005 par Linus Torvalds, afin de remplacer BitKeeper dans le cadre de la gestion des développements du noyau Linux.

C'est un logiciel de gestion de versions **décentralisé** : chaque développeur a une copie du dépôt sur son ordinateur, qu'il synchronise avec une autre copie – tous les développeurs ne sont pas obligés d'accéder à un unique serveur pour synchroniser le code source.

Comme tout logiciel de gestion de versions, Git mémorise l'historique complet de tous les fichiers contenus dans un projet, par le biais de révisions. Parmi ses fonctionnalités, on retrouve également la possibilité de gérer des branches parallèles, permettant ainsi de tester des modifications au code source sans pour autant le « casser ».



1 Installer Git

Git est proposé sous forme de paquet Debian...

Terminal
```
~# apt-get install git
```

Il faut également installer Git sur le client ; sur une machine Ubuntu, la commande sera :

Terminal
```
~$ sudo apt-get install git
```

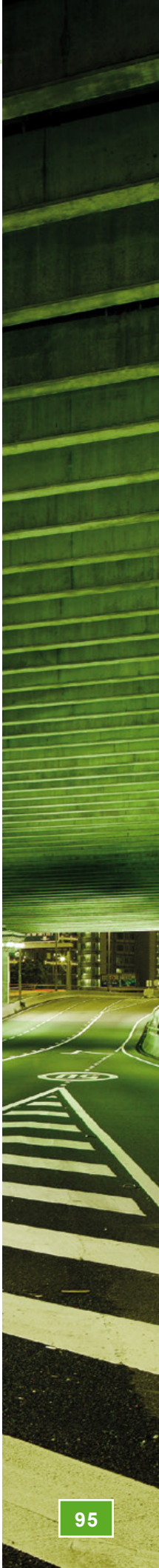
Git fonctionne de manière très simple, il n'a notamment pas de gestion multi-utilisateur : un dépôt Git ne peut être utilisé que par une et une seule personne. Sur un serveur, il est courant de créer un utilisateur **`git`** pour cela :

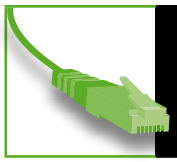
Terminal
```
~# adduser --shell /usr/bin/git-shell --disabled-password --home
/srv/git git
Ajout de l'utilisateur " git " ...
```

L'accès à ce dépôt se fera via le protocole SSH : on créera une clé SSH distincte pour chaque utilisateur devant accéder au dépôt ; tous se connecteront à l'utilisateur **`git`**.

Créons le fichier qui accueillera les clés des utilisateurs :

Terminal
```
~# su -m -c "mkdir /srv/git/.ssh" git
~# su -m -c "touch /srv/git/.ssh/authorized_keys" git
```





2 Initialisation d'un dépôt

Sur un serveur, pour initialiser un dépôt, il faut utiliser la commande **git init** de la manière suivante :

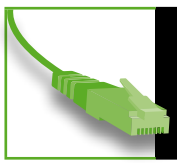
Terminal

```
~# su -m -c "mkdir /srv/git/monprojet; cd /srv/git/monprojet;
git init --bare; chown -R git.git /srv/git/monprojet"
Initialized empty Git repository in /srv/git/monprojet/
```

Si on vérifie le contenu du répertoire, on y trouve les fichiers qui constituent un dépôt Git :

Terminal

```
~# ls /srv/git/monprojet/
branches  config  description  HEAD  hooks  info  objects  refs
```



3 Clé SSH d'un client

Afin d'accéder à ce dépôt, il faut fournir une clé SSH au serveur. Commençons par en créer une, **sur un client** :

Terminal

```
~$ ssh-keygen -t rsa -C "Toto Dupont"
[...]
~$ cat .ssh/id_rsa.pub
ssh-rsa AAAAB3Nza[...]mSfSNmt Toto Dupont
```

L'argument **-C** à la commande **ssh-keygen** permet de définir un commentaire, qui permettra par la suite de reconnaître cette clé dans une liste qui peut devenir longue s'il y a de nombreux utilisateurs à autoriser.

Le résultat de la commande **cat .ssh/id_rsa.pub** est la clé publique, à transférer sur le serveur – ce résultat sera bien sûr différent chez chaque personne, la clé générée étant unique. Les paquets Debian et Ubuntu de SSH proposent la commande **ssh-copy-id**, qui permet de faire ce transfert de manière automatique **si l'on connaît le mot de passe de l'utilisateur destination**. Ce n'est pas notre cas, on ne peut donc pas utiliser cette commande.

La clé publique de l'utilisateur doit être injectée dans le fichier **authorized_keys** sur le serveur ; la commande suivante est utilisée :

Terminal

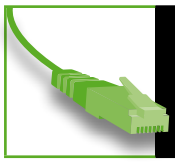
```
~# echo "ssh-rsa AAAAB3Nza[...]mSfSNmt Toto Dupont" >>
/srv/git/.ssh/authorized_keys
```

La commande est à adapter selon la clé de chacun, à placer entre les guillemets. Elle peut être exécutée autant de fois qu'il y a de clés à y copier, une par utilisateur et/ou par poste de travail.

À partir de ce moment-là, ce dépôt Git peut être utilisé par le poste de travail concerné. Sur ce dernier, les données d'identité de l'utilisateur sont créées :

Terminal
```
~$ git config --global user.email "toto@monsite.fr"
~$ git config --global user.name "Toto Dupont"
```

Cette manipulation n'est effectuée qu'une fois, pour l'ensemble des dépôts auxquels on est susceptible d'accéder.



4 Clone du dépôt

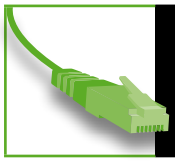
Pour cloner localement, **sur un poste de travail**, le dépôt du serveur, la commande à utiliser est :

Terminal
```
~$ git clone git@192.168.42.37:monprojet
Cloning into 'monprojet'...
[...]
```

Utilisée pour la première fois, cette commande demande de confirmer qu'on se connecte sur le bon serveur : SSH ne connaît pas encore l'identité de ce serveur. En répondant **yes**, on confirme que le *fingerprint* présenté correspond bien à notre serveur. En cas de doute sur cet élément, on peut retrouver le fingerprint avec la commande suivante sur le serveur :

Terminal
```
~# ssh-keygen -lf /etc/ssh/ssh_host_ecdsa_key
256 f4:23:b6:2b:b8:87:4a:b2:7a:2f:09:e8:1b:2c:5d:c0  root@gmlf
(ECDSA)
```

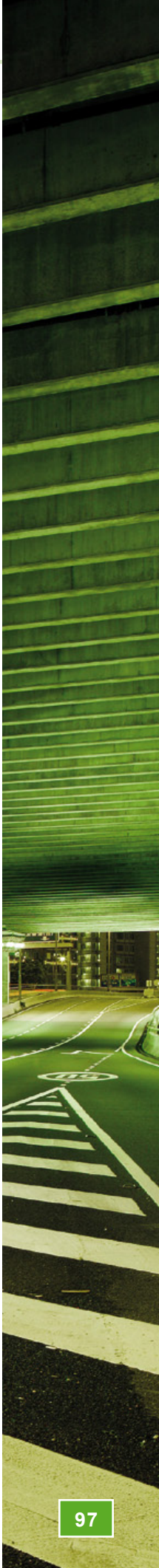
Ensuite, Git affiche une alerte indiquant que le dépôt cloné est vide. C'est bien le résultat attendu, tout va bien.



5 Création de contenu et « push » initial

Sur le client, on crée alors un fichier dans le répertoire créé ; on l'ajoute au projet et on « pousse » la modification sur le serveur :

Terminal
```
~$ cd monprojet
test@cao:~/monprojet$ echo "Hello world" > README
test@cao:~/monprojet$ git add README
test@cao:~/monprojet$ git commit -m "Initial commit"
[master (root-commit) b3feb57] Initial commit
```



Terminal

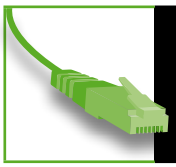
```
 1 file changed, 1 insertion(+)
 create mode 100644 README
~/monprojet$ git push origin master
Counting objects: 3, done.
Writing objects: 100% (3/3), 221 bytes, done.
Total 3 (delta 0), reused 0 (delta 0)
To git@192.168.42.37:monprojet
 * [new branch]      master -> master
```

Notons que, lors des prochaines mises à jour, il n'est plus nécessaire de donner les arguments **`origin master`** à Git. On n'utilisera plus que **`git push`**.

Avec la commande suivante, on valide que la mise à jour est bien arrivée sur le serveur :

Terminal

```
~# git --git-dir /srv/git/monprojet log
commit b3feb579ee384fd69ea3e324a2f4f120077840f8
Author: Toto Dupont <toto@monsite.fr>
Date:   Fri Jun 7 10:56:57 2013 +0200

    Initial commit
```



6 Client graphique : RabbitVCS

RabbitVCS est une surcouche à Git qui offre différents moyens d'être contrôlé : en ligne de commandes, au sein de l'éditeur Gedit, via le gestionnaire de fichiers Nautilus, Nemo ou Thunar, ou intégré au Finder d'OS X.

La distribution *Ubuntu 13.04 Raring Ringtail* inclut le client en ligne de commandes, le greffon pour Gedit et le greffon pour Nautilus. Notons cependant que le paquet pour ce dernier inclut un bogue qui n'est pas encore corrigé à la date d'écriture de cet article, bien que la source « upstream » l'ait corrigé. Nous allons donc installer le paquet central de RabbitVCS, puis récupérer le greffon directement sur le site.

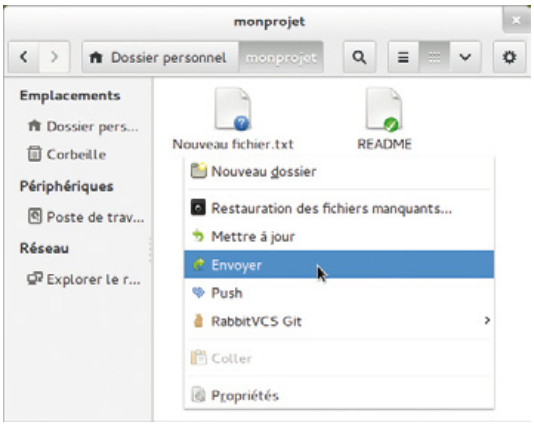
Terminal

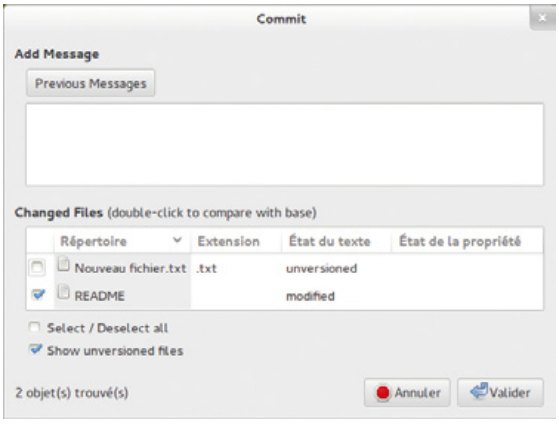
```
~$ sudo apt-get install rabbitvcs-core python-nautilus
[...]
~$ sudo wget http://rabbitvcs.googlecode.com/svn/trunk/clients/
nautilus-3.0/RabbitVCS.py -O /usr/share/nautilus-python/
extensions/RabbitVCS.py
```

Il suffit ensuite de redémarrer Nautilus (en se déconnectant, puis en se reconnectant sur sa session par exemple) pour voir des icônes sur les fichiers indiquant l'état de chacun d'entre eux et de nouvelles entrées dans le menu contextuel...

Ces nouvelles entrées liées à RabbitVCS sont :

- ***Mettre à jour*** : permet de récupérer la dernière version du dépôt du serveur (équivalent à **`git pull`**) ;



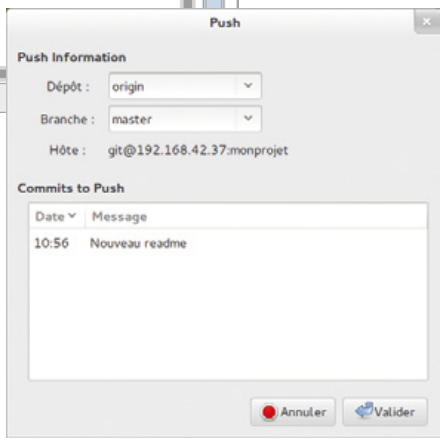


➥ ***Envoyer*** : intégrer les modifications dans le dépôt Git local (équivalent à `git commit`) ;

➥ ***Push*** : envoyer les modifications au dépôt du serveur (équivalent à `git push`).

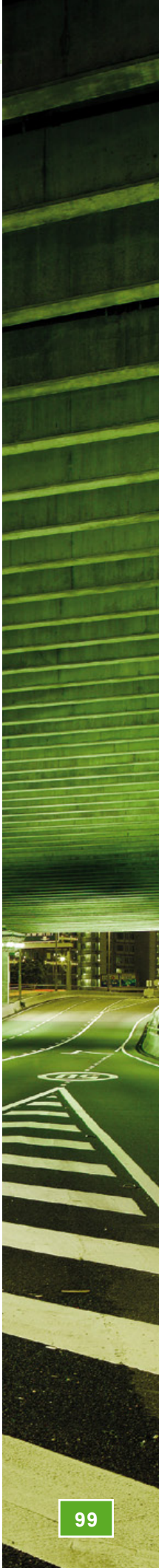
En cliquant sur ***Envoyer***, on peut choisir les fichiers à inclure dans le *commit* et indiquer le message à y associer.

Le menu contextuel des fichiers inclut notamment une entrée ***Comparer avec la base***, qui permet de voir d'une manière très pratique les différences entre le fichier modifié et celui qui est stocké dans le dépôt Git local ; pour cela, RabbitVCS utilise Meld.



Une fois la modification intégrée dans le dépôt local, c'est avec l'entrée ***Push*** du menu contextuel qu'on l'envoie sur le serveur.

Et voilà, la boucle est bouclée...
Bon développement ! ■



5

5 APPLICATIONS WEB

À découvrir dans cette partie...

5 APPLICATIONS WEB

Webmail avec Roundcube

Les interfaces web de consultation de courriers électroniques (en bref, les webmails) restent un moyen très populaire d'accéder à ses e-mails, sans avoir besoin d'installer un logiciel sur un ordinateur. Roundcube est aujourd'hui l'un des webmails open source les plus populaires.

Roundcube, dont la première version stable est sortie en 2008, utilise de nombreuses technologies récentes comme AJAX, TinyMCE... Son objectif est d'obtenir une facilité d'utilisation proche des logiciels de bureau, notamment en minimisant les temps de chargement...

Nous nous penchons ici sur l'installation par le biais du paquet Debian, bénéficiant ainsi du suivi de sécurité propre à la distribution. Notons toutefois que celui-ci est dans une version relativement ancienne (0.7.2, datant de mars 2012) ; ceux qui souhaitent profiter des dernières évolutions devront l'installer à partir des sources.

1 Environnement requis

Roundcube nécessite un serveur web (Apache ou Lighttpd par exemple), le langage PHP, une base SQL (MySQL, PostgreSQL, SQLite ou MSSQL), un serveur IMAP et un serveur SMTP.

Dans le présent tutoriel, nous allons utiliser un serveur sur lequel ont été installés :

- LAMP (Apache, MySQL, PHP) en HTTPS,
- le serveur SMTP Postfix,
- le serveur Courier-IMAP.

Il est tout à fait possible de faire fonctionner un webmail sur un lien HTTP (non chiffré), mais il est préférable d'utiliser une connexion HTTPS, car le nom d'utilisateur et le mot de passe transiteront forcément entre le client (navigateur web) et le serveur, ainsi que le contenu des e-mails.

2 Installation de Roundcube

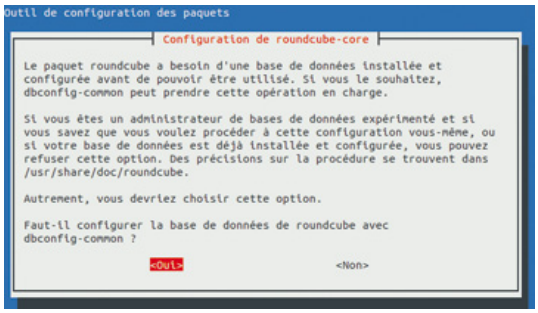
Roundcube est disponible sous forme de paquet Debian ; le paquet **roundcube-core** est à installer pour l'essentiel des fonctionnalités du logiciel et deux paquets sont disponibles pour l'interaction avec une base de données, **roundcube-mysql** (pour MySQL) et **roundcube-pgsql** (pour PostgreSQL) :

Terminal

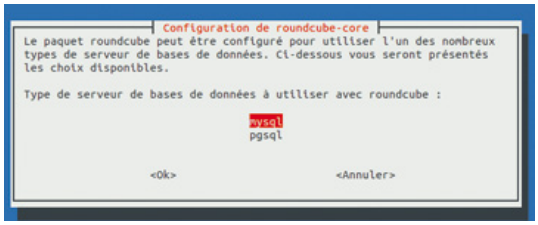
```
~# apt-get install roundcube-core
roundcube-mysql
```

3 Base de données

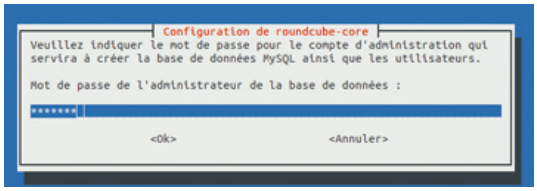
La procédure d'installation propose de configurer la base de données de Roundcube automatiquement, nous acceptons cette proposition :



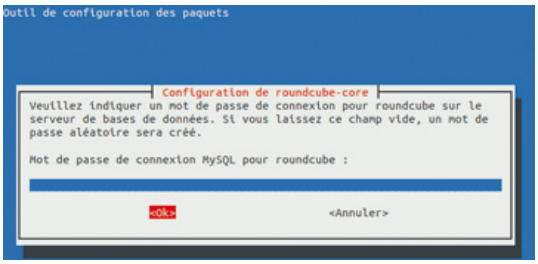
Le type de serveur de base de données est ensuite demandé, nous indiquons **`mysql`** :



La procédure a besoin du mot de passe administrateur de la base de données, c'est celui que nous avons renseigné lors de l'installation de MySQL :



Enfin, cette procédure demande le mot de passe qui sera attribué pour la connexion de Roundcube à la base SQL ; lorsque nous ne renseignons rien, un mot de passe aléatoire est créé.



4 Configuration d'Apache

Le paquet **`roundcube-core`** a mis en place une configuration pour Apache, avec une protectionoptimale. Il suffit alors de placer des alias dans l'hôte virtuel concerné, ici **`/etc/apache2/sites-available/www.monsite.fr-ssl`** :

Fichier

```
<VirtualHost _default_:443>
        ServerAdmin webmaster@monsite.fr
        DocumentRoot /srv/www.monsite.fr-ssl
        ErrorLog ${APACHE_LOG_DIR}/www.monsite.fr-ssl/error.log
        LogLevel warn
        CustomLog ${APACHE_LOG_DIR}/www.monsite.fr-ssl/access.log combined

        SSLEngine on
        SSLCertificateFile  /etc/ssl/certs/www.monsite.fr.crt
        BrowserMatch "MSIE [2-6]" \
                nokeepalive ssl-unclean-shutdown \
                downgrade-1.0 force-response-1.0
        BrowserMatch "MSIE [17-9]" ssl-unclean-shutdown

        Alias /roundcube/program/js/tiny_mce/ /usr/share/tinymce/www/
        Alias /roundcube /var/lib/roundcube
</VirtualHost>
```

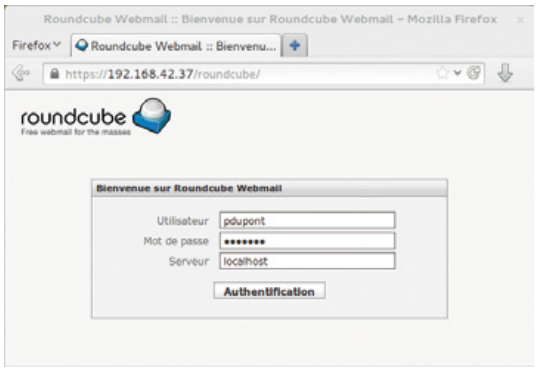
La configuration d'Apache est ensuite rechargée.

À savoir

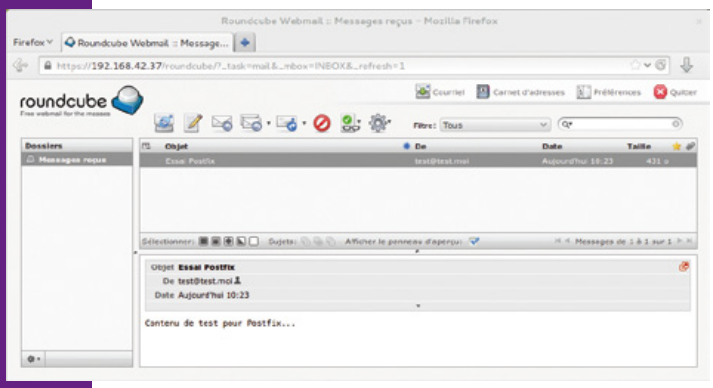
Lorsque l'on souhaite avoir un hôte virtuel dédié à Roundcube (par exemple webmail.monsite.fr), il suffit de faire pointer la directive `DocumentRoot` vers `/var/lib/roundcube`.

5 Connexion à Roundcube

Lorsque l'on accède en HTTPS au chemin **/roundcube/** sur le serveur, on se retrouve devant la mire de connexion de Roundcube.



On renseigne alors le nom de l'un des utilisateurs du système, son mot de passe, ainsi que le serveur « localhost ». Après avoir cliqué sur « Authentification », on se retrouve sur l'interface de Roundcube !



6 Configuration de Roundcube

La configuration de Roundcube s'effectue dans le fichier **main.inc.php**, qui est situé dans le répertoire **/etc/roundcube**. Nous n'allons pas détailler l'ensemble des directives disponibles dans ce fichier, elles sont bien trop nombreuses. Pour chaque directive indiquée ci-après, il s'agit de modifier la valeur existante : elles sont toutes déjà présentes dans ce fichier, aucune ligne n'est à ajouter.

7 Choix automatique du serveur IMAP

Pour que le champ « Serveur » ne soit pas affiché, il faut modifier la ligne de configuration de l'hôte dans le fichier **/etc/roundcube/main.inc.php** :

Fichier

```
$rcmail_config['default_
host'] = 'localhost';
```

Une fois cette modification effectuée, Roundcube ne demande plus le nom du serveur sur lequel se connecter.

8 Serveur SMTP

Sans serveur SMTP configuré, Roundcube utilise la fonction **mail()** de PHP pour envoyer des e-mails ; dans le fichier **/etc/roundcube/main.inc.php**, configurons le serveur SMTP à utiliser :

Fichier

```
$rcmail_config['smtp_
server'] = 'localhost';
```

9 Titre de l'interface

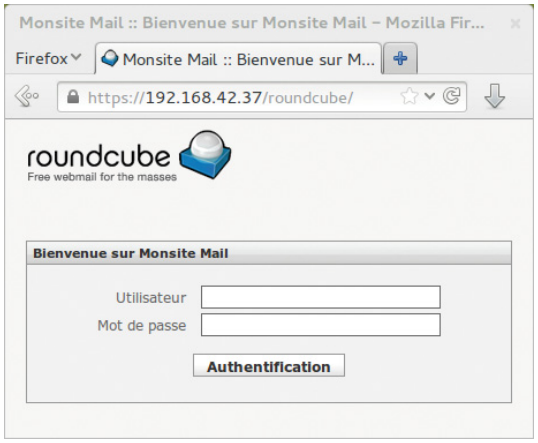
Entre autres éléments de l'interface utilisateur, il est possible de modifier le titre de l'interface par l'intermédiaire de la ligne suivante dans **main.inc.php** :

Fichier

```
$rcmail_config['product_
name'] = 'Monsite Mail';
```

10 Interface de connexion après paramétrage

Après le paramétrage des éléments ci-dessus, l'apparence de l'interface a légèrement changé :



11 Greffons pour Roundcube

De nombreux greffons sont disponibles pour Roundcube ; Debian les propose dans les paquets **roundcube-plugins** et **roundcube-plugins-extra** :

Terminal

```
~# apt-get install
roundcube-plugins
roundcube-plugins-extra
```

Les greffons sont ensuite à activer en les mentionnant dans la liste **$rcmail_config['plugins']** du fichier **main.inc.php**. La configuration des greffons eux-mêmes se fait soit dans le fichier **main.inc.php**, soit dans un fichier présent dans le répertoire **/etc/roundcube/plugins**.

De même que pour les directives de configuration de Roundcube, nous ne pouvons pas lister ici l'ensemble des greffons disponibles. Les greffons sont placés dans des répertoires **/var/lib/roundcube/plugins/<greffon>** où **<greffon>** est remplacé par le nom du greffon. Dans chacun de ces répertoires, on retrouve un fichier **<greffon>.php** qui contient souvent, en entête, une courte description ; on retrouve également parfois un fichier **README**, plus complet.

Lorsqu'un greffon nécessite une configuration particulière, un fichier **/etc/roundcube/plugins/<greffon>/config.inc.php** existe, un lien pointe vers celui-ci de **/var/lib/roundcube/plugins/<greffon>/config.inc.php** et un fichier **/var/lib/roundcube/plugins/<greffon>/config.inc.php.dist** contient les paramètres qui peuvent être appliqués dans ce fichier **config.inc.php**.

12 Greffon : newmail_notifier

Intéressons-nous par exemple au greffon **newmail_notifier**. Premièrement, activons ce greffon dans **main.inc.php** en modifiant la ligne **$rcmail_config['plugins']** :

Fichier

```
$rcmail_config['plugins'] =
array('newmail_notifier');
```

Le fichier **/var/lib/roundcube/plugins/newmail_notifier/config.inc.php.dist** indique que trois paramètres peuvent être placés dans **/etc/roundcube/plugins/newmail_notifier/config.inc.php**, **newmail_notifier_basic**, **newmail_notifier_sound** et **newmail_notifier_desktop** :

Fichier

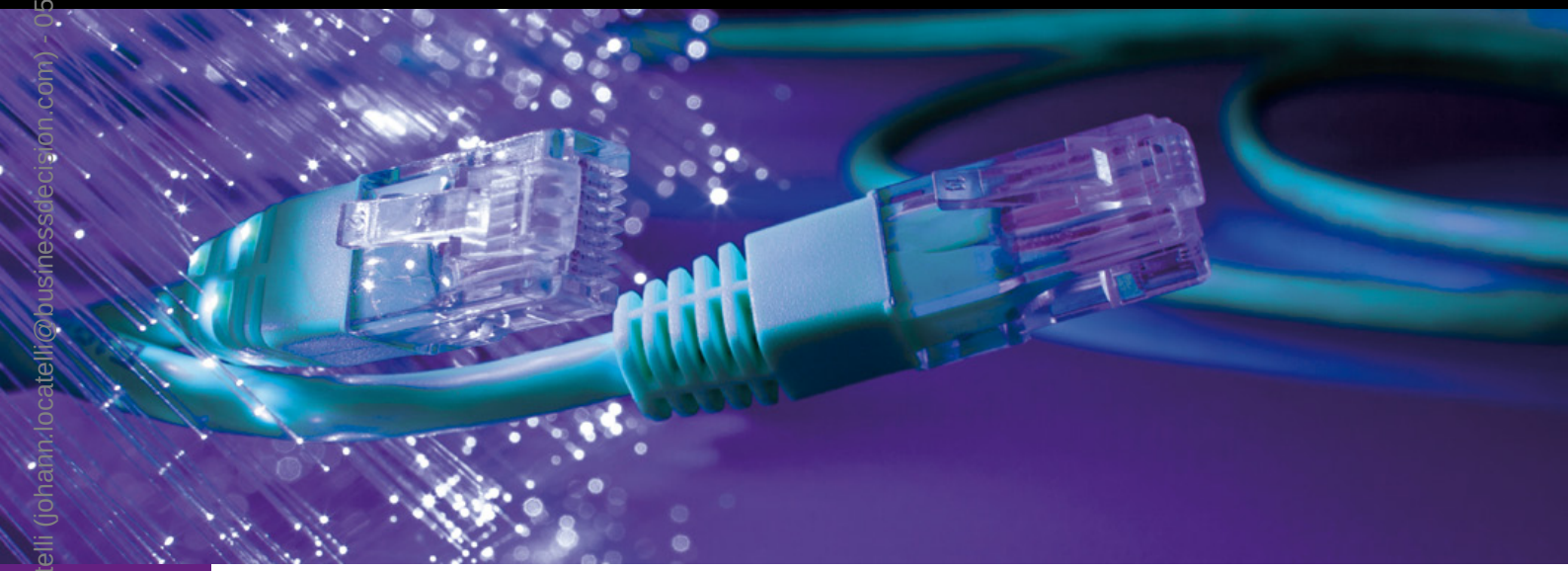
```
<?php
$rcmail_config['newmail_
notifier_basic'] = true;
$rcmail_config['newmail_
notifier_sound'] = false;
$rcmail_config['newmail_
notifier_desktop'] = false;
?>
```

Dans le cadre de ce greffon particulier, il s'agit de la configuration par défaut de ces trois paramètres, que l'utilisateur est alors capable de changer dans les paramètres de son compte Roundcube. ■

5 APPLICATIONS WEB

Faites-vous une place sur la Toile avec WordPress

Parmi les méthodes de publication sur Internet, l'une des plus populaires est le blog (pour « weblog », journal sur le Web). De nombreux logiciels existent, de même que des services hébergés, WordPress étant l'un des plus populaires de ces logiciels.



WordPress est né en 2003, en tant que fork du logiciel b2/cafelog. Il peut être utilisé pour faire fonctionner de nombreux sites sur une seule instance (c'est d'ailleurs comme ça qu'il est utilisé sur le site **wordpress.com**), mais également pour faire fonctionner un seul site.

C'est un logiciel très souple, dont les fonctionnalités peuvent être étendues avec de nombreux greffons disponibles (plus de 25000 sur le site officiel **wordpress.org**) et dont le design peut être modifié en profondeur grâce aux thèmes (plus de 1500 disponibles sur **wordpress.org**). De nombreuses sociétés ont développé des greffons et des thèmes, qui sont parfois gratuits et parfois payants. WordPress lui-même est un logiciel libre, distribué sous licence GNU GPL.

Avec WordPress, on peut bien sûr publier des pages horodatées (le principe même des blogs), mais également créer des pages classiques : de simple moteur de blog, il est aujourd'hui devenu un CMS complet.

1 Environnement requis

WordPress s'appuie sur les technologies classiques du Web : un serveur web, le langage PHP et une base de données MySQL. Pour ce tutoriel, nous allons simplement nous baser sur la configuration la plus classique : LAMP (Apache, MySQL, PHP).

On peut faire fonctionner WordPress sur un lien HTTPS pour sécuriser les accès (notamment lorsque l'on entre le mot de passe pour se connecter).

2 Installation de WordPress

Un paquet est fourni pour WordPress dans Debian. En complément du paquet « principal », nous installerons le paquet contenant les traductions de l'interface, afin d'en profiter en français :

Terminal

```
~# apt-get install
wordpress wordpress-l10n
```

La version de WordPress fournie par Debian est modifiée afin de rendre plus simple la gestion du système, avec d'un côté les fichiers de l'application (qui peuvent être concernées par des mises à jour du système) et d'un autre côté les données installées (semi-)manuellement. On peut alors retrouver :

- les fichiers de base de WordPress dans **/usr/share/wordpress** ;
- le fichier de configuration de WordPress dans **/etc/wordpress** ;
- les extensions (greffons, thèmes, ...) dans **/var/lib/wordpress**.

À savoir

Pour installer WordPress sur un hébergement mutualisé, il faut récupérer les fichiers de WordPress sur http://fr.wordpress.org/, éditer le fichier wp-config.php pour la configuration de la base de données (voir l'étape 4), placer ces fichiers sur l'espace d'hébergement, puis suivre ce tutoriel à partir de l'étape 8.

3 Emplacement du fichier de configuration

Habituellement, le fichier de configuration de WordPress est situé dans la même arborescence que le reste des fichiers et s'appelle **wp-config.php**. Dans la version proposée par Debian, ce fichier est modifié afin de rechercher la configuration de la manière suivante :

- D'abord, il cherche dans **/etc/wordpress/config-<nom d'hôte>.php** ;
- Si le fichier précédent n'est pas trouvé, il cherche dans **/etc/wordpress/config-<domaine>.php** ;
- Si le fichier précédent n'est pas trouvé, il cherche dans **/etc/wordpress/config-default.php** ;
- Si le fichier précédent n'est pas trouvé, il retourne une erreur 404 avec un message indiquant l'absence de ces fichiers.

Par exemple, pour le site **www.monsite.fr**, il va chercher dans :

- **/etc/wordpress/config-www.monsite.fr.php** ;
- **/etc/wordpress/config-monsite.fr.php** ;
- **/etc/wordpress/config-default.php**.

4 Fichier de configuration

Nous allons donc créer un fichier de configuration dans **/etc/wordpress**. Pour cet exemple, on utilisera le nom **www.monsite.fr** – chacun utilisera bien sûr le nom de son site – et nous créerons donc le fichier **/etc/wordpress/config-www.monsite.fr.php** :

Fichier

```
<?php
define('DB_NAME', 'monsite_wordpress');
define('DB_USER', 'monsite_wp');
define('DB_PASSWORD', 'super_mot_de_
passe');
define('DB_HOST', 'localhost');
define ('WPLANG', 'fr_FR');
/*
 * https://api.wordpress.org/secret-
key/1.1/salt/
 */
define('AUTH_KEY',          'G;p[...]%y?');
define('SECURE_AUTH_KEY',   '$3x[...]cx+');
define('LOGGED_IN_KEY',     'rZU[...]k?]');
define('NONCE_KEY',         'Kh+[...]kd&');
define('AUTH_SALT',         '+wB[...]V#1');
define('SECURE_AUTH_SALT', '|68[...],wT');
define('LOGGED_IN_SALT',    '|$u[...]b+-');
define('NONCE_SALT',        '9BX[...]USy');
?>
```

En début de fichier, les constantes définies sont les suivantes :

- **DB_NAME** : nom de la base MySQL ;
- **DB_USER** : utilisateur pour la connexion à MySQL ;
- **DB_PASSWORD** : mot de passe pour la connexion à MySQL ;
- **DB_HOST** : hôte auquel se connecter ;
- **WPLANG** : langue dans laquelle afficher WordPress.

Les constantes **AUTH_KEY**, **SECURE_AUTH_KEY**, **LOGGED_IN_KEY**, **NONCE_KEY**, **AUTH_SALT**, **SECURE_AUTH_SALT**, **LOGGED_IN_SALT** et **NONCE_SALT** doivent être des chaînes aléatoires uniques. Ces lignes complètes (prêtes à être copiées-collées) peuvent être récupérées sur **https://api.wordpress.org/secret-key/1.1/salt/** : cette adresse retourne un ensemble de chaînes aléatoires différent à chaque requête.

Modifications ultérieures de la configuration

S'il y a besoin de modifier cette configuration ultérieurement, en suivant les instructions d'un greffon qui a besoin de paramètres complémentaires ou pour toutes les autres directives disponibles pour WordPress par exemple, c'est ce fichier qui doit être modifié au lieu du fichier `wp-config.php` : toute référence à ce dernier dans une documentation devra donc intelligemment être remplacée.

5 Configuration d'Apache

Nous allons ensuite créer un hôte virtuel dédié à WordPress, le contenu du fichier de configuration **`/etc/apache2/sites-available/www.monsite.fr`** étant alors :

Fichier

```
<VirtualHost *:80>
   ServerName www.monsite.fr
   ServerAdmin pdupont@monsite.fr

   DocumentRoot /usr/share/wordpress
   Alias /wp-content /var/lib/
wordpress/wp-content

   ErrorLog ${APACHE_LOG_DIR}/error.log
   LogLevel warn
   CustomLog ${APACHE_LOG_DIR}/access.
log combined
</VirtualHost>
```

Nous activons ensuite ce site et nous rechargeons la configuration d'Apache :

Terminal

```
~# a2ensite www.monsite.fr
~# service apache2 reload
```

Des exemples de configuration d'Apache, pour différents cas de figure, peuvent être trouvés dans **`/usr/share/doc/wordpress/examples/apache.conf`**.

6 Base de données

Enfin, il faut créer la base de données pour WordPress :

Terminal

```
~# mysql -p
Enter password:
Welcome to the MySQL monitor.
Commands end with ; or \g.
[...]
mysql> CREATE DATABASE monsite_
wordpress CHARACTER SET utf8;
Query OK, 1 row affected (0.00 sec)
mysql> GRANT ALL ON monsite_
wordpress.* TO 'monsite_
wp'@'localhost' IDENTIFIED BY
'super_mot_de_passe';
Query OK, 0 rows affected (0.00 sec)
mysql> quit
Bye
```

7 Permettre à WordPress de modifier ses fichiers

WordPress propose la gestion des thèmes et des extensions directement dans son interface web. Si on souhaite utiliser ces fonctionnalités, il faut donner les droits à l'utilisateur d'Apache sur les répertoires concernés :

Terminal

```
~# chown -R www-data /var/
lib/wordpress/wp-content
```

Il faut également indiquer à WordPress la méthode d'accès aux fichiers, en ajoutant la ligne suivante au fichier **`/etc/wordpress/config-<nom du site>.php`** :

Fichier

```
define('FS_METHOD', 'direct');
```

Bien sûr, cela donne à WordPress un accès total à ces répertoires. Pour les paranoïaques de la sécurité qui préfèrent tout gérer manuellement, il est possible de ne pas faire ces actions et de télécharger manuellement les différents thèmes et extensions, pour les mettre en place dans les sous-répertoires de `/var/lib/wordpress/wp-content`.

8 Initialisation

Une fois WordPress, Apache et MySQL en place et configurés, il ne reste plus qu'à initialiser WordPress. Et à partir de maintenant, tout se passe dans le navigateur web ! Accédons donc à l'adresse du blog ; dans notre exemple, c'est `http://www.monsite.fr` :

Pour s'initialiser, WordPress demande les éléments suivants :

- Le titre du site (il s'affichera par défaut en entête du site et dans la barre de titre du navigateur) ;
- L'identifiant du premier utilisateur, qui aura les droits d'administrateur ;
- Le mot de passe du premier utilisateur, en double ;
- L'adresse e-mail du premier utilisateur ;
- Un choix concernant la vie privée : le site doit-il être indexé par les moteurs de recherche, ou doivent-ils être bloqués ?

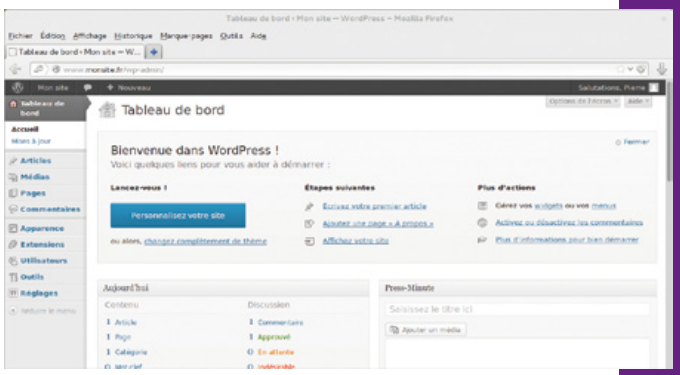
9 Fin de l'initialisation

Une fois que l'on a cliqué sur le bouton ***Installer WordPress*** à l'écran précédent, l'initialisation est terminée !

En cliquant sur ***Se connecter***, on se retrouve alors sur la page d'authentification à WordPress, où on peut renseigner son mot de passe et cliquer sur ***Se connecter***.

10 Tableau de bord

Une fois connecté, on se retrouve face au tableau de bord de WordPress. Celui-ci permet d'administrer le site ainsi que de créer de nouveaux contenus.



Cet écran comporte un menu sur la gauche, dont les différentes sections sont les suivantes :

- ***Accueil*** : c'est l'écran d'accueil, qui affiche un résumé du site et différentes informations ;
- ***Articles*** : cette section permet de gérer les différents articles à écrire (pages horodatées) ;

- ***Médias*** : ici, on peut gérer les différents médias qu'on téléverse sur le site (images, vidéos, sons, etc.) ;
- ***Pages*** : dans cette section sont gérées les pages non horodatées, utilisées par exemple pour des pages comme « À propos », « Présentation », « Contact », etc. ;
- ***Commentaires*** : tous les commentaires des visiteurs sont listés ici et peuvent être supprimés ou modifiés ;
- ***Apparence*** : cette section permet de gérer l'apparence du blog : thème, widgets, menus, etc. ;
- ***Extensions*** : cette section donne accès aux différentes extensions que l'on peut installer sur le blog afin d'étendre ses capacités ;
- ***Utilisateurs*** : on peut ici gérer les utilisateurs ayant accès au site et leur donner différents droits : administrateur, éditeur, contributeur… ;
- ***Outils*** : cette section réunit différents outils qui peuvent être utiles à l'occasion ;
- ***Réglages*** : on choisit ici les différents réglages fins du blog.

De plus, au premier accès au tableau de bord, on est accueilli par le cadre ***Bienvenue dans WordPress !***, qui propose différentes actions que l'on peut effectuer dès la création de son blog, notamment :

- Le bouton ***Personnalisez votre site*** qui vous envoie vers la page de personnalisation du thème courant ;
- Le lien ***Changez complètement de thème*** qui redirige vers l'écran de configuration des thèmes dans la section ***Apparence***.

11 Format des adresses

Par défaut, les adresses des pages sont de la forme **http://www.monsite.fr/?p=123** : ce n'est pas très « sexy ». WordPress permet de choisir une forme différente, cela se passe dans l'écran ***Réglages > Permaliens***.

Lorsqu'une notation alternative est choisie et enregistrée, WordPress essaie d'enregistrer un fichier `.htaccess` lui permettant de la mettre en œuvre. Sur un hébergement mutualisé, cela fonctionne automatiquement, car WordPress a accès à tous ses fichiers.

Avec le paquet Debian, cela ne fonctionne pas, car par mesure de sécurité, WordPress a accès à un minimum de choses. Par ailleurs, dans la mesure où on maîtrise le serveur et notamment Apache, il est préférable de modifier la configuration du site plutôt que de créer un fichier `.htaccess`.

Le fichier correspondant au VirtualHost est donc modifié (dans notre exemple, `/etc/apache2/sites-available/www.monsite.fr`) :

Fichier

```
<VirtualHost *:80>
   ServerName www.monsite.fr
   ServerAdmin pdupont@monsite.fr

   DocumentRoot /usr/share/wordpress
   Alias /wp-content /var/lib/wordpress/
wp-content

   <Directory /usr/share/wordpress>
      RewriteEngine On
      RewriteBase /
      RewriteRule ^index\.php$ - [L]
      RewriteCond %{REQUEST_FILENAME} !-f
      RewriteCond %{REQUEST_FILENAME} !-d
      RewriteRule . /index.php [L]
   </Directory>

   ErrorLog ${APACHE_LOG_DIR}/error.log
   LogLevel warn
   CustomLog ${APACHE_LOG_DIR}/access.log
combined
</VirtualHost>
```

Le module `rewrite` est ensuite activé et la configuration est rechargée :

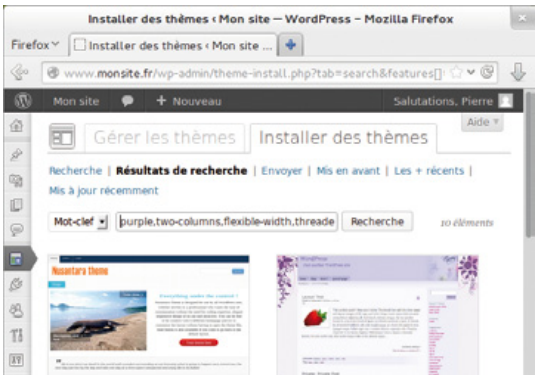
Terminal

```
~# a2enmod rewrite
~# service apache2 restart
```

12 Choix d'un thème

Dans l'écran ***Apparences > Thèmes***, on clique sur l'onglet ***Installer des thèmes*** pour télécharger de nouveaux thèmes et remplacer « Twenty Twelve », le thème par défaut très simpliste. On peut alors choisir différents critères de recherche pour trouver le thème qui nous correspond le mieux.

Une fois les critères sélectionnés et la recherche effectuée, WordPress propose différents thèmes. On peut alors cliquer sur le lien ***Aperçu*** pour voir le thème en situation, ***Détails*** pour en savoir plus ou ***Installer maintenant*** pour directement installer le thème qui nous plaît.



Une fois un thème téléchargé, on peut l'activer en cliquant sur le lien ***Activer*** de la page de confirmation du téléchargement ; ce lien se retrouve également sur la page de gestion des thèmes.

13 Personnaliser le thème

Le lien ***Personnaliser***, qui se trouve sur la page de gestion des thèmes, permet également de sélectionner les options d'apparence du thème : titre et slogan, couleurs, images, options spécifiques au thème lorsque celui-ci en propose...

14 Widgets

L'écran ***Apparence > Widgets*** permet de gérer les widgets affichés sur les différentes pages. Les widgets, ce sont ces petits cadres que l'on peut trouver sur les côtés ou en bas de page, présentant la liste des articles, la liste des commentaires, un calendrier des publications, les catégories, un nuage de mots-clés, etc.

Cet écran permet de sélectionner les widgets à afficher ou non, de les placer dans les différents espaces mis à disposition (chaque thème propose un certain nombre d'espaces dédiés aux widgets, cet écran sera alors différent selon le thème choisi). Toute cette configuration se fait simplement par glisser-déposer...

15 Anti-spam

Parmi les extensions proposées pour WordPress, on retrouve Akismet, qui est un excellent anti-spam : il deviendra rapidement nécessaire si vous autorisez les commentaires. Il fonctionne avec une base centralisée chez son éditeur, où de nombreux tests sont effectués sur chaque commentaire qui lui est soumis, afin de décider s'il s'agit d'un spam ou non. D'ailleurs, Akismet est directement inclus dans le paquet WordPress de Debian : c'est dire s'il est populaire et quasiment indispensable !

Pour utiliser Akismet, il faut d'abord obtenir une clé d'API : celle-ci est payante pour les utilisateurs professionnels, mais gratuite pour les blogs personnels. Cette clé s'obtient à l'adresse **https://akismet.com/signup/**. Une fois la clé obtenue, l'extension Akismet est activée en cliquant sur le lien ***Activer*** sous son nom, dans l'écran ***Extensions***.

Un message s'affiche alors en haut de page :

« *Akismet est presque prêt. Vous devez saisir votre clef d'API Akismet pour que cela fonctionne.* »

En cliquant sur le lien ***votre clef d'API Akismet***, l'écran de configuration d'Akismet s'affiche, comprenant le champ pour renseigner la clé. Une fois celle-ci renseignée, il ne reste plus qu'à cliquer sur ***Mettre à jour les options***.

16 À vous de jouer !

En espérant que ce tutoriel vous aura permis de mettre le pied à l'étrier, nous vous invitons à parcourir les menus de WordPress, à explorer les thèmes et extensions disponibles, à personnaliser votre blog et à publier vos articles ! ■

5 APPLICATIONS WEB

Mettre en place un site collaboratif avec MediaWiki

Le wiki est l'outil le plus populaire pour créer des sites web participatifs. Il permet à tout un chacun de modifier une page en cliquant sur un simple bouton, souvent sans même devoir s'identifier. C'est comme cela que fonctionne la très populaire encyclopédie Wikipédia. D'ailleurs, c'est pour Wikipédia que MediaWiki a été créé : ce logiciel est aujourd'hui l'un des plus utilisés de son domaine.

MediaWiki, développé à partir de 2001 par Magnus Manske, est utilisé par Wikipédia depuis 2002. Il a connu de nombreuses améliorations et offre aujourd'hui un grand nombre de fonctionnalités, développées par une équipe qui compte aujourd'hui 200 bénévoles.

Parmi les fonctionnalités de MediaWiki, on peut compter les espaces de noms (permettant de structurer le contenu), les sous-pages (offrant la possibilité d'un stockage hiérarchique), les catégories, les modèles (permettant de créer rapidement de nouvelles pages), la modification de pages par section (de cette manière, inutile de chercher la phrase que l'on veut modifier dans un long document source : on ne modifie que le paragraphe concerné), la gestion des droits d'accès et la gestion de données multimédias.

1 Installation de MediaWiki

Comme bon nombre d'applications web, MediaWiki repose sur Apache, MySQL et PHP. Il peut être installé sur un serveur web créé grâce au tutoriel LAMP.

Plusieurs paquets relatifs à MediaWiki sont disponibles dans les sources officielles de Debian. On retrouve le paquet **`mediawiki`** qui correspond au logiciel lui-même et différents paquets **`mediawiki-extensions-*`**, des extensions pour MediaWiki. Ces dernières peuvent être toutes installées grâce au méta-paquet **`mediawiki-extensions`**, qui dépend de l'ensemble d'entre elles. Installons simplement l'application :

Terminal

```
~# apt-get install mediawiki
```

2 Configuration d'Apache

Nous allons aborder ici la configuration d'un hôte virtuel pour desservir MediaWiki à sa racine. Créons un fichier spécifique à cet hôte dans la configuration d'Apache, nous prendrons comme exemple **`/etc/apache2/sites-available/doc.monsite.fr`** :

Fichier

```
<VirtualHost *:80>
   ServerName doc.monsite.fr
   ServerAdmin pdupont@monsite.fr

   DocumentRoot /var/lib/mediawiki

   ErrorLog ${APACHE_LOG_DIR}/error.log
   LogLevel warn
   CustomLog ${APACHE_LOG_DIR}/access.
log combined
</VirtualHost>
```

Cette configuration est activée avec les commandes suivantes :

Terminal

```
~# a2ensite doc.monsite.fr
~# service apache2 reload
```

Si on essaie alors d'accéder à ce site, on obtient la page de MediaWiki qui indique que la configuration ne peut être trouvée...

Si l'on souhaite desservir MediaWiki sur un sous-répertoire d'un site déjà existant plutôt que de créer un nouvel hôte virtuel, il suffit d'ajouter la ligne suivante à la configuration du site en question :

Fichier

```
Alias /mediawiki /var/lib/mediawiki
```

Le chemin **`/mediawiki`** peut bien sûr être modifié : **`/wiki`**, **`/monsuperwiki`**, etc.

3 Base de données

Comme pour toute application basée sur LAMP, il faut créer une base de données MySQL pour MediaWiki :

Terminal

```
~# mysql -p
[...]
mysql> CREATE DATABASE mon_petit_wiki
CHARACTER SET utf8;
Query OK, 1 row affected (0.00 sec)
mysql> GRANT ALL ON mon_petit_wiki.*
TO 'mw'@'localhost' IDENTIFIED BY 'mot
de passe';
Query OK, 0 rows affected (0.00 sec)
mysql> quit
```

Pour un hébergement mutualisé, cette base de données existe déjà, elle est fournie par l'hébergeur...

4 Configuration de MediaWiki

En cliquant sur le lien « *complete the installation* » de la page d'erreur vue plus haut, on peut configurer MediaWiki...

4.1 Langue

MediaWiki demande d'abord la langue dans laquelle configurer le wiki. Le paramètre ***Votre langue*** permet de définir la langue à utiliser pendant l'installation, tandis que le paramètre ***Langue du wiki*** permet de configurer la langue dans laquelle les pages seront le plus souvent écrites.

4.2 Vérifications

La page suivante vérifie les fonctionnalités présentes sur le serveur et indique les conditions d'utilisation. Grâce au système de paquets de Debian, l'environnement est correct.

4.3 Connexion à la base

L'écran qui suit demande les habituelles informations de connexion à une base de données. Les champs à renseigner sont les suivants :

- ***Type de base de données*** : le serveur de bases de données à utiliser (dans notre cas, il n'y a que le choix de MySQL parce que c'est le seul type pour lequel le support PHP est installé) ;
- ***Nom d'hôte de la base de données*** : l'hôte auquel se connecter (dans le cas d'un hébergement mutualisé, c'est le nom qui aura été fourni par l'hébergeur) ;

- ***Nom de la base de données*** : nom de la base tel qu'il a été configuré à l'étape 3, ou nom fourni par l'hébergeur ;
- ***Préfixe des tables de la base de données*** : lorsque l'on utilise une même base pour plusieurs applications (c'est souvent le cas pour un hébergement mutualisé), on peut préfixer le nom des tables ;
- ***Nom d'utilisateur de la base de données*** : nom avec lequel se connecter, tel que configuré à l'étape 3 ou fourni par l'hébergeur ;
- ***Mot de passe de la base de données*** : le mot de passe associé au nom...

4.4 Paramètres de la base de données

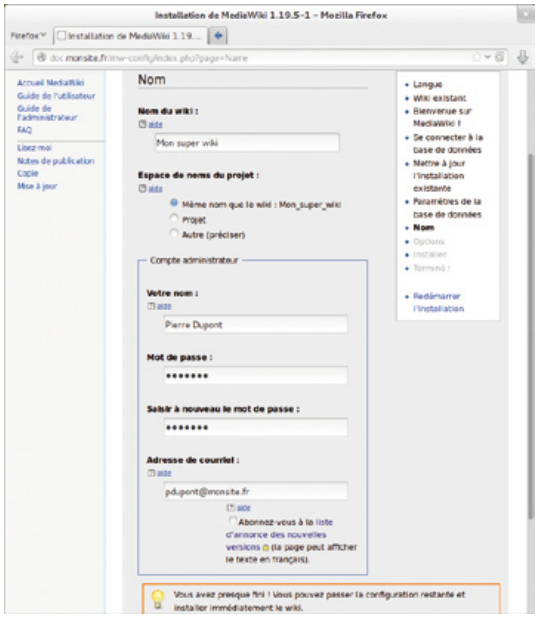
MediaWiki offre la possibilité de configurer l'accès à MySQL un peu plus finement que la plupart des autres applications web.

Cet écran demande les informations suivantes :

- ***Compte de la base de données pour l'accès Web*** : on peut configurer MediaWiki pour utiliser un compte différent que celui que l'on a configuré à l'écran précédent – ce dernier ne sera alors utilisé que pour la création des tables (cela permet de ne pas donner le droit de créer des tables à l'utilisateur de MediaWiki, par exemple) ;
- ***Moteur de stockage*** : MySQL propose deux moteurs de stockage par défaut : MyISAM qui est le moteur historique et InnoDB, plus performant ;
- ***Jeu de caractères de la base de données*** : on peut dire à MediaWiki de stocker les données directement en UTF-8 dans la base de données, mais c'est plus limité que de le configurer pour stocker les données binaires – ces données seront plus difficilement lisibles par d'autres applications, mais cela est préférable pour un usage exclusif des données par MediaWiki.

4.5 Nom

L'écran suivant demande les informations nominatives sur le wiki et sur le premier utilisateur :



- ***Nom du wiki*** : le nom du wiki, tout simplement ;
- ***Espace de noms du projet*** : par défaut, un espace de noms est créé pour le projet lui-même, l'objectif de cet espace de noms étant de contenir les pages qui parlent du wiki lui-même et non les pages qui parlent du sujet du wiki ;
- ***Votre nom*** : le nom avec lequel vous vous connecterez (oui, les espaces sont autorisés) ;
- ***Mot de passe*** : le mot de passe pour ce nom d'utilisateur ;
- ***Adresse de courrier*** : il n'est pas obligatoire de renseigner cette adresse ; elle permet notamment de réinitialiser le mot de passe si on l'a oublié et de recevoir des messages des autres utilisateurs du wiki.

Au bas de cet écran, on peut choisir si l'on veut continuer la configuration (« Me poser davantage de questions ») ou si l'on veut immédiatement utiliser le wiki ; choisissons la première solution.

5 Options

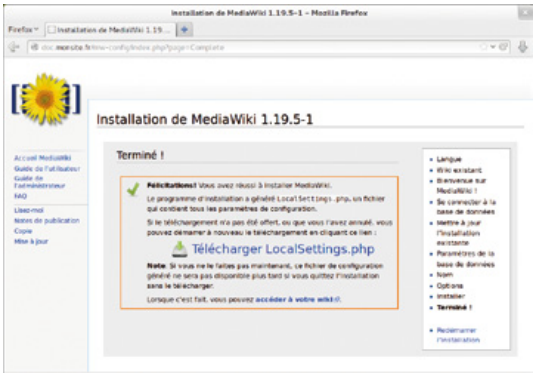
L'écran des options propose de nombreux paramétrages intéressants :

- ***Profil des droits d'utilisateurs*** : ce paramètre permet de configurer le comportement du wiki par rapport aux autorisations d'édition :
  - ***Wiki traditionnel*** : n'importe qui, même non enregistré, peut modifier les pages ;
  - ***Création de compte requise*** : n'importe qui peut créer un compte et, une fois identifié, modifier les pages ;
  - ***Éditeurs autorisés seulement*** : seules les personnes autorisées peuvent modifier les pages, n'importe qui peut les lire ;
  - ***Wiki privé*** : il faut être autorisé pour lire et modifier les pages ;
- ***Droits d'auteur et licence*** : cette option permet de définir le texte qui sera placé en bas de page. On a le choix entre les licences *Creative Commons*, la GNU FDL, le domaine public (notons toutefois qu'en France on ne peut pas mettre une œuvre dans le domaine public) et aucun texte de licence ;
- ***Paramètres de courriel*** : cette section propose différentes options de configuration des e-mails : l'activation globale ou non des e-mails, l'adresse qui est présentée comme expéditeur des e-mails, les e-mails entre utilisateurs, différentes notifications, ainsi que l'authentification par courriel (l'utilisateur doit répondre à un e-mail pour que son compte soit créé) ;
- ***Extensions*** : cette section permet de paramétrer les extensions à activer – par défaut, il y en a une vingtaine ; si on a installé `mediawiki-extensions` il y en a plus ;
- ***Téléchargement des images et des fichiers*** : cette section offre la possibilité d'activer le téléchargement des fichiers, de changer le logo du wiki et d'activer la possibilité de réutiliser facilement du contenu de *Wikimedia Commons* ;
- ***Configuration avancée*** : cette section contient le paramètre de mise en cache avec `memcached`, pour les utilisations dans une architecture de taille importante.

6 Installer

Dernière chance de modifier l'un des paramètres... La page demande confirmation de l'installation de MediaWiki. On arrive ensuite sur une étape intermédiaire, qui indique le bon avancement de l'installation.

Le dernier écran de l'installation confirme que tout s'est correctement effectué et offre au téléchargement le fichier `LocalSettings.php`.



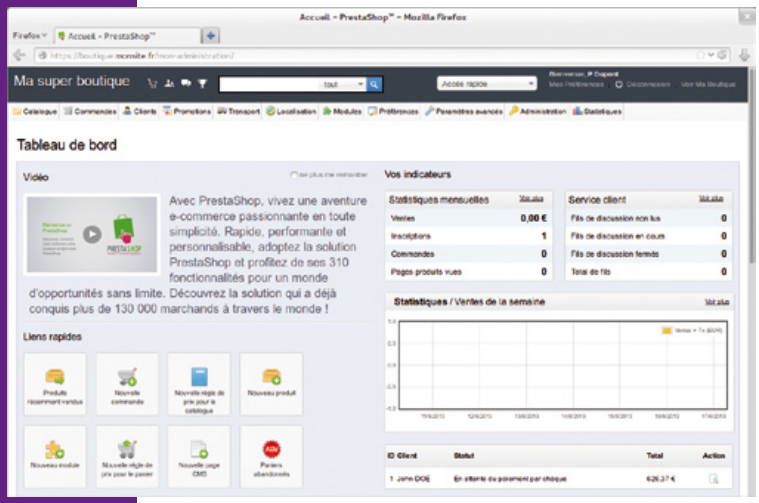
7 Accès au wiki

Et voilà ! Il suffit alors d'accéder à l'adresse du wiki pour pouvoir commencer à en modifier le contenu (en s'authentifiant, si le profil idoine a été choisi dans la page ***Options***). ■

Votre boutique en ligne avec PrestaShop

Du point de vue des professionnels du commerce, ce que la démocratisation d'Internet a vraiment changé c'est la façon de vendre. Aujourd'hui, on peut trouver presque tout sur Internet : un professionnel a tout intérêt à vendre ses produits ou ses services via Internet. Différentes plateformes de commerce électronique existent, l'une des plus populaires étant PrestaShop...

La première version de PrestaShop a été publiée en 2007, après 2 ans de développement qui ont débuté au sein de l'école informatique Epitech près de Paris. Le projet s'appelait à l'origine phpOpenStore. Ce logiciel est maintenu par la société du même nom, ainsi que par de nombreux contributeurs bénévoles.



La boutique par défaut de PrestaShop, avant personnalisation

PrestaShop est particulièrement souple et modulaire, capable de s'adapter à de nombreuses situations différentes. On peut changer autant les fonctions proposées aux acheteurs que la présentation du site de vente en ligne. Ce logiciel a également comme avantage de s'adapter aux spécificités locales (monnaies, taxes, lois...) et d'être traduit dans plus de 50 langues.

On l'aura compris, PrestaShop est une solution tout à fait valable lorsque l'on veut créer une boutique de vente en ligne ne se basant que sur du logiciel libre.

1 Environnement requis

PrestaShop s'appuie sur le langage PHP et le serveur de bases de données MySQL. Il faut également bien sûr un serveur web, Apache par exemple. Le tutoriel LAMP est donc tout à fait indiqué pour préparer l'environnement accueillant cette application.

Toute boutique en ligne sérieuse est desservie via le protocole HTTPS : le chiffrement SSL dans une transaction commerciale, c'est un gage de sérieux. Techniquement, cela n'est pas obligatoire, mais attendez-vous à ce que vos acheteurs fassent la moue s'ils ne voient pas le fameux cadenas affiché par le navigateur web !

PrestaShop n'est pas disponible en tant que paquet Debian, nous allons donc l'installer d'une manière classique. Cela veut également dire que la procédure sera quasiment la même sur un hébergement mutualisé.

2 Téléchargement

Pour télécharger PrestaShop, il suffit de se rendre sur le site de l'éditeur : **http://www.prestashop.com/fr/telechargement**. Cette page demande votre adresse e-mail et votre profil, mais le téléchargement fonctionne tout à fait si l'on ne renseigne pas ces informations avant de cliquer sur le bouton vert « Téléchargez PrestaShop ». Une fois le téléchargement terminé, vous serez en possession de l'archive de la version la plus récente (version 1.5.4.1 lors de l'écriture de ces lignes).

3 Décompression

Pour un hébergement mutualisé, décompressez le fichier `.zip` ainsi téléchargé, puis transférez le répertoire `prestashop` obtenu vers votre espace d'hébergement – ce répertoire peut être renommé pour que l'adresse soit plus compréhensible pour vos visiteurs, par exemple en `boutique`.
Vous pouvez également transférer le contenu de ce répertoire à la racine de votre hébergement.

Pour un serveur dédié, on transfère plutôt l'archive, que l'on décompresse sur le serveur ; dans le cadre de ce tutoriel, on a transféré le fichier `.zip` dans `/root` et l'application sera installée dans `/srv/prestashop` :

Terminal
```
/srv# unzip ~/prestashop_1.5.4.1.zip
[...]
~# rm Install_PrestaShop.html
```

4 Certificat SSL

Pour une boutique en ligne, il est indispensable d'utiliser un certificat SSL signé par une « vraie » autorité de certification, pour que vos visiteurs ne soient pas confrontés à un message d'erreur lorsqu'ils se connectent à votre site. Vous pouvez acheter un tel certificat auprès d'une autorité de certification (les tarifs vont d'une dizaine d'euros à quelques dizaines de milliers d'euros par an, selon les garanties offertes) : cherchez « certificat SSL » avec votre moteur de recherche préféré.

5 Configuration d'Apache

Pour un hébergement mutualisé, vous n'avez pas la main sur la configuration d'Apache. Si votre hébergeur vous donne la possibilité de définir le certificat SSL à utiliser avec votre site en HTTPS, alors fournissez celui que vous aurez acheté auprès de votre autorité de certification.

Pour un serveur dédié, il faut configurer Apache afin d'utiliser ce certificat SSL. Dans ce tutoriel, nous allons configurer Apache pour qu'il puisse desservir plusieurs hôtes virtuels en SSL (grâce à SNI) et la boutique sera à l'adresse `boutique.monsite.fr` ; de plus, en HTTP, l'adresse de la boutique renverra automatiquement vers le protocole HTTPS – comme pour tout tutoriel, chacun adaptera à son propre cas.

Tout d'abord, la ligne suivante est ajoutée au fichier `/etc/apache2/ports.conf` (si elle n'y était pas déjà) :

Fichier
```
NameVirtualHost *:443
```

Ensuite, le fichier **/etc/apache2/sites-available/boutique** est créé :

Fichier

```
<VirtualHost *:443>
   ServerName boutique.monsite.fr

   ServerAdmin webmaster@monsite.fr
   DocumentRoot /srv/prestashop
   ErrorLog ${APACHE_LOG_DIR}/
boutique/error.log
   LogLevel warn
   CustomLog ${APACHE_LOG_DIR}/
boutique/access.log combined

   SSLEngine on
   SSLCertificateFile    /etc/ssl/
certs/boutique.monsite.fr.crt
   SSLCertificateKeyFile /etc/ssl/
private/boutique.monsite.fr.key
</VirtualHost>
<VirtualHost *:80>
   ServerName boutique.monsite.fr

   ServerAdmin webmaster@monsite.fr
   DocumentRoot /var/www
   ErrorLog ${APACHE_LOG_DIR}/
boutique/error.log
   LogLevel warn
   CustomLog ${APACHE_LOG_DIR}/
boutique/access.log combined

   RewriteEngine On
   RewriteRule ^/(.*)
https://%{SERVER_NAME}/$1 [R,L]

</VirtualHost>
```

Il ne faut pas oublier de modifier la configuration d'un éventuel site HTTPS qui aurait été configuré sans hôte virtuel ou le désactiver, comme dans le fichier **default-ssl** par exemple (la définition d'un tel site commence par **<VirtualHost _default_:443>**).

L'étape suivante consiste à activer cette configuration :

Terminal

```
~# mkdir /var/log/apache2/boutique
~# a2ensite boutique
Enabling site boutique
To activate the new configuration,
you need to run:
  service apache2 reload
~# a2enmod rewrite
Enabling module rewrite.
To activate the new configuration,
you need to run:
  service apache2 restart
~# service apache2 restart
```

Terminal

```
[....] Reloading web server config:
apache2apache2: Could not reliably
determine the server's fully
qualified domain name, using
127.0.1.1 for ServerName
. ok
```

6 Base de données

Avec un hébergement mutualisé, PrestaShop s'appuiera sur la base de données fournie par l'hébergeur : généralement, vous êtres déjà en possession des informations liées à cette base.

Dans le cas d'un serveur dédié, vous pouvez créer une base de données dédiée à PrestaShop :

Terminal

```
~# mysql -p
[...]
mysql> CREATE DATABASE boutique
CHARACTER SET utf8;
Query OK, 1 row affected (0.00 sec)
mysql> GRANT ALL ON boutique.* TO
'prestashop'@'localhost' IDENTIFIED BY
'mot de passe';
Query OK, 0 rows affected (0.00 sec)
mysql> quit
```

7 Modules PHP

PrestaShop nécessite des modules PHP qui ne sont pas installés par défaut avec le paquet Debian : **GD** et **Mcrypt**. La commande suivante permet de les installer :

Terminal

```
~# apt-get install php5-gd php5-mcrypt
```

Sans ces modules, une alerte sera retournée lors de la configuration de PrestaShop, à l'étape « Compatibilité système ».

8 Droits sur les fichiers

PrestaShop a besoin d'avoir les droits de modification sur certains fichiers de son arborescence. La commande suivante permettra de les lui donner :

Terminal

```
~# cd /srv/prestashop
~# chown -R www-data config cache
log img mails modules themes/default/
lang themes/default/cache translations
upload download sitemap.xml
```

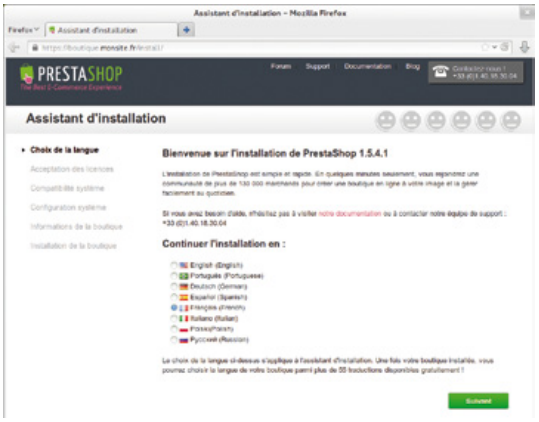
Si ces droits ne lui sont pas donnés, une alerte sera retournée lors de sa configuration, à l'étape « Compatibilité système ».

9 Configuration de PrestaShop

Une fois le système prêt, on peut passer à la configuration de l'outil lui-même. Cette configuration se fait par l'interface web directement, comme la majorité des applications web. Accédons donc à **https://boutique.monsite.fr** : on arrive alors sur l'assistant d'installation.

10 Choix de la langue

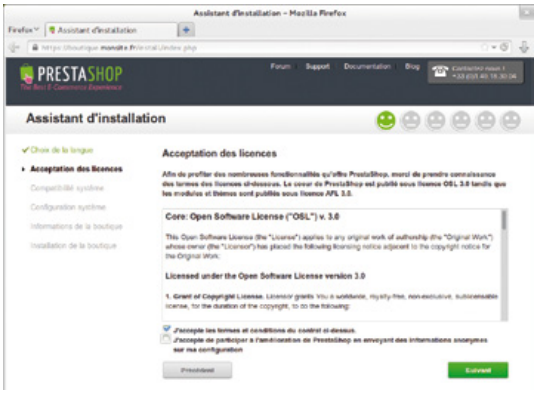
La première étape est de choisir la langue de la procédure d'installation. Cela ne conditionne pas la langue de la boutique une fois activée. On choisit le français, puis on clique sur « Suivant ».



11 Acceptation des licences

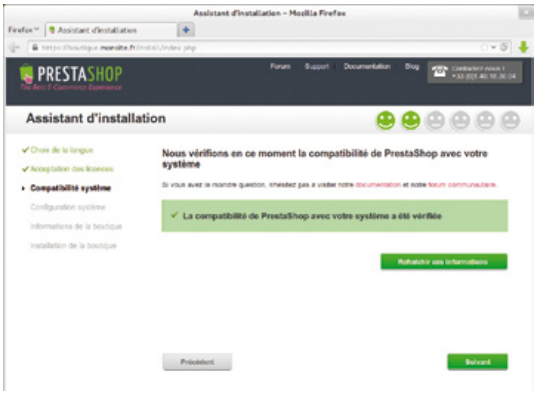
L'étape suivante demande d'accepter les licences : *Open Software License* pour le cœur de PrestaShop et *Academic Free License* pour les modules et les thèmes. Des cases à cocher sont présentes :

- La première permet d'accepter les licences (obligatoire) ;
- La seconde permet de participer anonymement à l'amélioration de PrestaShop en envoyant des informations sur la configuration à l'éditeur (chacun choisira s'il souhaite contribuer de cette manière).



12 Compatibilité système

PrestaShop teste ensuite la conformité du système avec ses besoins. Si tout va bien, il indique que la compatibilité a été vérifiée. S'il y a des problèmes, il demande d'abord de les résoudre.



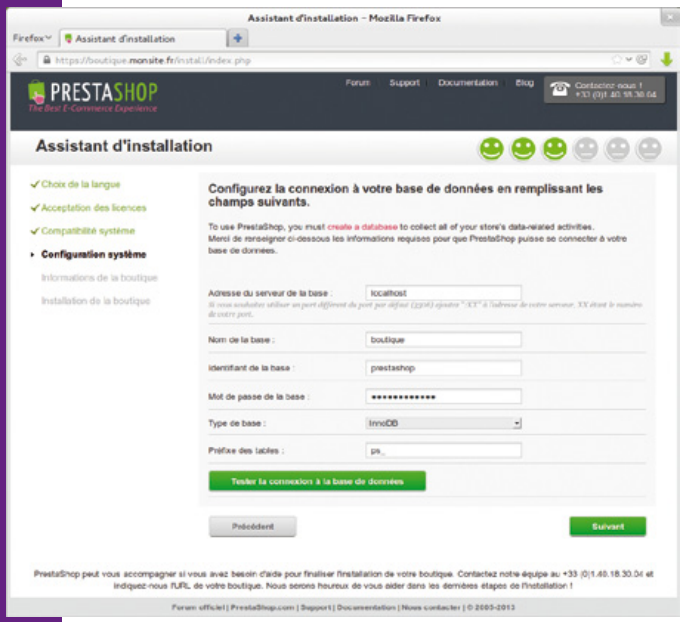
13 Configuration système

L'écran suivant demande les informations d'accès à la base de données :

- l'adresse du serveur de la base de données ;
- le nom de la base à laquelle se connecter ;

- le nom d'utilisateur avec lequel se connecter à la base ;
- le mot de passe à utiliser pour se connecter à la base ;
- le moteur des tables de la base de données (MyISAM pour le moteur historique, InnoDB pour un moteur plus performant) ;
- le préfixe des tables (utile si plusieurs applications se servent de la même base).

Le bouton « Tester la connexion à la base de données » permet de confirmer la validité de ces informations.



14 Informations de la boutique

Ensuite, l'assistant d'installation demande quelques informations de base :

- ***Nom de votre boutique*** : le nom de la boutique à présenter aux visiteurs ;
- ***Activité principale*** : selon l'activité choisie, PrestaShop proposera les fonctionnalités adaptées ;
- ***Pays par défaut*** : le pays principal auquel la boutique sera dédiée ;
- ***Fuseau horaire*** : le fuseau horaire dans lequel doit être située la boutique ;
- ***Logo de la boutique*** (facultatif) : le logo de la boutique ;
- ***Prénom*** : prénom du premier utilisateur (super-administrateur) ;
- ***Nom*** : nom du premier utilisateur (super-administrateur) ;
- ***Adresse e-mail*** : adresse du premier utilisateur (utilisée comme identifiant pour se connecter à l'interface de gestion) ;
- ***Mot de passe*** : mot de passe du premier utilisateur ;
- ***S'inscrire à la newsletter de PrestaShop*** : cette option permet au premier utilisateur de recevoir les conseils publiés régulièrement par l'éditeur.

15 Installation de la boutique

Une fois tous les renseignements donnés, la dernière étape indique l'avancement de l'initialisation de la boutique, dans le détail. Une fois cette initialisation terminée, l'assistant rappelle l'identifiant (l'adresse e-mail de l'utilisateur), ainsi que le mot de passe (caché par des astérisques, il peut être révélé si l'on ne s'en rappelle pas). Deux liens sont également présentés :

- ***Administration*** (bouton « Gérez votre boutique ») permet d'accéder à la gestion de la boutique ;
- ***Front Office*** (bouton « Découvrez votre boutique ») permet de voir la boutique telle qu'elle sera vue par vos visiteurs.

Notez toutefois que ce second bouton vous renvoie vers le port HTTP (non sécurisé) : si vous n'avez configuré Apache pour ce site qu'en HTTPS, vous obtiendrez alors une erreur 404.

16 Suppression du répertoire d'installation

Par mesure de sécurité, comme proposé par l'assistant d'installation, nous pouvons maintenant supprimer le répertoire `install` de PrestaShop.

Sur un hébergement mutualisé, ce répertoire sera supprimé par l'intermédiaire d'un client FTP.
Sur un serveur dédié, la commande suivante peut être utilisée :

Terminal
```
~# rm -fr /srv/prestashop/install
```

17 Changement du nom du répertoire « admin »

Par mesure de sécurité (encore une fois, même si cette fois-ci c'est plus de la sécurité par l'obscurité), PrestaShop refuse de donner accès à l'interface de gestion par l'adresse `/admin`.
Il faut alors renommer ce répertoire.

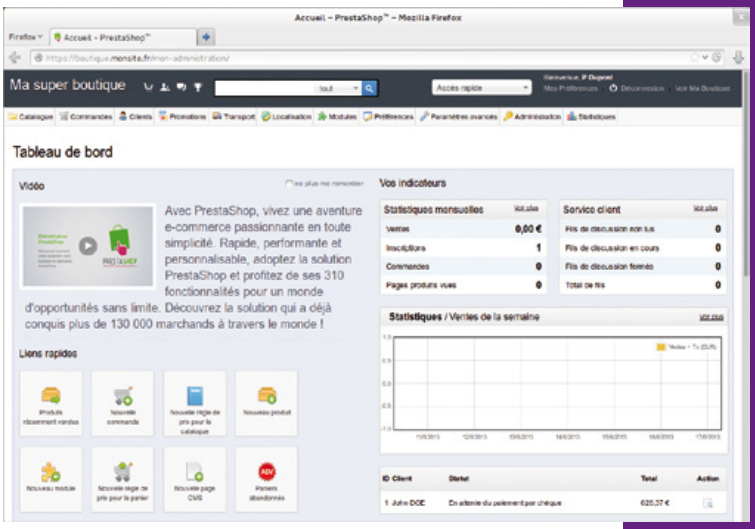
Sur un hébergement mutualisé, ce répertoire sera renommé par l'intermédiaire d'un client FTP. Sur un serveur dédié, la commande suivante peut être utilisée :

Terminal
```
~# mv /srv/prestashop/admin /srv/
prestashop/mon-administration
```

On accède alors à l'interface d'administration par le chemin ainsi modifié, ici `/mon-administration`.

18 Interface d'administration

Une fois connecté à PrestaShop, on se retrouve face à l'interface d'administration ; celle-ci permet de gérer l'intégralité de la boutique. Le premier écran rencontré est le tableau de bord, qui indique les statistiques de la boutique, offre des liens rapides, de la documentation, etc.



Cette interface d'administration propose notamment un menu horizontal, dont les sections sont les suivantes :

- ***Catalogue*** : cette section permet de gérer les produits et services à la vente, ainsi que leur catégorisation, leurs caractéristiques, etc. ;
- ***Commandes*** : ici, on a accès à l'intégralité des commandes des clients, leur état, les factures, les avoirs émis, etc. ;
- ***Clients*** : dans cette section, on gère les clients inscrits sur la boutique ;
- ***Promotions*** : ce menu donne accès aux fonctionnalités de codes promotionnels, réductions en cas d'achat en grande quantité et aux outils de marketing ;
- ***Transport*** : cette entrée permet de gérer les différents moyens de livraison de la boutique ;
- ***Localisation*** : cette section concerne tous les paramètres relatifs aux données locales des pays (taxes, etc.), ainsi qu'à la langue dans laquelle la boutique est proposée ;

- ***Modules*** : de nombreux modules sont disponibles pour PrestaShop, ajoutant des fonctionnalités diverses – cette section propose également le paramétrage des modules de paiement ;
- ***Préférences*** : les paramètres essentiels de la boutique sont présents dans ce menu ;
- ***Paramètres avancés*** : cette section contient différents outils avancés, pas nécessairement utiles dans l'utilisation et l'administration de la boutique au quotidien, mais plutôt dédiés à des actions ponctuelles comme des mises à jour, migrations, etc. ;
- ***Administration*** : cette section permet de gérer les données relatives aux accès à la boutique : liste des employés, menus activés ou cachés, etc. ;
- ***Statistiques*** : cette entrée donne accès à de nombreuses statistiques sur la boutique : nombre de visites, commandes, informations clients, produits les plus achetés et de nombreuses autres.

À titre d'exemple, après installation, PrestaShop contient quelques produits, ainsi qu'un client qui a passé une commande, ces données peuvent bien sûr être supprimées.

19 Réécriture d'URL

Par défaut, les adresses des différents éléments ne sont pas très « jolies » ; on peut par exemple avoir une adresse **https://boutique.monsite.fr/index.php?id_product=2&controller=product**. Pour que ces adresses soient plus agréables à lire, il faut autoriser la réécriture d'URL. Pour cela, il faut autoriser l'écriture dans le fichier `.htaccess`.

Sur un hébergement mutualisé, cela devrait fonctionner tout seul. Avec un serveur dédié, on donne les droits précisément sur ce fichier :

Terminal
```
~# touch /srv/prestashop/.htaccess
~# chown www-data /srv/
prestashop/.htaccess
```

Sur un serveur dédié, il faut également activer le module `rewrite` ; cela a été fait à l'étape 5 de ce tutoriel pour la redirection de HTTP vers HTTPS, mais si vous avez sauté cette partie il faut alors tout de même exécuter les commandes suivantes :

Terminal
```
~# a2enmod rewrite
~# service apache2 restart
```

Une fois ces opérations effectuées, on va dans le menu ***Préférences > SEO & URLs***, puis dans la section ***Configuration des URLs***. Dans cette section, on coche la case ***Oui*** de la ligne ***URL simplifiée***. On clique ensuite sur le bouton ***Enregistrer***.

20 Adresses un peu plus « jolies »

Par défaut, avec la réécriture d'URL, PrestaShop génère des adresses terminant par `.html` pour les produits. Cette extension n'est pas obligatoire. Pour l'enlever, toujours dans la page ***Préférences > SEO & URLs***, on descend dans la section ***Options*** pour modifier le paramètre ***Route vers les produits***, en y mettant : `{category:/}{id}-{rewrite}{-:ean13}`.

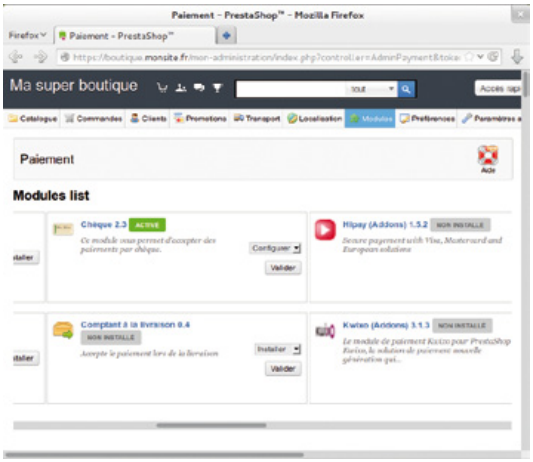
L'adresse du même produit est alors **https://boutique.monsite.fr/musique-ipods/2-ipod-shuffle**. Chacun pourra modifier ce paramètre, ainsi que les autres routes présentées, afin de correspondre à ses besoins : on peut par exemple se passer du nom de la catégorie...

21 Modules de paiement

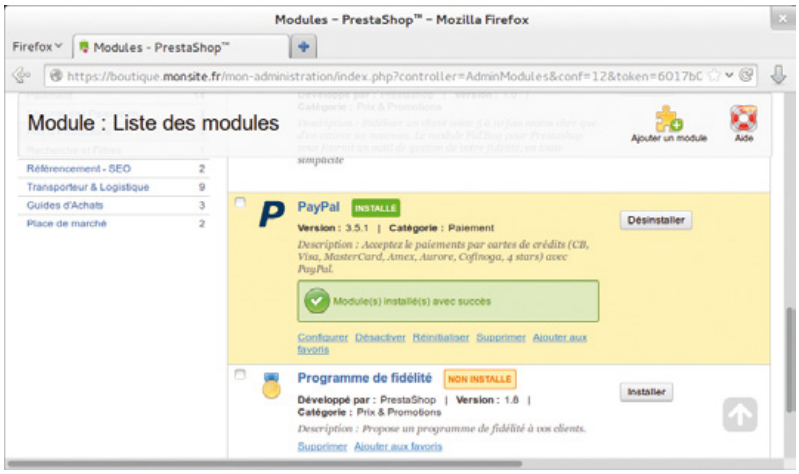
Une étape indispensable lors de la mise en place d'une boutique est la configuration des modules de paiement. Par défaut, les modules ***Chèque et Virement bancaire*** sont activés, mais ils ne sont pas configurés. Dans l'écran ***Modules > Paiement***, on retrouve tous les modules disponibles, activés ou non.

Pour les modules ***Chèque et Virement bancaire***, on choisit ***Configurer*** dans la liste déroulante, puis on clique sur ***Valider***. On peut alors configurer les modules :

- ***ordre du chèque*** et ***adresse*** pour les paiements par chèque ;
- ***titulaire***, ***détails*** (RIB, BIC, IBAN, etc.) et ***adresse de la banque*** pour les paiements par virement bancaire.



On peut également activer d'autres modules, en cliquant sur le bouton ***Installer*** du cadre de chacun d'entre eux. Une fois ce bouton cliqué, la page d'installation de modules est affichée, indiquant que le module est installé avec succès. On peut alors cliquer sur le lien ***Configurer*** pour paramétrer le module installé (ici, Paypal) ;



l'écran affiché est le même que lorsque l'on utilise ***Configurer***, puis ***Valider*** sur la liste spécifique des modules de paiement, comme décrit ci-dessus.

22 Transporteurs

Dans l'écran ***Transport > Transporteurs***, on peut cliquer sur le bouton ***Créer*** afin de créer un nouveau profil de transporteur. On peut également utiliser le bouton ***Liste des modules*** pour installer des modules spécifiques à des transporteurs : Kiala, Colissimo, TNT, etc. Ces modules permettent de simplifier la gestion des transporteurs.

Dans l'écran ***Transport > Transport***, on configure les frais supplémentaires, ainsi que le coût du transport selon les zones d'expédition et les transporteurs proposés. On peut également définir des tranches de frais de port, basées sur le prix de la commande ou sur le poids total, grâce aux écrans ***Transport > Tranches de prix*** et ***Transport > Tranches de poids***.

23 Pour aller plus loin...

Et voilà ! Il ne vous reste plus qu'à créer vos produits dans le menu ***Catalogue*** et à explorer l'intégralité de PrestaShop, qui est un produit très complet ! ■

Votre galerie de photos avec Piwigo

S'il y a un usage que le numérique a révolutionné en profondeur, c'est bien la photographie : aujourd'hui, n'importe qui peut prendre des photos, que ce soit avec un téléphone à 100€ ou avec un appareil Reflex à 2000€ ; et aujourd'hui, chacun veut partager ses photos avec ses amis, sa famille, le monde entier... Parmi les logiciels de publication de galeries photos sur Internet, Piwigo est l'un des plus populaires...

Piwigo est né sous le nom PhpWebGallery en 2001, créé par Pierrick Le Gall ; c'est en 2009 que le nom a changé, à l'occasion de la sortie de la version 2, mais le logiciel est resté le même. Avec Piwigo, vous pouvez évidemment publier vos photographies ; vous pouvez organiser les photos dans des albums, les identifier avec des étiquettes (tags), naviguer par date, donner des permissions différentes à des utilisateurs différents, personnaliser le look de votre site grâce à des thèmes...

Piwigo existait en tant que paquet sur la version stable précédente de Debian (Squeeze), mais il n'existe plus dans la version actuelle (Wheezy). Nous allons donc l'installer à partir de ses sources, « à l'ancienne ».

Ce tutoriel peut être suivi pour une installation sur un serveur dédié comme pour une installation sur un hébergement mutualisé : les deux approches seront détaillées.

1 Environnement requis

Piwigo utilise les grands classiques : Apache, MySQL, PHP. Nous pouvons alors nous baser sur un serveur LAMP, tel qu'installé dans un précédent tutoriel...

2 Téléchargement de Piwigo

Piwigo est simplement téléchargé depuis son site officiel : **http://fr.piwigo.org/basics/downloads**. Nous allons utiliser la version « package ». Pour l'installation sur un hébergement mutualisé, on téléchargera Piwigo en cliquant sur le bouton « Télécharger le Package ». Pour l'installation sur un serveur dédié, on y téléchargera l'archive en ligne de commandes :

Terminal

```
~# wget http://piwigo.org/download/
dlcounter.php?code=latest -O
piwigo.zip
```

3 Décompresser et mettre en place

Dans le cas d'un hébergement mutualisé, on peut décompresser l'archive téléchargée sur son ordinateur, puis transférer le répertoire ainsi créé sur l'espace fourni par l'hébergeur, généralement

avec un client FTP. Il est préférable de renommer ce répertoire en **piwigo** tout simplement (inutile de montrer le numéro de version aux visiteurs...). On peut également placer le contenu du répertoire à la racine de l'hébergement si on veut que Piwigo soit à la racine du site.

Pour un serveur dédié, il faut d'abord installer la commande **unzip** si elle n'est pas déjà installée, puis décompresser le fichier téléchargé ; dans notre exemple, nous allons placer Piwigo dans le répertoire **/srv** :

Terminal

```
~# cd /srv
/srv# unzip ~/piwigo.zip
```

4 Configuration d'Apache

Bien évidemment, il n'y a aucune configuration du serveur web à effectuer pour un hébergement mutualisé. Cette étape ne s'adresse alors qu'à ceux qui installent Piwigo sur leur serveur dédié, en auto-hébergement ou en « housing ».

Intéressons-nous à la création d'un hôte virtuel dédié à Piwigo ; nous allons créer le fichier **/etc/apache2/sites-available/photos.monsite.fr** :

Fichier

```
<VirtualHost *:80>
    ServerName photos.monsite.fr
    ServerAdmin pdupont@monsite.fr

    DocumentRoot /srv/piwigo

    ErrorLog ${APACHE_LOG_DIR}/error.
log
    LogLevel warn
    CustomLog ${APACHE_LOG_DIR}/
access.log combined
</VirtualHost>
```

On active ensuite ce site et on recharge la configuration d'Apache :

Terminal

```
~# a2ensite photos.monsite.fr
~# service apache2 reload
```

5 Base de données

Dans le cas d'un hébergement mutualisé, la base de données est fournie par l'hébergeur et on devra utiliser les informations de connexion qu'il fournit.

Dans le cas d'un serveur dédié, il faut créer une base et un utilisateur pour MySQL :

Terminal

```
~# mysql -p
Enter password:
Welcome to the MySQL monitor.
Commands end with ; or \g.
[...]

mysql> CREATE DATABASE mes_photos
CHARACTER SET utf8;
Query OK, 1 row affected (0.00 sec)

mysql> GRANT ALL ON mes_photos.* TO
'piwigo'@'localhost' IDENTIFIED BY 'mot
de passe';
Query OK, 0 rows affected (0.00 sec)

mysql> quit
Bye
```

6 Paramétrage de Piwigo

La suite de l'installation se fait en ligne, à l'adresse à laquelle Piwigo est accessible. Dans notre exemple, c'est **http://photos.monsite.fr/**, chacun l'adaptera à sa situation...

Cet écran demande différentes informations :

- La langue par défaut (au cas où Piwigo n'arrive pas à détecter la langue du navigateur) ;
- Les paramètres de connexion à la base de données (hôte, utilisateur, mot de passe, nom de la base) ;
- Le préfixe des noms de tables, ce qui permet de mettre plusieurs applications dans la même base, généralement le cas sur un hébergement mutualisé ;
- L'identifiant, le mot de passe et l'e-mail du premier utilisateur, qui aura le statut d'administrateur.

Une fois ces renseignements saisis, on clique sur le bouton ***Démarrer l'installation***.

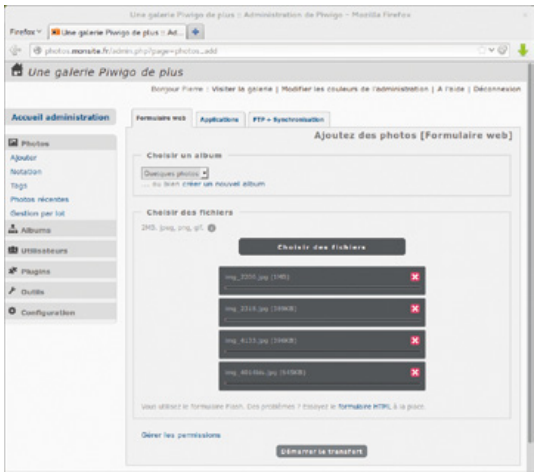
7 Fin du paramétrage

Et voilà, Piwigo n'a pas besoin de plus de renseignements, le paramétrage initial est terminé ! Il ne reste qu'à cliquer sur ***Visiter la galerie*** pour accéder à Piwigo.

8 Ajout initial de photographies

C'est la première fois que l'on accède à Piwigo, nous sommes donc accueillis par un écran qui propose de mettre en ligne des photos immédiatement. On clique alors sur le bouton ***Je veux ajouter des photos*** pour accéder à l'écran de téléversement de photographies de l'interface d'administration.

On choisit en premier l'album dans lequel on veut placer les photographies ; on peut également en créer un, ce qui est obligatoire lorsque l'on publie des photographies pour la première fois. En cliquant sur ***Choisir des fichiers***, on peut alors sélectionner lesquels on veut publier : une fois les photographies choisies, on clique sur ***Démarrer le transfert***.



9 Interface d'administration

L'interface d'administration que l'on a sous les yeux après installation, ou à laquelle on accède via le lien ***Administration*** des pages de la galerie, offre les sections suivantes :

- ***Photos*** : cette section permet de gérer les photographies (ajout, suppression, notes, tags...) ;
- ***Albums*** : cette section permet de gérer les albums dans lesquels sont réparties les photographies ;
- ***Utilisateurs*** : cette section permet de gérer les utilisateurs ayant accès à Piwigo, que ce soit pour gérer les photos ou pour accéder à des photos privées ;
- ***Plugins*** : cette section permet de gérer les greffons, ceux-ci permettant d'ajouter des fonctionnalités à Piwigo ;
- ***Outils*** : cette section regroupe de nombreux outils de gestion de la base de photographies ;
- ***Configuration*** : cette section permet de configurer Piwigo, notamment son aspect visuel (nom de la galerie, taille des photos, filigrane, etc.). ■

Introduction

Rappels et notions de base concernant le réseau et la sécurité

HTTP/web

Apache, Lighttpd, Nginx, trois solutions pour installer votre serveur web

Mail

Des solutions pour mettre en place votre propre serveur de mails

Les autres services utiles

Échange de fichiers (FTP), suivi de versions (Git) et réseau privé (OpenVPN)

Applications web

Blog, galerie de photos, wiki, boutique en ligne, ...