

tutoriel, BROUILLON

# Auto-hébergement

Ce tutoriel décrit comment installer un serveur personnel complet tournant sous Debian et permettant d'héberger un certain nombre de services <sup>1)</sup>.

Tout sera géré en ligne de commande, à distance via SSH depuis un PC du réseau.



Mieux vaut tout mettre en place d'une traite jusqu'au serveur LAMP pour avoir une base fonctionnelle et sécurisée. On peut ensuite installer chaque service petit à petit.

## **Attention !**

S'auto-héberger, c'est être responsable d'une machine qui tourne 24h/24, qui héberge des services plus ou moins critiques et ouverte sur l'extérieur.



La machine doit :

- être sécurisée,
- tourner en permanence (une dizaine de watts pour un Raspberry Pi)

et il faut y consacrer du temps, en particulier pour la maintenance.

## Pré-requis

- un **ordinateur** dédié à cet usage, tournant sous un OS Debian <sup>2)</sup>. Cet ordinateur doit être allumé et connecté à Internet 24h/24.
- une **connexion xDSL** <sup>3)</sup>.
- On peut aussi (*facultatif*) :
  - acheter un **nom de domaine** ou d'autres services.
  - louer un serveur chez un hébergeur <sup>4)</sup> :
    - Un **VPS** (serveur virtuel, suffisant pour ce qu'on va installer) chez OVH revient ainsi à 5€ par mois hors taxe, 10€ par mois chez Dedibox avec un plus gros espace de stockage
    - Un **serveur dédié**, plus puissant et sur lequel vous aurez un accès root revient à

une dizaine d'euros par mois.

- un **onduleur** si l'alimentation électrique fluctue.
- etc.

## Première étape : mise en place d'un accès SSH au serveur

### Adresse IP du serveur

Pour connaître les adresses IP du serveur, voir [Connaître l'adresse IP d'une machine sous Linux](#)

### Mise en place du serveur SSH

- Dans le cas d'un Raspberry Pi, le serveur SSH est déjà en place et fonctionnel : il suffit de s'y connecter.
- Pour un PC, voir [Openssh : un serveur ssh](#)

### Connexion au serveur depuis une machine du réseau

Depuis une machine du réseau, lancez :

- `ssh -p port utilisateur@ip_du_serveur`

; mot de passe demandé

celui du compte *utilisateur* sur le serveur.

; **-p** (en minuscules)

permet de spécifier le port (22 par défaut, )



Attention à ne pas confondre : pour **scp**, le port s'indique par **-P** (en majuscule).

ou utilisez **Putty** sous Windows.

### Autres étapes

Désormais, la configuration se fera via SSH, donc à distance de la machine serveur qui n'aura plus

besoin d'écran ni de souris.

## Configuration du pare-feu

On va maintenant activer et configurer le pare-feu du noyau (**iptables**) pour protéger notre serveur.

Vérifiez qu'**iptables** est bien disponible en tapant :

- `sudo iptables -L`

qui va lister toutes les règles disponibles.

On va créer un script qui se lancera à chaque démarrage et définira les règles du pare-feu (iptables étant remis à zéro à chaque démarrage du serveur).

Ouvrez avec les droits d'administration le fichier **/etc/init.d/firewall** pour y écrire des règles.

Rendez-le exécutable :

- `chmod +x /etc/init.d/firewall`

et exécutez-le à chaque démarrage :

- `update-rc.d firewall defaults`



Attention, si vous configurez mal ces règles, vous risquez de ne plus pouvoir vous connecter en SSH à la machine. Une précaution minimale est d'avoir une session root ouverte en SSH et de **ne pas la fermer** avant d'avoir testé que les règles fonctionnent.

On a ainsi défini une politique assez restrictive pour le pare-feu :

- Par défaut, tout est bloqué en INPUT, OUTPUT et FORWARD.
- On autorise les connexions déjà existantes
- et on traite les nouvelles connexions au cas par cas.
- Quelques limitations et protection basiques sont également ajoutées.

Lisez bien la documentation d'iptables et celle qu'on trouve sur internet pour voir ce que fait chaque règle, avant de l'insérer dans le fichier de configuration.

Redémarrez le serveur pour vérifier qu'il n'y a aucun problème avec la configuration, en particulier

pour vérifier que vous pouvez encore vous connecter en SSH au serveur. Si ce n'est pas le cas, rebranchez un écran et un clavier sur le serveur et éditez les règles jusqu'à ce que ça marche (ou le redémarrer en mode rescue)

On peut aussi l'exécuter directement depuis la console via `sh /etc/init.d/firewall` en cas de besoin.

Pour vider votre configuration iptables, il faut rentrer



- `iptables -F; iptables -X`

puis remettre les politiques par défaut par protocoles comme vous le souhaitez (par exemple

```
iptables -P INPUT ACCEPT
```

).

## Installation de logiciels de sécurisation supplémentaires

### fail2ban

Voir

<http://www.commentcamarche.net/faq/18225-utiliser-fail2ban-pour-protoger-votre-application-web>

Fail2ban permet de bannir automatiquement de votre serveur des adresses IP qui échoueraient un trop grand nombre de fois à une authentification et de vous avertir par e-mail.

Installer le paquet fail2ban pour l'installer. Pour modifier la configuration, il faut modifier les fichiers `/etc/default/fail2ban`, `/etc/fail2ban/fail2ban.conf`, `/etc/fail2ban/jail.conf` et `/etc/fail2ban/jail.local`.

Ce dernier fichier est celui qui va nous intéresser (c'est en fait le même que `jail.conf` mais il est prévu pour être modifié et donc pour conserver les réglages après une mise à jour du paquet).

Configuration (fichier `jail.local`) :

Paramètre	Commentaire
<b>ignoreip</b>	Laisser 127.0.0.1 pour que fail2ban ne bloque jamais votre propre serveur.
<b>bantime</b>	Définit la durée du ban, 600 est une bonne valeur.
<b>maxretry</b>	Nombre d'échecs avant bannissement, 3 est bien aussi.
<b>destemail</b>	Mettre ici l'adresse e-mail à laquelle vous souhaitez recevoir des rapports de fail2ban.

Voilà pour la configuration générale. Il faut ensuite modifier les ports (si on les a modifié manuellement pour SSH par exemple) pour les faire correspondre aux ports réels, dans la section

jails. Par exemple, si votre connexion SSH se fait sur le port 3134 et non sur le port 22, il faut mettre 3134 à la place de 22 dans la section jails. N'oubliez pas de passer enabled à true pour les jails que vous souhaitez activer (sûrement ssh, ssh-ddos, apache, apache-noscript, apache-overflows pour commencer). Noter qu'il existe énormément de règles pré-définies, certaines pourront vous être utiles par la suite (postfix/sasl par exemple si vous installez un serveur de mails sur votre serveur).

On peut également rajouter des règles comme la règle apache-w00tw00t pour bloquer les requêtes w00tw00t.

On trouvera plus d'informations sur la configuration de fail2ban sur le wiki debian-fr

Vous pourrez alors tester fail2ban en entrant volontairement un mauvais mot de passe sur une page protégée par un fichier .htaccess (après avoir installé apache) ou sur une connexion SSH.

Normalement, vous devriez être banni. Soit vous attendez la fin du bantime, soit vous avez un autre accès à la machine et dans ce cas, vous pouvez supprimer la règle iptables qui vous interdit l'accès au serveur (fail2ban bloque en utilisant iptables). Une solution est d'avoir une session SSH ouverte en arrière-plan pour pouvoir se débannir facilement.

Note (peut-être dépassée lorsque vous lirez ceci) : il semble qu'il y ait un problème dans une regex de fail2ban pour identifier les erreurs de connexion avec postfix. Voir ce sujet sur [debien-fr.org](https://debien-fr.org) pour plus de détails et pour corriger la regex.

Liens divers :

- Utiliser fail2ban pour protéger votre application web (avec une partie sur l'écriture de règles personnalisées pour utiliser fail2ban avec ses scripts)
- Bannir les bots phpmyadmin et w00tw00t avec fail2ban
- la doc [ubuntu-fr.org](https://ubuntu-fr.org)

**portsentry**

**rkhunter**

**logwatch**

**Quelques astuces avant d'installer les services (linux, rien à voir en particulier avec l'autohébergement)**

**Utiliser des alias pour les commandes qu'on lancera souvent**

**Completion intelligente dans le shell**

**Réglage automatique de l'heure avec NTP**

## **(Putty) SSH en français**

## **Ok, mon serveur est installé, mais comment j'y accède ?**

## **Installation des services souhaités**

### **Installation d'un serveur LAMP**

#### **Installation des paquets**

#### **Configuration d'Apache et de PHP**

#### **Configuration de MariaDB**

#### **Test du serveur web/mariadb**

## **Quelques services webs utiles**

### **Un agrégateur de flux RSS**

**Shaarli de Sebsauvage** : l'alternative à del.icio.us rapide et libre (agrégation de marques-pages et plus)

### **Wordpress** : le moteur de blog

### **Owncloud** : l'alternative libre à dropbox (et bien plus)

### **Single File PHP Gallery** : Une galerie d'image qui tient en un seul fichier

### **Un wiki KISS avec WiKISS**

### **Roundcube / Squirrelmail** : le webmail chez soi

### **FluxBB**, le moteur de forum libre

### **Se passer de Google Analytics grâce à Piwik**

**L'alternative aux réseaux sociaux, pour tout poster chez soi et diffuser après, Known**

**Services divers**

**Utilisation de SSL avec son serveur Apache pour sécuriser les connexions**

**Installation d'un serveur Jabber pour héberger sa propre messagerie instantanée**

**Installation de Murmur (mumble-server) pour héberger son propre serveur Mumble**

**Installation d'un serveur email**

**Héberger ses propres DNS**

**Un serveur Git chez soi**

**Node.js**

**Partagez vos images avec Lutim**

**Adieu Facebook, bonjour Diaspora**

**Un remplaçant pour Google Docs**

**Mise en place d'une procédure de sauvegarde**

**Un mot sur la sécurité**

**Aller plus loin avec SSH**

**Envoyer des emails avec son serveur**

**Note à propos de FTP et de l'absence de doc sur l'installation d'un serveur FTP**

**Bonus : Utiliser son nom de domaine pour repérer ses périphériques**

**Liens en vracs**

## Conclusion

## Problèmes connus

## Voir aussi

- Pourquoi s'auto-héberger ?
  - **(fr)** [conférence de Benjamin Bayart](#), président de la FDN (French Data Network, FAI associatif français) ; il compare l'évolution actuelle d'Internet et le modèle du Minitel. Internet a été construit dans une optique de décentralisation mais à l'heure actuelle on est en train de recentraliser tous les services, comme le proposait le Minitel.
  - **(fr)** [Un aperçu global de ce qu'est l'auto-hébergement](#)
  - **(fr)** [Avantages et inconvénients](#) et [rappel des contraintes légales](#)
  - **(fr)** [Un autre aperçu général](#)
  - **(fr)** [Un témoignage parmi tant d'autres](#)
- **(fr)** <https://phyks.me/autohebergement.html>

---

Contributeurs principaux : [jamaïque](#).

<sup>1)</sup>

pour s'affranchir des services centralisés : Google, Twitter, DropBox ...

<sup>2)</sup>

un vieil ordinateur, même peu puissant ou un Raspberry Pi qui consomme très peu (10 W) ; seule l'unité centrale est nécessaire

<sup>3)</sup>

**ethernet** ou **CPL** si possible ; les débits en wifi sont insuffisants et trop fluctuants

<sup>4)</sup>

évite d'avoir à vous soucier du matériel

From:

<https://doc.nfrappe.fr/> - **Documentation du Dr Nicolas Frappé**

Permanent link:

<https://doc.nfrappe.fr/doku.php?id=tutoriel:internet:auto-hebergement:start0>



Last update: **2022/11/08 19:40**