tutoriel

LAMP : un serveur HTTP sous Linux, Apache 2, MySQL et PHP

Un serveur **LAMP** est un serveur Web basé sur **Apache**

L'acronyme **LAMP** signifie **L**inux **A**pache **M**ySQL **P**HP :

Linux
: système d'exploitation du système

Apache
: serveur HTTP

MySQL
: système de gestion de base de données.

PHP/Python/Perl
: langage de programmation associé

Pour d'autres solutions avec Lighttpd et Sqlite (moins gourmands en ressources) :

- **Lighttpd "how to" - serveur Web rapide et sécurisé**
- **Création d'un serveur HTTP (Lighty) + PHP + SQLite**

Pré-requis

Définir l'emplacement de la racine du serveur :Déplacer la racine (répertoire de base) d'un serveur HTTP

Autres étapes

Installation

L'installation se fait nécessairement **en ligne de commande**, *pour pouvoir répondre aux demandes*.

Mettez à jour le système :

```
...@...:~ $ sudo apt update
```

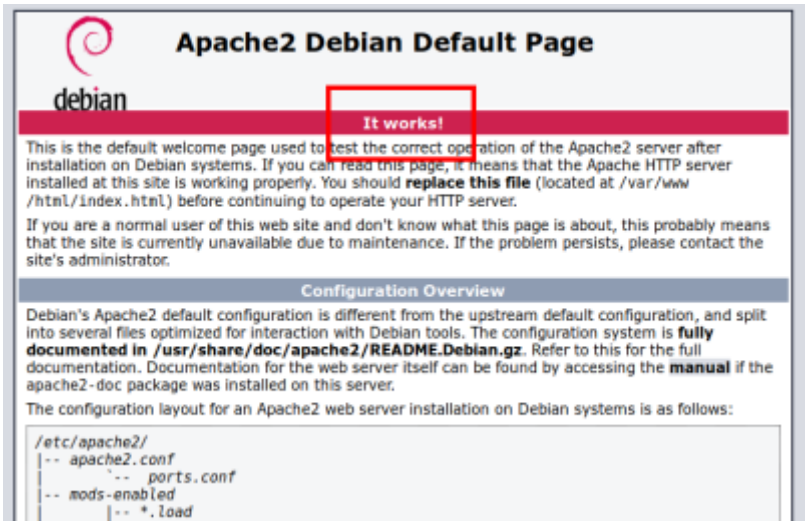
```
...@...:~ $ sudo apt upgrade
...@...:~ $ sudo apt dist-upgrade
...@...:~ $ sudo apt autoremove
...@...:~ $ sudo apt autoclean
```

Installez Apache, PHP et Mariadb :

```
...@...:~ $ sudo apt install {apache2,php,php-{pear,mysql},libapache2-mod-php,mariadb-server}
```

- **Apache** : serveur web [1)]
- **PHP** : langage interprété, pour rendre un site dynamique (l'utilisateur envoie des informations au serveur qui lui renvoie les résultats modifiés en fonction de ces infos).
- Lors de l'installation de Mariadb, un mot de passe est demandé pour le compte administrateur MySQL (admin), attention à bien le retenir, car il sera utilisé plus tard.
- **Mariadb** : SGBD libre et puissant

2. **Installez Adminer** (remplace avantageusement PHPMyAdmin) : interface simplifiée en PHP pour manipuler les bases de données SQL
3. **Testez** :
Apache : ouvrez la page **http://IP_du_serveur** (par exemple http://192.168.0.32). Si tout va bien, vous obtenez :

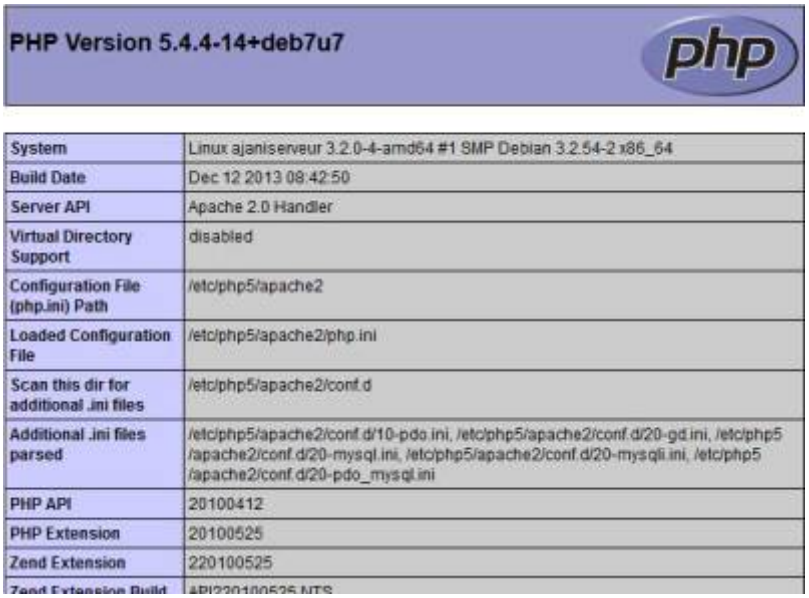


PHP :
créez le fichier :

/var/www/html/php/index.php

```
<?php phpinfo(); ?>
```

Ouvrez la page http://IP_du_serveur/php (par exemple http://192.168.0.32/php). Elle doit afficher :



PHP Version 5.4.4-14+deb7u7

System	Linux ajaniserveur 3.2.0-4-amd64 #1 SMP Debian 3.2.54-2 x86_64
Build Date	Dec 12 2013 08:42:50
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-gd.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
Zend Extension	220100525
Zend Extension Build	API220100525,NTS

3. **Mariadb** (sur la machine serveur) :

```
...@...:~ $ mysql -u admin -pVotreMotDePasse
Welcome to the MariaDB monitor.  Commands end with ; or \g.
...
MariaDB [(none)]> quit;
Bye
...@...:~ $
```

4. **Adminer** : accédez à l'adresse http://IP_du_serveur/adminer (par exemple http://192.168.0.32/adminer)

4. **Déplacer la racine du site**
Sauvegardez l'original du fichier /etc/apache2/sites-available/000-default :

```
...@...:~ $ sudo cp /etc/apache2/sites-available/000-default
/etc/apache2/sites-available/000-default.original
```

Éditez avec les droits d'administration le fichier **/etc/apache2/sites-available/000-default** pour y remplacer les occurrences de **/var/www/html** par le chemin voulu (par exemple **/[DISQUE]/www/html** :

- Remplacez

```
DocumentRoot /var/www/html
```

par

```
DocumentRoot /media/[DISQUE]/www
```

- Remplacez

```
<Directory /var/www/>
```

par

```
<Directory /media/[DISQUE]/www/html/>
```

3. **Redémarrez le serveur** apache par :

```
...@...:~ $ sudo apachectl restart
```

4. Désormais, en visitant l'adresse http://IP_du_serveur, on aboutit dans le répertoire /media/[DISQUE]/www.
5. **Installez les modules rewrite et vhost_alias** :

```
...@...:~ $ sudo a2enmod rewrite vhost_alias
...@...:~ $ sudo apachectl restart
```

6. Créez avec les droits d'administration le fichier **/etc/conf-available/monsite.tld.conf** (**monsite.tld** = adresse de votre site sur internet pour les visiteurs) :

/etc/conf-available/monsite.tld.conf

```
(...)
ServerName monsite.tld
```

7. Activez cette configuration et relancez apache :

```
...@...:~ $ sudo a2enconf monsite.tld
...@...:~ $ sudo apache2ctl restart
```

Exemple avec localhost :

/etc/apache2/conf-available/localhost.conf

```
ServerName localhost
<Directory />
        Options Indexes
FollowSymLinks Multiviews
        AllowOverride All
        Require all denied
</Directory>

<Directory /var/www/html/>
        Options Indexes
FollowSymLinks MultiViews
        AllowOverride All
        Require all granted
```

```
            </Directory>
```

```
...@...:~ $ sudo a2enconf localhost
...@...:~ $ sudo apache2ctl restart
```

Installation des Modules PHP

Vous pouvez installer des modules de PHP.

Pour lister les modules PHP5 disponibles, tapez ceci :

```
$ apt-cache search --names-only ^php-
```

cas particulier du Raspberry Pi

```
$ apt-cache search --names-only ^php5-
```

Installez ceux qui vous intéressent, en particulier :

```
$ sudo apt install php-dev php-gd php-curl php-pear dh-php dh-make
```

cas particulier du Raspberry Pi

```
$ sudo apt-get install php5-dev php5-gd php5-curl php-pear dh-php5 dh-make
```

Vous pouvez aussi installer les paquets :

```
$ sudo apt install php-cgi php-idn php-imagick php-imap php-intl php-mcrypt
php-memcache php-ming php-ps php-pspell php-recode php-snmp php-tidy php-
xmlrpc php-xsl
```

cas particulier du Raspberry Pi

```
$ sudo apt-get install php5-cgi php5-idn php5-imagick php5-imap php5-intl
php5-mcrypt php5-memcache php5-ming php5-ps php5-pspell php5-recode php5-
snmp php5-tidy php5-xmlrpc php5-xsl
```

Configuration

Nouveautés de la version 2 de Apache

Sections de configuration au niveau requête
: Les sections If, <ElseIf> et <Else> permettent de définir une configuration en fonction de critères liés à la requête.

Directive ~~NameVirtualHost~~
: Cette directive est maintenant obsolète.

Directives autorisées dans les fichiers .htaccess
: La nouvelle directive AllowOverrideList permet de contrôler de manière plus précise la liste des directives autorisées dans les fichiers .htaccess.

Variables dans les fichiers de configuration
: La directive Define permet de définir des variables dans les fichiers de configuration, améliorant ainsi la clarté de la présentation si la même valeur est utilisée en plusieurs points de la configuration.

Support des gros fichiers
: httpd supporte les fichiers d'une taille supérieure à 2GB.

Simplification de la Configuration
: Les directives Port et BindAddress ont disparu.

  Désormais seule la directive Listen sert de liaison pour les adresses IP
  la directive ServerName ne précise le nom du serveur et son port que pour les redirections et la gestion des hôtes virtuels.

mod_dav
: Apparu dans Apache httpd 2.0, ce module implémente les spécifications HTTP de gestion distribuée de versions et de rédaction (Distributed Authoring and Versioning - DAV), destinées à la mise en ligne et à la maintenance des contenus Web.

Pour la configuration d'Apache 2, voir le paragraphe Configuration de Apache 2 de la page qui présente **Apache 2**.

Pour permettre à chaque utilisateur d'avoir son propre site dans public_html, voir la page Apache 2 : un serveur web HTTP.

Sécurisation

Éditez avec les droits d'administration le fichier **/etc/apache2/conf-available/security.conf** pour mettre à **Off** les lignes :

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

```
ServerSignature Off
(...)
TraceEnable Off
```

Faites une copie de sauvegarde du fichier **/etc/apache2/sites-available/default** en la renommant, par exemple, **`default.original`**, puis éditez avec les droits d'administration le fichier **/etc/apache2/sites-available/default** pour le modifier comme ceci :

```
$ sudo cp /etc/apache2/sites-available/default /etc/apache2/sites-
available/default.original
```

Pour sécuriser le site, il faut d'abord rendre inaccessible de l'extérieur la racine et spécifier les accès autorisés à cette racine.

Créez avec les droits d'administration le fichier **/etc/apache2/conf-available/local** pour y écrire :

/etc/apache2/conf-available/local

```
<Directory /var/www/>
    Options Indexes FollowSymLinks MultiViews
    AllowOverride None
    Require all deny
    Allow from 127.0.0.1 # local
    Allow from 192.168.0.0/24 # reseau
</Directory>
```

source : http://doc.ubuntu-fr.org/tutoriel/lamp_repertoires_de_travail

Ainsi, seule la machine locale (127.0.0.1) et celles du réseau local (à condition que celui-ci utilise l'adresse 192.168.0.0) auront accès aux pages web situées dans le dossier /var/www/.

Préparation du fichier /etc/hosts

Le fichier **/etc/hosts** donne les correspondances entre les adresses IP et les noms de site.

Une petite modification est nécessaire pour un bon fonctionnement de Apache.

Éditez avec les droits d'administration le fichier **/etc/hosts** pour changer les deux lignes

/etc/hosts

```
127.0.0.1 localhost
...
127.0.0.1 nom_réseau_du_pc
```

en la ligne :

/etc/hosts

```
127.0.0.1    localhost.localdomain    localhost
nom_réseau_du_pc
```

Relancez le serveur apache :

```
...@...:~ $ sudo apache2ctl restart
```

Les hôtes virtuels

- Avec **Apache 2**, **les hôtes virtuels sont indispensables**.
- Il faut créer **un hôte virtuel pour chaque projet**
- Lancez les sites locaux par un hôte virtuel pour chacun : **http://projet1/** ou **http://projet2/** et non par **http://localhost/projet1/** ou **http://localhost/projet2/**,

Dans nos exemples, nous parlerons :

- du site **monsite.com** pour le domaine
- et du site **doc.monsite.com** pour les sous-domaines

Les hôtes virtuels d'Apache permettent d'exécuter sur le même ordinateur :

- différents serveurs pour différentes adresses IP,
- différents noms d'hôte
- ou différents ports

Pour chaque hôte virtuel, on crée un fichier de configuration dans **/etc/apache2/sites-available**. Le fichier config configure l'ensemble.

Activez les hôtes virtuels en lançant en ligne de commande :

```
$ sudo a2enmod vhost_alias
Enabling module vhost_alias.
To activate the new configuration, you need to run:
  service apache2 restart
```

Pour créer un **hôte virtuel** Apache, deux étapes :

Créez et activez la définition du **VirtualHost**

Ajoutez votre nouveau nom de domaine dans le fichier **/etc/hosts** ou sur votre serveur DNS.

Définition de l'hôte virtuel

<VirtualHost> (...) </VirtualHost>
groupe de directives qui ne s'appliquent qu'à un serveur virtuel particulier.
Toute directive autorisée dans un contexte de serveur virtuel peut être utilisée.
Lorsque le serveur reçoit un requête pour un

document hébergé par un serveur virtuel particulier, il applique les directives de configuration rassemblées dans la section <VirtualHost>. adresse IP peut être une des entités suivantes, éventuellement suivies d'un caractère ':' et d'un numéro de port (ou *) :

L'adresse IP du serveur virtuel
Un nom de domaine entièrement qualifié correspondant à l'adresse IP du serveur virtuel (non recommandé)
Le caractère *, qui agit comme un caractère générique, et correspond à toute adresse IP.
La chaîne _default_, dont la signification est identique à celle du caractère *

UseCanonicalName On|Off|DNS (défaut : UseCanonicalName Off)

Définit la manière dont le serveur détermine son propre nom et son port
Dans de nombreuses situations, Apache httpd doit construire une URL auto-identifiante – c'est à dire une URL qui fait référence au serveur lui-même. Avec UseCanonicalName On, Apache httpd va utiliser le nom d'hôte et le port spécifiés par la directive ServerName pour construire le nom canonique du serveur. Ce nom est utilisé dans toutes les URLs auto-identifiantes, et affecté aux variables SERVER_NAME et SERVER_PORT dans les programmes CGI.
Avec UseCanonicalName Off, Apache httpd va construire ses URLs auto-identifiantes à l'aide du nom d'hôte et du port fournis par le client, si ce dernier en a fourni un (dans la négative, Apache utilisera le nom canonique, de la même manière que ci-dessus). Ces valeurs sont les mêmes que celles qui sont utilisées pour implémenter les serveurs virtuels à base de nom, et sont disponibles avec les mêmes clients. De même, les variables CGI SERVER_NAME et SERVER_PORT seront affectées des valeurs fournies par le client.
Cette directive peut s'avérer utile, par exemple, sur un serveur intranet auquel les utilisateurs se connectent en utilisant des noms courts tels que www. Si les utilisateurs tapent un nom court suivi d'une URL qui fait référence à un

répertoire, comme http://www/splat, sans le slash terminal, vous remarquerez qu'Apache httpd va les rediriger vers http://www.example.com/splat/. Si vous avez activé l'authentification, ceci va obliger l'utilisateur à s'authentifier deux fois (une première fois pour www et une seconde fois pour www.example.com – voir la foire aux questions sur ce sujet pour plus d'informations). Par contre, si UseCanonicalName est définie à Off, Apache httpd redirigera l'utilisateur vers http://www/splat/.
Pour l'hébergement virtuel en masse à base d'adresse IP, on utilise une troisième option, UseCanonicalName DNS, pour supporter les clients anciens qui ne fournissent pas d'en-tête Host:. Apache httpd effectue alors une recherche DNS inverse sur l'adresse IP du serveur auquel le client s'est connecté afin de construire ses URLs auto-identifiantes.

ServerName [protocole://]nom-de-domaine|adresse-ip[:port]
Nom d'hôte et port que le serveur utilise pour s'authentifier lui-même.
Cette directive est aussi utilisée lors de la création d'URLs de redirection relatives quand la directive UseCanonicalName est définie à une valeur autre que la valeur par défaut.
Par exemple, si le nom de la machine hébergeant le serveur web est simple.example.com, la machine possède l'alias DNS www.example.com, et si vous voulez que le serveur web s'identifie avec cet alias, vous devez utilisez la définition suivante :
ServerName www.example.com
La directive ServerName peut apparaître à toutes les étapes de la définition du serveur. Toute occurrence annule cependant la précédente (pour ce serveur).
Si la directive ServerName n'est pas définie, le serveur tente de déterminer le nom d'hôte visible du point de vue du client en demandant tout d'abord au système d'exploitation le nom d'hôte système, et en cas d'échec, en effectuant une recherche DNS inverse sur une adresse IP présente sur le système.
Si la directive ServerName ne précise pas de port, le serveur utilisera celui de la requête entrante. Il est recommandé de spécifier un nom d'hôte et un port spécifiques à l'aide de la directive ServerName pour une fiabilité optimale et à titre préventif.

Si vous définissez des serveurs virtuels à base de nom, une directive ServerName située à l'intérieur d'une section <VirtualHost> spécifiera quel nom d'hôte doit apparaître dans l'en-tête de requête Host: pour pouvoir atteindre ce serveur virtuel.
Parfois, le serveur s'exécute en amont d'un dispositif qui implémente SSL, comme un mandataire inverse, un répartiteur de charge ou un boîtier dédié SSL. Dans ce cas, spécifiez le protocole https:// et le port auquel les clients se connectent dans la directive ServerName, afin de s'assurer que le serveur génère correctement ses URLs d'auto-identification.
Voir la description des directives UseCanonicalName et UseCanonicalPhysicalPort pour les définitions qui permettent de déterminer si les URLs auto-identifiantes (par exemple via le module mod_dir) vont faire référence au port spécifié, ou au port indiqué dans la requête du client.

ServerAlias nom serveur [nom serveur] ...
Autres noms d'un serveur utilisables pour atteindre des serveurs virtuels à base de nom
ServerPath chemin d'URL
Nom de chemin d'URL hérité pour un serveur virtuel à base de nom accédé par un navigateur incompatible

/etc/apache2/sites-available/host.example.com.conf

```
<VirtualHost 10.1.2.3:80>
  ServerAdmin
webmaster@host.example.com
  DocumentRoot
"/www/docs/host.example.com"
  ServerName host.example.com
  ErrorLog "logs/host.example.com-
error_log"
  TransferLog
"logs/host.example.com-access_log"
</VirtualHost>
```

/etc/apache2/sites-available/monsite.com.conf

```
<VirtualHost *:80>
    ServerName www.monsite.com
    ServerAlias monsite.com
    ServerAdmin admin@mail.fr
```

```
    VirtualDocumentRoot /var/www/html/monsite.com

    <Directory />
        Options Indexes FollowSymLinks MultiViews
        AllowOverride All
        Require all denied
    </Directory>

    <Directory /var/www/html/monsite.com>
        Options Indexes FollowSymLinks MultiViews
        AllowOverride All
        Require all granted
    </Directory>
</VirtualHost>
```

www.exemple.com

```
  ServerName www.exemple.com
  ServerAlias exemple.com
  ServerAdmin webmaster@localhost
  DocumentRoot /var/www/html/exemple.com
  <Directory "/">
    Require all denied
  </Directory>
  <Directory "/var/www/html/exemple.com">
    Require all granted
  </Directory>
```

ServerName
: définit la requête, le nom d'hôte et le port que le serveur utilise pour s'identifier. C'est utilisé lors de la redirection d'URL. Dans le contexte des hôtes virtuels, la directive ServerName spécifie le nom d'hôte qui doit apparaître dans l'en-tête de la requête pour correspondre à cet hôte virtuel.

<Directory “/”> ...
: Interdit l'accès à tout ; la suite va préciser ce qui est accessible.

Require all granted
: accès autorisé sans restriction.

Require all denied
: accès systématiquement refusé.

Création d'un sous-domaine en local

Pour cela :

- on crée un répertoire pour le sous-domaine
- on modifie le fichier hosts pour y ajouter le sous-domaine
- on crée un fichier du nom du sous-domaine dans le répertoire sites-available

- on crée un lien par a2ensite
- et on relance apache

Supposons que nous voulons créer le sous-domaine `doc.localhost` tel qu'en utilisant l'adresse http://doc.localhost, on accède au répertoire /srv/www/dokuwiki (la racine a été déplacée en /srv/www et redirigée vers le home selon la méthode du paragraphe 2.

On commence par créer un sous-répertoire pour le sous-domaine :

```
$ sudo mkdir /srv/www/dokuwiki
```

(attention, il faut être root, d'où l'utilité de rediriger…)

ou plus simplement, si on a redirigé vers le home, on crée le sous-répertoire dokuwiki dans le répertoire ~/www (donc ~/www/dokuwiki)

Puis on modifie le fichier /etc/hosts en ajoutant à la fin la ligne :

```
  127.0.0.1  doc.localhost
```

Faire une copie du fichier /etc/apache2/sites-available/default en la nommant par exemple doc.localhost :

```
$ sudo cp /etc/apache2/sites-available/default /etc/apache2/sites-
available/doc.localhost
```

Éditez avec les droits d'administration le fichier **/etc/apache2/sites-available/doc.localhost** pour y ajouter les lignes :

```
<VirtualHost *:80>
 DocumentRoot "/srv/www/html"
 ServerName localhost
</VirtualHost>

<VirtualHost *:80>
    DocumentRoot "/srv/www/html/dokuwiki"
    ServerName doc.localhost
</VirtualHost>
```

Il ne reste plus qu'à enregistrer ce nouveau fichier et à relancer apache :

```
$ sudo a2ensite doc.localhost
$ sudo /etc/init.d/apache2 reload
```

Le contenu de dokuwiki s'affiche désormais en tapant http://doc.localhost.

Configuration d'hôtes virtuels

Voir la page : Apache 2 : sous-domaines (serveurs virtuels)

Pour créer un hôte virtuel, vous devrez modifier les lignes d'hôte virtuel, fournies à titre d'exemple, dans le fichier httpd.conf, ou créer votre propre section d'hôte virtuel. N'oubliez pas que vous ne pouvez utiliser des hôtes virtuels basés sur le nom qu'avec votre serveur Web non sécurisé ; vous devrez utiliser des hôtes virtuels basés sur adresse IP si vous avez besoin d'autres hôtes virtuels compatibles SSL. Les exemples de lignes se présentent comme suit :

```
<VirtualHost ip.address.of.host.some_domain.com>
    ServerAdmin webmaster@host.some_domain.com
    DocumentRoot /www/docs/host.some_domain.com
    ServerName host.some_domain.com
    ErrorLog logs/host.some_domain.com-error_log
    CustomLog logs/host.some_domain.com-access_log common
</VirtualHost>
```

Ajoutez les informations correctes concernant votre ordinateur et/ou votre hôte virtuel à chaque ligne.

première ligne : remplacez **adresse.ip.domaine.com** → adresse IP de votre propre serveur

ServerName → un nom DNS valable à utiliser pour l'hôte virtuel. (n'inventez rien)

Vous devrez aussi supprimer le commentaire de l'une des lignes NameVirtualHost dans le fichier httpd.conf :

```
NameVirtualHost 12.34.56.78:80
#NameVirtualHost 12.34.56.78
```

Remplacez le l'exemple d'adresse IP par l'adresse IP (et le port, si nécessaire) correspondant à cet hôte virtuel.

Pour qu'un hôte virtuel travaille spécifiquement pour un port donné ajoutez le numéro de port à la première ligne de la configuration de l'hôte virtuel omme ceci :

```
<VirtualHost adresse_ip_du_serveur:12331>
```

Cette ligne créerait un hôte virtuel scrutant le port 12331. Ajoutez l'adresse IP de votre serveur et substituez le numéro de port que vous voulez utiliser à 12331 dans l'exemple précédent.

Sous les lignes Listen du fichier httpd.conf, ajoutez une ligne comme la suivante, qui donnera pour instruction au serveur Web de scruter le port 12331 :

```
Listen 12331
```

Création de sous-domaines automatisés

Installez le module vhost_alias :

```
...@...:~ $ sudo a2enmod vhost_alias
...@...:~ $ /etc/init.d/apache2 reload
```

Tout se passe dans le répertoire **/etc/apache2/sites-available/** qui contient un fichier de

configuration par serveur.

Le serveur par défaut est défini par le fichier **default**.

Nous prendrons comme exemple le sous-domaine **doc.framboise.home** pour un dokuwiki de racine **/media/Reservoirs/www/doc**.

- Sous-répertoires de domaine :
  - **xxx**.domaine.tld → répertoire www/domaine/**xxx**
  - **yyy**.domaine.tld → répertoire www/domaine/**yyy**
  - Mais on peut aussi faire des répertoires plus élaborés :
    - **aaa**.domaine.tld → répertoire www/domaine/**aaa**/un_répertoire
    - **bbb**.domaine.tld → répertoire www/domaine/**bbb**/un_répertoire

    Créez une copie de **/etc/apache2/sites-available/default** → **/etc/apache2/sites-available/framboise.home**

    Ajoutez une directive VirtualDocumentRoot et ServerAlias dans la configuration de votre VirtualHost comme dans l'ex suivant :

    framboise.home

```
# obtenir le nom du serveur à partir de l'entête "Host:"
UseCanonicalName Off

<VirtualHost *:80>
    DocumentRoot /media/Reservoirs/www
    ServerName framboise.home
    ServerAlias *.framboise.home
    VirtualDocumentRoot /media/Reservoirs/%-3
    # format de journal avec l'appel
    LogFormat "%V %h %l %u %t \"%r\" %s %b" vcommon
    CustomLog access.log vcommon
</VirtualHost>

# réglages pour les répertoires du dokuwiki
<Directory /media/Reservoirs/www/doc>
    Order deny,allow
    allow from all
</Directory>

<LocationMatch "/doc/(data|conf|bin|inc)/">
    Order allow,deny
    Deny from all
    Satisfy All
</LocationMatch>
```

    Ici, %-3 permet d'éviter les problèmes dus à www.

    - variable **%0** = le sous-domaine déduit de l'adresse appelée. (xxx.domaine.com)

- variable **%1** = premier élément de l'adresse appelée. (xxx)
- variable **%2** = 2e élément de l'adresse appelée. (domaine)
- variable **%-1** = dernier élément de l'adresse appelée. (com)
- variable **%-2** = avant-dernier élément de l'adresse appelée. (domaine)
- variable **%-3** = avant-avant-dernier élément de l'adresse appelée. (xxx, yyy, ...)

3. Enregistrez le fichier framboise.home et relancez Apache :

```
...@...:~ $ sudo a2ensite framboise.home
...@...:~ $ sudo service apache2 restart
```

Méthode simple pour un domaine

Pour créer un sous-domaine (par ex. doc.framboise), nous allons créer un fichier **doc.framboise** dans ce répertoire.

Éditez avec les droits d'administration le fichier **/etc/hosts** en lui ajoutant la ligne :

/etc/hosts

```
192.168.0.100  doc.framboise
```

Redémarrez Apache :

```
...@...:~ $ sudo /etc/init.d/apache2 restart
```

Copiez le fichier /etc/apache2/sites-available/default → doc.framboise :

```
...@...:~ $ scp framboise:/etc/apache2/sites-available/default
/etc/apache2/sites-available/doc.framboise
```

Éditez avec les droits d'administration le fichier **/etc/apache2/sites-available/doc.framboise** pour y ajouter les lignes :

doc.framboise

```
NameVirtualHost doc.framboise

<VirtualHost doc.framboise>
    ServerAdmin webmaster@domaine.fr
    ServerName www.doc.framboise
    ServerAlias doc.framboise
    DocumentRoot /media/Reservoirs/www/dokuwiki

    # réglages pour le répertoire
```

```
    <Directory /media/Reservoirs/www/dokuwiki>
        Order deny,allow
        allow from all
    </Directory>
    <LocationMatch "/dokuwiki/(data|conf|bin|inc)/">
        Order allow,deny
        Deny from all
        Satisfy All
    </LocationMatch>
</VirtualHost>
```

Activez le site et relancez apache :

```
...@...:~ $ sudo a2ensite doc.framboise
...@...:~ $ sudo /etc/init.d/apache2 restart
```

Pour pouvoir accéder depuis deux adresses (par exemple une en local et une sur internet) (Les adresses doc.framboise et 1.2.3.4 doivent exister dans le DNS ou le /etc/host) :

doc.framboise

```
NameVirtualHost doc.framboise
NameVirtualHost 1.2.3.4

<VirtualHost 1.2.3.4 doc.framboise>
    ServerAdmin webmaster@domaine.fr
    ServerName www.doc.framboise
    ServerAlias doc.framboise
    DocumentRoot /media/Reservoirs/www/dokuwiki

    # réglages pour le répertoire
    <Directory /media/Reservoirs/www/dokuwiki>
        Order deny,allow
        allow from all
    </Directory>
    <LocationMatch "/dokuwiki/(data|conf|bin|inc)/">
        Order allow,deny
        Deny from all
        Satisfy All
    </LocationMatch>
</VirtualHost>
```

Autres exemples de sous-domaines

fichier de configuration apache /etc/apache2/sites-available/domaine.com (http://forum.ubuntu-fr.org/viewtopic.php?id=361680) :

/etc/apache2/sites-available/domaine.com

```
NameVirtualHost *:80

<VirtualHost *:80>

    ServerAdmin admin@domaine.com
    ServerName   www.domaine.com
    ServerAlias domaine.com

    UseCanonicalName Off

    DocumentRoot /websites/www.domaine.com
        <Directory /websites/www.domaine.com>
            Options Indexes FollowSymlinks
            AllowOverride none
            Order deny,allow
            Allow from all
        </Directory>

    ErrorLog /var/log/apache2/www_error.log
    LogLevel warn
    CustomLog /var/log/apache2/www_access.log combined
    ServerSignature On

</VirtualHost>
```

UseCanonicalName Off → le nom du serveur est déduit du contenu de l'entête Host: de la requête0.

/etc/apache2/sites-available/sql.domaine.com

```
<VirtualHost *:80>
    ServerName sql.domaine.com
    ServerAdmin "admin@domaine.com"

    DocumentRoot /websites/sql.domaine.com
        <Directory /websites/sql.domaine.com>
            Options Indexes FollowSymlinks
            AllowOverride none
            Order deny,allow
            Allow from all
        </Directory>
    ErrorLog /var/log/apache2/sql_error.log
    LogLevel warn
    CustomLog /var/log/apache2/sql_access.log combined
</VirtualHost>
```

/etc/apache2/sites-available/dev.domaine.com

```
<VirtualHost *:80>
        ServerName dev.domaine.com
        ServerAdmin "admin@domaine.com"

        DocumentRoot /websites/dev.domaine.com
        <Directory /websites/dev.domaine.com>
            Options Indexes FollowSymlinks
            AllowOverride none
            Order deny,allow
            Allow from all
        </Directory>


        ErrorLog /var/log/apache2/dev_error.log
        LogLevel warn
        CustomLog /var/log/apache2/dev_access.log
combined
</VirtualHost>
```

/etc/apache2/sites-available/phpmyadmin.domaine.com

```
<VirtualHost *:80>
        ServerName phpmyadmin.domaine.com
        ServerAdmin "admin@domaine.com"

        DocumentRoot /websites/phpmyadmin.domaine.com
        <Directory /websites/phpmyadmin.domaine.com>
            Options Indexes FollowSymlinks
            AllowOverride none
            Order deny,allow
            Allow from all
        </Directory>
        ErrorLog /var/log/apache2/phpmyadmin_error.log
        LogLevel warn
        CustomLog /var/log/apache2/phpmyadmin_access.log
combined
</VirtualHost>
```

blog.domaine.com

```
<VirtualHost *:80>
        ServerName blog.domaine.com
       ServerAdmin "admin@domaine.com"

        DocumentRoot /websites/blog.domaine.com
        <Directory /websites/blog.domaine.com>
            Options Indexes FollowSymlinks
            AllowOverride none
```

```
            Order deny,allow
            Allow from all
        </Directory>
        ErrorLog /var/log/apache2/blog_error.log
        LogLevel warn
        CustomLog /var/log/apache2/blog_access.log
combined
</VirtualHost>
```

Rendre accessible votre serveur depuis internet

Pour rendre accessible votre serveur depuis internet, allez dans la gestion de votre box (livebox, freebox, etc.) et mettez en place la redirection des ports.

Nous allons configurer le routeur pour autoriser les connexions ssh et des trafics Web grâce à son pare-feu au Raspberry Pi.

Par sécurité, le mot de passe de l'utilisateur pi doit avoir été changé et éventuellement un nouvel utilisateur mis en place.

Il faut rediriger les adresses parvenant sur le routeur (la freebox) vers le Raspberry Pi.

Les ports à rediriger en TCP sont :

- le port 80 (pour http)
- le port 22 (pour ssh)
- le port 1723 (pour vpn)

Optimisation du Raspberry Pi pour Lighttpd Owncloud, WordPress et plus

source : http://c-mobberley.com/wordpress/index.php/2013/05/18/raspberry-pi-lighttpd-optimisation-for-wordpress-owncloud-and-more/

Lighttpd, sont installés. Nous allons optimiser **lighttpd** pour le Raspberry Pi en installant PHP APC, en activant fast cgi et en mettant en place la compression avec gzip.

Tout d'abord, Ouvrir avec les droits d'administration le fichier **/etc/lighttpd/lighttpd.conf** et remplir la section server.modules avec les modules suivants :

«code - > server.modules = (

```
  #"mod_rewrite",
  "mod_redirect",
  "mod_alias",
  "mod_access",
  #"mod_auth",
  #"mod_status",
  #"mod_simple_vhost",
  #"mod_evhost",
  #"mod_userdir",
  #"mod_secdownload",
  #"mod_fastcgi",
  #"mod_proxy",
  #"mod_cgi",
  #"mod_ssi",
  #"mod_compress",
  #"mod_usertrack",
  #"mod_expire",
  #"mod_rrdtool",
  #"mod_accesslog"
```

) </code>

PostgreSQL

- Voir la page PostGreSQL sur un Raspberry Pi : une base de données SQL

Configurer le SSL avec Apache 2

Le site par défaut de notre serveur LAMP utilise le protocole non-sécurisé HTTP et écoute sur le port 80.

Par souci de sécurité et de confidentialité, il peut être intéressant de le passer en HTTPS qui écoute sur le port 443.

Le plus rapide : utiliser les certificats SSL par défaut d'Apache 2.

Par défaut, Apache 2 contient deux sites préconfigurés : **default** et **default-ssl** qui pointent tous les deux vers le répertoire « /var/www » mais le premier écoute sur le port 80 (HTTP) et le second sur le port 443 (HTTPS).

Dans la configuration d'origine, seul le site **default** est actif, ce qui permet d'accéder à la page **It Works !** après l'installation.

Pour activer l'accès en https :

- Activer le module SSL d'Apache
- Activer le site « default-ssl » d'Apache
- recharger Apache et le site sera accessible en HTTPS.

Voici les commandes à saisir :

```
...@...:~ $ sudo a2enmod ssl
...@...:~ $ sudo a2ensite default-ssl.conf
...@...:~ $ sudo apachectl restart
```

</code>Pas besoin de générer de certificat SSL : il y en a déjà un par défaut (valable 10 ans). Son emplacement peut se lire dans le fichier **/etc/apache2/sites-available/default-ssl**

Répertoires de travail

Créez les répertoires voulus : ici, un répertoire public et un répertoire privé :

```
$ mkdir /srv/www/html/public
$ mkdir /srv/www/html/private
```

Donner les droits utilisateur :

```
$ sudo chown -R $USER:users /srv/www
```

Pour accéder à ces répertoires, il faut un alias de la forme : http://nom_de_domaine.tld/nom_de_redirection

Il faut donc ajouter deux alias, un pour public et un pour private.

Éditez avec les droits d'administration le fichier **/etc/apache2/sites-available/default** pour ajouter :

/etc/apache2/sites-available/default

```
...
## ZONE PUBLIQUE
Alias /public /srv/www/html/public
<Directory /srv/www/html/public>
    Options Indexes FollowSymLinks MultiViews
    AllowOverride All
    Require all granted
</Directory>

## ZONE PRIVEE
Alias /private /srv/www/html/private
<Directory /srv/www/html/private>
    Options Indexes FollowSymLinks MultiViews
    AllowOverride All
```

```
    Require all deny
    Allow from 127.0.0.1            # Local
    Allow from 192.168.1.0/24       # Réseau
    Allow from .w3.org              # W3C
    Allow from sandbox.paypal.com   # Paypal
</Directory>
```

Cela règle les accès à la zone privée.

Conclusion

Problèmes connus

Voir aussi

- **(fr)** https://raspbian-france.fr/installer-serveur-web-raspberry/
- **(fr)** http://howto.landure.fr_gnu-linux_debian-4-0-etch_installer-et-configurer-apache-2-sur-debian-4-0-etch
- **(fr)** http://wiki.gandi.net/fr/hosting/using-linux/tutorials/ubuntu/virtualhosts
- **(fr)** http://julien-pauli.developpez.com/tutoriels/apache/vhosts/
- **(fr)** http://httpd.apache.org/docs/2.2/fr/vhosts/examples.html
- vhosts

Basé sur « http://howto.landure.fr_gnu-linux_debian-4-0-etch_installer-et-configurer-apache-2-sur-debian-4-0-etch » par Landure.

1)
le plus utilisé, abondamment documenté, un choix sûr