

TP Administration : Serveur HTTP (Apache)

© 2019 tv <tvaira@free.fr> - v.1.0

Mise en situation	2
Apache HTTP Server	2
Installation	3
Tests	4
Configuration	6
Modules	7
<i>Virtual Hosts</i>	7
Journalisation	12
Moteur de réécriture d'URL	13
Module PHP	14

TP Administration

L'objectif de cette activité est de réaliser une installation d'un serveur Web Apache sur une machine serveur.



Mise en situation

Vous devez disposer d'un PC possédant un système d'exploitation Linux ou Windows et du logiciel de virtualisation *VirtualBox*. Le système invité sera une installation du **serveur Ubuntu 18.04 LTS**.

Apache HTTP Server

Le logiciel libre *Apache HTTP Server* (Apache) est un **serveur HTTP** créé et maintenu au sein de la fondation Apache. C'est le serveur HTTP le plus populaire du *World Wide Web*.

La version 2 d'Apache propose entre autres le support de plusieurs plates-formes (Windows, Linux et UNIX, Solaris, BSD, MAC OS X), le support de processus légers UNIX (*threads*), une nouvelle API et le support IPv6. [Source : http://fr.wikipedia.org/wiki/Apache_HTTP_Server]

```
$ apt-cache search apache | grep -E "^apache"
apache2 - Serveur HTTP Apache
apache2-bin - Serveur HTTP Apache (modules et autres fichiers binaires)
apache2-data - Serveur HTTP Apache (fichiers communs)
apache2-dbg - Symboles de débogage Apache
apache2-dev - Serveur HTTP Apache (en-têtes de développement)
apache2-doc - Serveur HTTP Apache (documentation sur site)
apache2-ssl-dev - Apache HTTP Server (mod_ssl development headers)
apache2-utils - Serveur HTTP Apache - utilitaires pour serveurs web
apache2-suexec-custom - programme configurable suexec pour mod_suexec du serveur HTTP Apache
apache2-suexec-pristine - programme suexec standard pour mod_suexec du serveur HTTP Apache
apachedex - calcul d'APDEX à partir de journaux dans le style d'Apache
apacheds - serveur d'annuaire d'Apache Directory
apachetop - Outil de surveillance de Apache en temps réel
```

Apache est conçu pour prendre en charge de nombreux **modules** lui donnant des fonctionnalités supplémentaires : interprétation du langage Perl, PHP, Python et Ruby, serveur *proxy*, *Common Gateway Interface*, *Server Side Includes*, réécriture d'URL, négociation de contenu, protocoles de communication additionnels, etc ...

```
$ apt-cache search libapache | grep -E "^libapache"
libapache2-mod-auth-pgsql - module pour Apache2 d'authentification avec PostgreSQL
libapache2-mod-auth-plain - Module pour Apache2 permettant l'authentification en texte brut
(non chiffré)
```

libapache2-mod-perl2 - intégration de Perl avec le serveur Web Apache2

...

```
$ apt-cache search libapache | grep -E "^libapache" | wc -l
126
```



Néanmoins, il est à noter que l'existence de nombreux modules Apache complexifie la configuration du serveur web. En effet, les bonnes pratiques recommandent de ne charger que les modules utiles : de nombreuses failles de sécurité affectant uniquement les modules d'Apache sont régulièrement découvertes.

Installation

Il faut installer le paquet apache2 :

```
$ sudo apt update
```

```
$ sudo apt-get install apache2
```

Lecture des listes de paquets... Fait

Construction de l'arbre des dépendances

Lecture des informations d'état... Fait

Les paquets supplémentaires suivants seront installés :

```
apache2-bin apache2-data apache2-utils libapr1 libaprutil1 libaprutil1-dbd-sqlite3
libaprutil1-ldap liblua5.2-0 ssl-cert
```

Paquets suggérés :

```
www-browser apache2-doc apache2-suexec-pristine | apache2-suexec-custom openssl-blacklist
```

Les NOUVEAUX paquets suivants seront installés :

```
apache2 apache2-bin apache2-data apache2-utils libapr1 libaprutil1 libaprutil1-dbd-sqlite3
libaprutil1-ldap liblua5.2-0 ssl-cert
```

0 mis à jour, 10 nouvellement installés, 0 à enlever et 8 non mis à jour.

Il est nécessaire de prendre 1 729 ko dans les archives.

Après cette opération, 6 982 ko d'espace disque supplémentaires seront utilisés.

...

Question 1. Installer le serveur HTTP Apache 2.

Une fois l'installation terminée, on peut vérifier l'état du serveur HTTP Apache :

```
$ systemctl status apache2
```

apache2.service - The Apache HTTP Server

Loaded: loaded (/lib/systemd/system/apache2.service; enabled; vendor preset: enabled)

Drop-In: /lib/systemd/system/apache2.service.d

- apache2-systemd.conf

Active: active (running) since Tue 2019-09-10 06:17:22 UTC; 1min 33s ago

Main PID: 5872 (apache2)

Tasks: 55 (limit: 1109)

CGroup: /system.slice/apache2.service

|

- 5872 /usr/sbin/apache2 -k start

- 5874 /usr/sbin/apache2 -k start

- 5875 /usr/sbin/apache2 -k start

```
sept. 10 06:17:22 serveur systemd[1]: Starting The Apache HTTP Server...
```

```
sept. 10 06:17:22 serveur apachectl[5849]: AH00558: apache2: Could not reliably determine
the server's fully qualified domain name, using 127.0.1.1. Set the 'ServerName' dir
sept. 10 06:17:22 serveur systemd[1]: Started The Apache HTTP Server.
```

```
// Anciennement :
```

```
$ /etc/init.d/apache2 status
Apache2 is running (pid 4060).
```

```
$ service apache2 status
```

Question 2. Vérifier l'état du serveur HTTP Apache 2.

Tests

Quelle est la version installée ?

```
$ apache2 -v
Server version: Apache/2.4.29 (Ubuntu)
Server built: 2019-08-26T13:41:23
```

Existe-t-il une page d'accueil ?

```
$ ls -l /var/www/html
total 12
-rw-r--r-- 1 root root 10918 sept. 10 06:17 index.html
```


Le meilleur moyen de tester consiste à demander la page d'accueil à Apache.

Il est possible d'accéder à la page par défaut via l'adresse IP du serveur Apache. Si vous ne connaissez pas l'adresse IP de votre serveur, vous pouvez l'obtenir de différentes manières à partir de la ligne de commande.

```
// Adresse locale privée
$ hostname -I
192.168.52.18 2a01:cb1c:91c:b500:a00:27ff:fef8:2b7b
```

```
// Adresse publique
$ curl -4 icanhazip.com
```

Il suffit d'entrer dans la barre d'adresse du navigateur : http://adresse_ip_serveur



Apache2 Ubuntu Default Page

It works!

This is the default welcome page used to test the correct operation of the Apache2 server after installation on Ubuntu systems. It is based on the equivalent page on Debian, from which the Ubuntu Apache packaging is derived. If you can read this page, it means that the Apache HTTP server installed at this site is working properly. You should **replace this file** (located at `/var/www/html/index.html`) before continuing to operate your HTTP server.

If you are a normal user of this web site and don't know what this page is about, this probably means that the site is currently unavailable due to maintenance. If the problem persists, please contact the site's administrator.

Configuration Overview

Ubuntu's Apache2 default configuration is different from the upstream default configuration, and split into several files optimized for interaction with Ubuntu tools. The configuration system is **fully documented in `/usr/share/doc/apache2/README.Debian.gz`**. Refer to this for the full documentation. Documentation for the web server itself can be found by accessing the **manual** if the `apache2-doc` package was installed on this server.

The configuration layout for an Apache2 web server installation on Ubuntu systems is as follows:

```
/etc/apache2/
|-- apache2.conf
|   |-- ports.conf
|-- mods-enabled
|   |-- *.load
|   |-- *.conf
|-- conf-enabled
|   |-- *.conf
|-- sites-enabled
|   |-- *.conf
```

- `apache2.conf` is the main configuration file. It puts the pieces together by including all remaining configuration files when starting up the web server.
- `ports.conf` is always included from the main configuration file. It is used to determine the listening ports for incoming connections, and this file can be customized anytime.
- Configuration files in the `mods-enabled/`, `conf-enabled/` and `sites-enabled/` directories contain particular configuration snippets which manage modules, global configuration fragments, or virtual host configurations, respectively.
- They are activated by symlinking available configuration files from their respective `*-available/` counterparts. These should be managed by using our helpers **`a2enmod`**, **`a2dismod`**, **`a2ensite`**, **`a2dissite`**, and **`a2enconf`**, **`a2disconf`**. See their respective man pages for detailed information.
- The binary is called `apache2`. Due to the use of environment variables, in the default configuration, `apache2` needs to be started/stopped with `/etc/init.d/apache2` or `apache2ctl`. **Calling `/usr/bin/apache2` directly will not work** with the default configuration.

Document Roots

By default, Ubuntu does not allow access through the web browser to any file apart of those located in `/var/www`, **`public_html`** directories (when enabled) and `/usr/share` (for web applications). If your site is using a web document root located elsewhere (such as in `/srv`) you may need to whitelist your document root directory in `/etc/apache2/apache2.conf`.

The default Ubuntu document root is `/var/www/html`. You can make your own virtual hosts under `/var/www`. This is different to previous releases which provides better security out of the box.

Reporting Problems

Please use the `ubuntu-bug` tool to report bugs in the Apache2 package with Ubuntu. However, check **existing bug reports** before reporting a new bug.

Please report bugs specific to modules (such as PHP and others) to respective packages, not to the web server itself.

Question 3. Tester le serveur HTTP Apache 2 en accédant à sa page d'accueil.

On modifie la page d'accueil existante du serveur web :

```
$ sudo mv /var/www/html/index.html /var/www/html/index.html.bak
```

```
$ sudo vim /var/www/html/index.html
```

```
<html><body><h1>It works!</h1>
<p>En construction ...</p>
</body></html>
```

Pour tester directement sur le serveur, il faut installer `lynx`, un navigateur en mode console :

```
$ sudo apt-get install lynx
```

Puis, on accède à la page :

```
$ lynx http://127.0.0.1/
```

Question 4. Tester localement la nouvelle page d'accueil.



En cas d'erreur 403 (*Forbidden*), il vous faut évidemment vérifier tout d'abord les droits d'accès (XX5 pour les répertoires et XX4 pour les fichiers). En cas de présence d'une directive `Directory` sur `/var/www` dans `apache2.conf` pour une version **2.4.6-2** d'Apache, il vous faut appliquer la règle : `Require all granted` au lieu de `Require all denied`.

Configuration



La version d'Apache installée sur les distributions d'Ubuntu est une version **2.4.x**. Cette version implique des changements de certaines directives par rapport à la version précédente 2.2.

Lire : <http://httpd.apache.org/docs/2.4/upgrading.html>. Cette page décrit, en détails, les changements entre la version 2.2 et la version 2.4.

La configuration d'Apache est située dans le répertoire `/etc/apache2/` :

```
/etc/apache2/
|-- apache2.conf
|   '-- ports.conf
|-- mods-enabled
|   |-- *.load
|   '-- *.conf
|-- conf.d
|   '-- *
'-- sites-enabled
    '-- *
```



Comme toujours sous Linux, les fichiers de configuration sont de simples fichiers "texte" ASCII éditables avec un éditeur de texte comme `vim`, `emacs`, `nano`, ...

La plupart de ces fichiers sont plus ou moins spécifiques à Debian/Ubuntu et permettent de séparer la configuration en plusieurs parties :

- `httpd.conf` est le fichier utilisé par Apache 1, il est conservé vide dans Apache 2 pour assurer la rétrocompatibilité. On ne l'utilisera pas ;
- `apache2.conf` est le fichier principal de configuration et il contient les directives de configuration. Il se charge aussi d'inclure les autres fichiers de configuration ;
- `ports.conf` contient la directive `Listen` qui spécifie les adresses et les ports d'écoutes ;
- `envvars` est utilisé pour définir des variables d'environnement propres à Apache, `magic` est lui utilisé pour déterminer le type de contenu d'un document en regardant les quelques premiers octets de ce contenu ;
- `conf.d` est un répertoire qui contient plusieurs fichiers qui seront analysés par apache. Par exemple, le fichier `charset` permettra de spécifier l'encodage à utiliser par défaut pour tous les fichiers ;
- `mods-available` contient la liste des modules d'apache disponibles ;
- `mods-enabled` celle des modules activés ;
- `sites-available` contient la liste des `vhosts` (*Virtual Host*) disponibles ;
- `sites-enabled` celle des `vhosts` (*Virtual Host*) activés.

L'ensemble de ces fichiers définissent une **configuration par défaut** qui rend le serveur HTTP Apache fonctionnel dès le départ.

Port d'écoute du serveur ?

```
$ cat /etc/apache2/ports.conf | grep -i "listen"
```

```
Listen 80
Listen 443
Listen 443
```

Question 5. À quoi correspondent les ports d'écoute 80 et 443 ?

Racine des documents web du serveur ?

```
$ cat /etc/apache2/sites-enabled/000-default.conf | grep -i "root"
DocumentRoot /var/www/html
```

Modules

Apache est un **serveur modulaire** et la plupart des fonctionnalités sont implémentées dans des modules externes que le programme charge pendant son initialisation. La configuration par défaut n'active que les modules les plus courants et les plus utiles.

La liste complète des modules standards d'Apache est disponible sur le site :

<http://httpd.apache.org/docs/2.4/mod/index.html>

La commande `a2enmod <module>` permet d'activer un nouveau module tandis que `a2dismod <module>` désactive un module. Ces deux programmes ne font rien d'autre que de créer ou supprimer des liens symboliques dans `/etc/apache2/mods-enabled/` pointant vers des fichiers de `/etc/apache2/mods-available/`.

Virtual Hosts

Un **hôte virtuel** est une identité (supplémentaire) assumée par le serveur web. Le Serveur HTTP Apache 2 est capable de gérer simultanément plusieurs arborescences Web grâce à la notion *Virtual Hosts* (hôtes virtuels).

Apache distingue deux types d'hôtes virtuels :

- ceux qui se basent sur l'adresse IP : cette méthode nécessite une adresse IP différente pour chaque site ;
- ceux qui reposent sur le nom DNS du serveur web : celle-ci n'emploie qu'une adresse IP et différencie les sites par le nom d'hôte communiqué par le client HTTP (ce qui ne fonctionne qu'avec la version 1.1 du protocole HTTP, employée par tous les navigateurs web actuels).



La rareté des adresses IPv4 fait en général privilégier cette deuxième méthode. Elle est cependant impossible si chacun des hôtes virtuels a besoin de HTTPS.

Apache fournit quelques commandes utilitaires :

- `a2ensite` : activer un hôte virtuel
- `a2dissite` : désactiver un hôte virtuel

La configuration par défaut d'Apache 2 a déjà activé les hôtes virtuels basés sur le nom grâce à la directive `NameVirtualHost` du fichier `/etc/apache2/sites-enabled/000-default.conf`. Ce fichier décrit en outre un hôte virtuel par défaut qui sera employé si aucun hôte virtuel correspondant n'existe.

On commence par désactiver la configuration par défaut :

```
$ sudo a2dissite 000-default
```



Il faut renseigner le fichier `/etc/hosts` afin d'assurer la résolution de nom (Nom -> Adresse IP) des hôtes virtuels que l'on va créer. Ceci n'est nécessaire que si votre serveur DNS (serveur bind) n'est pas installé et/ou configuré pour vos domaines ou encore, que les noms DNS attribués aux hôtes virtuels sont purement fictifs ou ne vous appartiennent pas.

Chaque hôte virtuel supplémentaire est ensuite décrit par un fichier placé dans le répertoire `/etc/apache2/sites-available/`. Ainsi, la mise en place du domaine `www.bts-sn.lan` se résume à créer le fichier ci-dessous puis à l'activer avec la commande `a2ensite`.

```
$ sudo vim /etc/apache2/sites-available/www.bts-sn.lan.conf
```

```
<VirtualHost www.bts-sn.lan:80>
# ServerName définit le nom utilisé pour le vhost. Mettez le nom de l'hôte du domaine
ServerName www.bts-sn.lan
# ServerAlias définit les autres sous domaines pour lesquels le serveur répondra
#ServerAlias bts-sn.lan *.bts-sn.lan
# ServerAdmin vous permet de spécifier un email à utiliser en cas de problème, sur une page
# d'erreur 404 par exemple
# DocumentRoot définit le dossier racine dans lequel seront stockés les fichiers du site
DocumentRoot /srv/www/html/www.bts-sn.lan
</VirtualHost>
```

On va ensuite créer la racine web du serveur et une page d'accueil personnalisée :

```
$ sudo mkdir -p /srv/www/html/www.bts-sn.lan/

$ sudo cp /var/www/html/index.html /srv/www/html/www.bts-sn.lan/

$ sudo vim /srv/www/html/www.bts-sn.lan/index.html

<html><body><h1>Bienvenue sur www.bts-sn.lan</h1>
<p>En construction ...</p>
</body></html>
```

Il faut maintenant activer le vhost :

```
$ sudo a2ensite www.bts-sn.lan
Enabling site www.bts-sn.lan.
To activate the new configuration, you need to run:
systemctl reload apache2
```

La commande `a2ensite` a créé un lien symbolique du fichier de `sites-available/` vers `sites-enabled/` :

```
$ ls -l /etc/apache2/sites-enabled/
total 0
lrwxrwxrwx 1 root root 37 sept. 10 07:58 www.bts-sn.lan.conf -> ../sites-available/www.bts-sn.lan.conf
```

Si votre **serveur DNS** n'est pas fonctionnel, on assure alors la résolution de noms (ici en local sur le serveur) :

```
$ sudo vim /etc/hosts

...
127.0.1.1      www.bts-sn.lan
```


Pour terminer, il faut demander à Apache de recharger les fichiers de configuration :

```
$ sudo systemctl reload apache2
```

On peut tester l'accès au vhost avec lynx :

```
$ lynx http://www.bts-sn.lan/
```



En cas d'erreur 403 (*Forbidden*), il vous faut évidemment vérifier tout d'abord les droits d'accès (XX5 pour les répertoires et XX4 pour les fichiers). En cas de présence d'une directive `Directory` sur `/srv` dans `/etc/apache2/apache2.conf` pour une version **2.4.6-2** d'Apache, il vous faut appliquer la règle : `Require all granted` au lieu de `Require all denied`.

```
$ sudo vim /etc/apache2/apache2.conf
```

```
...
<Directory /srv/>
    Options Indexes FollowSymLinks
    AllowOverride None
    Require all granted
</Directory>
...
```

```
$ sudo systemctl reload apache2
```

Question 6. Créer un hôte virtuel supplémentaire `serveur.bts-sn.lan`.

Question 7. Modifier l'hôte virtuel `serveur.bts-sn.lan` pour qu'il soit accessible sur le port 8080.

Question 8. Tester maintenant ces trois accès possibles.

```
$ lynx http://serveur.bts-sn.lan:8080/
$ lynx http://www.bts-sn.lan/
$ lynx http://127.0.0.1/
```

Configuration des *Virtual Hosts* :

Comme on vient de le voir, les balises `<VirtualHost>` et `</VirtualHost>` permettent de créer un conteneur soulignant les caractéristiques d'un **hôte virtuel**. Le conteneur `VirtualHost` accepte la plupart des **directives** de configuration.

Les balises `<Directory /path/to/directory>` et `</Directory>` créent un conteneur utilisé pour entourer un groupe de directives de configuration devant uniquement s'appliquer à ce répertoire et à ses sous-répertoires. Toute directive applicable à un répertoire peut être utilisée à l'intérieur de balises `Directory`.

```
$ cat /etc/apache2/apache2.conf
```

```
...
<Directory /var/www/>
    Options Indexes FollowSymLinks
    AllowOverride None
    Require all granted
</Directory>
...
```

La directive **Options** contrôle les fonctionnalités spécifiques du serveur qui sont disponibles dans un répertoire particulier. Elle est suivie d'une liste d'options à activer. L'option **None** désactive toutes les options. Inversement, l'option **All** les active toutes sauf **MultiViews**.

Voici les options existantes :

- **ExecCGI** indique qu'il est possible d'exécuter des scripts CGI.
- **FollowSymLinks** indique au serveur qu'il doit suivre les liens symboliques et donc effectuer la requête sur le fichier réel qui en est la cible.
- **SymlinksIfOwnerMatch** a le même rôle mais impose la restriction supplémentaire de ne suivre le lien que si le fichier pointé appartient au même propriétaire.
- **Includes** active les inclusions côté serveur SSI (*Server Side Includes*). Il s'agit de directives directement intégrées dans les pages HTML et exécutées à la volée à chaque requête.
- **Indexes** autorise le serveur à retourner le contenu du dossier si la requête HTTP pointe sur un répertoire dépourvu de fichier d'index (tous les fichiers de la directive **DirectoryIndex** ayant été tentés en vain).
- **MultiViews** active la négociation de contenu, ce qui permet notamment au serveur de renvoyer la page web correspondant à la langue annoncée par le navigateur web.

La directive **AllowOverride** définit si des **Options** peuvent être annulées par les instructions présente dans un fichier **.htaccess**. Par défaut, aussi bien le répertoire racine que le répertoire **DocumentRoot** sont paramétrés pour ne permettre aucune annulation via **.htaccess**.

La directive **DirectoryIndex** précise la liste des fichiers à essayer pour répondre à une requête sur un répertoire (une URL se terminant par **/**). Le premier fichier existant est appelé pour générer la réponse. S'il ne trouve aucun des fichiers et que **Options Indexes** est paramétrée pour ce répertoire, le serveur génère et renvoie une liste au format HTML, des sous-répertoires et fichiers contenus dans le répertoire (à moins que la fonctionnalité de listage des répertoires ne soit désactivée).

Les directives **Allow from** (autoriser en provenance de) et **Deny from** (refuser en provenance de), qui s'appliquent à un répertoire et à toute l'arborescence qui en est issue, paramètrent les restrictions d'accès.

La directive **Order** contrôle simplement l'ordre dans lequel les directives **Allow** et **Deny** sont analysées.

La directive **Allow** spécifie le client pouvant accéder à un répertoire donné. Le client peut être **all**, un nom de domaine, une adresse IP, une adresse IP partielle, une paire réseau/masque réseau, etc.

La directive **Deny** fonctionne selon le même principe que **Allow**, sauf que cette fois-ci, l'accès est refusé à un client donné.

```
<Directory /var/www/lib>
    Order Deny,Allow
    Deny from All
</Directory>
```



Dans la version 2.4, le contrôle d'accès est assuré via le nouveau module **mod_authz_host**. Bien que le module **mod_access_compat** soit fourni à des fins de compatibilité avec les anciennes configurations, les anciennes directives de contrôle d'accès devront être remplacées par les nouveaux mécanismes d'authentification. Lire : <http://httpd.apache.org/docs/2.4/upgrading.html>. Cette page décrit quelques exemples de contrôle d'accès avec l'ancienne et la nouvelle méthode.

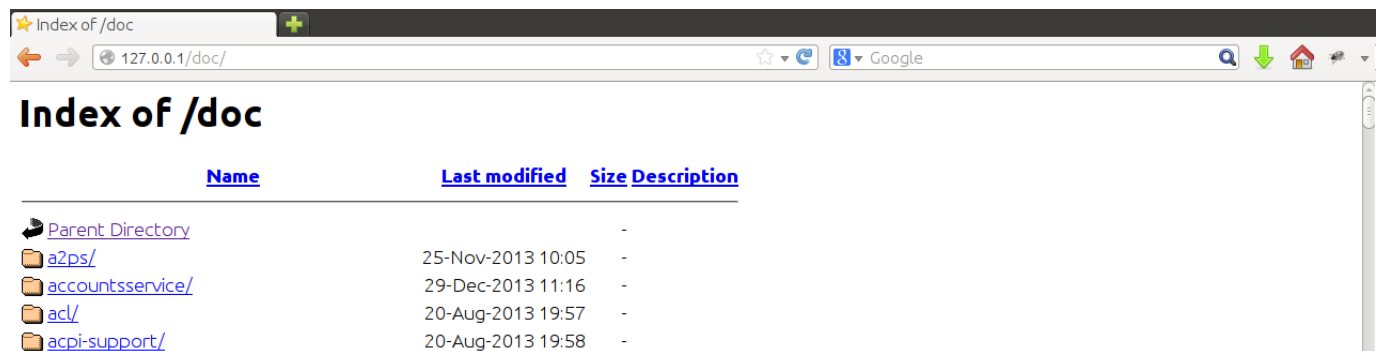
La directive **AccessFileName** nomme le fichier que le serveur doit utiliser pour les informations de contrôle d'accès dans chaque répertoire. La valeur par défaut est **.htaccess**.



Le fichier `.htaccess` contient des directives de configuration d'Apache, prises en compte à chaque fois qu'une requête concerne un élément du répertoire où est il stocké. Sa portée embrasse également les fichiers de toute l'arborescence qui en est issue. La plupart des directives qu'on peut placer dans un bloc `Directory` peuvent également se trouver dans un fichier `.htaccess`.

La directive `Alias` permet d'accéder aux répertoires se trouvant en dehors du répertoire `DocumentRoot`. Toute URL se terminant par un alias sera automatiquement convertie en chemin d'accès vers l'alias.

```
Alias /doc/ "/usr/share/doc/"
<Directory "/usr/share/doc/">
    Options Indexes MultiViews FollowSymLinks
    AllowOverride None
    Order deny,allow
    Deny from all
    Allow from 127.0.0.0/255.0.0.0 ::1/128
</Directory>
```



Mais :



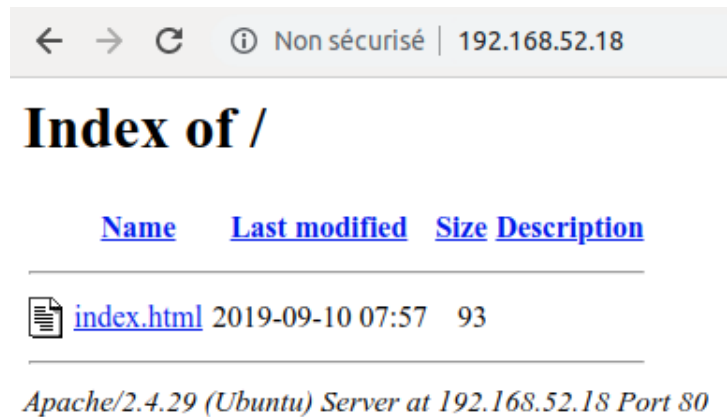
Question 9. En complétant le fichier ci-dessous, configurer votre domaine `www.bts-sn.lan` pour qu'il n'accepte que des requêtes de l'adresse IP votre client, qu'il recherche la page `default.html` par défaut sinon celui-ci retournera le contenu du dossier (sous-répertoires et fichiers) au format HTML. Désactiver les autres options.

```
<VirtualHost www.bts-sn.lan:80>
ServerName www.bts-sn.lan
#ServerAlias bts-sn.lan *.bts-sn.lan
DocumentRoot /srv/www/html/www.bts-sn.lan
<Directory "/srv/www/html/www.bts-sn.lan">
    Options +Indexes +MultiViews +FollowSymLinks -Includes -ExecCGI
    DirectoryIndex ...
    AllowOverride ...
    Order deny,allow
```

```

    Deny from all
    Allow from ...
</Directory>
</VirtualHost>

```



Journalisation

Apache permet la journalisation (*log*) des erreurs et des accès. La journalisation des erreurs se configure avec les directives `ErrorLog` et `LogLevel`. Les réglages par défaut sont fixés dans le fichier `/etc/apache2/apache2.conf` :

```

$ cat /etc/apache2/apache2.conf | grep ErrorLog
// # ErrorLog: The location of the error log file.
// # If you do not specify an ErrorLog directive within a <VirtualHost>
ErrorLog ${APACHE_LOG_DIR}/error.log

```

```

$ cat /etc/apache2/apache2.conf | grep LogLevel
// # LogLevel: Control the number of messages logged to the error_log.
LogLevel warn

```

```

$ cat /etc/apache2/envvars | grep "APACHE_LOG_DIR"
export APACHE_LOG_DIR=/var/log/apache2$SUFFIX

```

L'ensemble des erreurs seront donc journalisées dans le fichier : `/var/log/apache2/error.log`.



Les niveaux disponibles par ordre de criticité décroissante pour la directive `LogLevel` sont : `emerg`, `alert`, `crit`, `error`, `warn`, `notice`, `info`, `debug`. Lorsqu'un niveau particulier est spécifié, les messages de tous les autres niveaux de criticité supérieure seront aussi enregistrés. Voir : <http://httpd.apache.org/docs/2.4/fr/mod/core.html#loglevel>

Apache fournit plusieurs directives pour personnaliser la journalisation des accès. Trois directives sont fournies : `TransferLog` pour créer un fichier journal, `LogFormat` pour définir un format personnalisé, et `CustomLog` pour définir un fichier journal et le format en une seule étape.

```

$ cat /etc/apache2/apache2.conf | grep LogFormat
LogFormat "%v:%p %h %l %u %t \"%r\" %>s %O \"%{Referer}i\" \"%{User-Agent}i\""
    vhost_combined

```

```
LogFormat "%h %l %u %t \"%r\" %>s %O \"%{Referer}i\" \"%{User-Agent}i\"" combined
LogFormat "%h %l %u %t \"%r\" %>s %O" common
LogFormat "%{Referer}i -> %U" referer
LogFormat "%{User-agent}i" agent
```

```
$ cat /etc/apache2/sites-available/000-default.conf | grep CustomLog
CustomLog ${APACHE_LOG_DIR}/access.log combined
```



Ici, `vhost_combined`, `combined`, `common`, et `agent` sont les noms donnés aux formats définis avec la directive `LogFormat` et utilisables avec la directive `CustomLog`. Les formats disponibles sont :

http://httpd.apache.org/docs/2.4/fr/mod/mod_log_config.html#formats

Actuellement, le serveur Apache est configuré pour n'utiliser qu'un seul fichier de *log* pour tous les hôtes virtuels (ce qu'on pourrait changer en intégrant des directives `CustomLog` dans les définitions des hôtes virtuels).

```
$ cat /etc/apache2/conf.d/other-vhosts-access-log
// # Define an access log for VirtualHosts that don't define their own logfile
CustomLog ${APACHE_LOG_DIR}/other_vhosts_access.log vhost_combined
```

Question 10. Configurer votre serveur Apache pour qu'il journalise l'ensemble des messages d'informations sauf ceux de débogage.

Question 11. Configurer vos deux hôtes virtuels pour qu'ils assurent une journalisation des accès dans des fichiers de *log* séparés : `/var/log/apache2/www.bts-sn.lan.log` et `/var/log/apache2/serveur.bts-sn.lan.log`.

Moteur de réécriture d'URL

Apache fournit un moteur de réécriture à base de règles permettant de réécrire les URLs des requêtes à la volée (http://httpd.apache.org/docs/2.4/fr/mod/mod_rewrite.html). Il accepte un nombre illimité de règles, ainsi qu'un nombre illimité de conditions attachées à chaque règle, fournissant ainsi un mécanisme de manipulation d'URL vraiment souple et puissant. Les manipulations d'URL peuvent dépendre de nombreux tests, des variables du serveur, des variables d'environnement, des en-têtes HTTP ou de l'horodatage.

Il faut commencer par activer le module `mod_rewrite` et redémarrer le serveur Apache :

```
$ sudo a2enmod rewrite
```

```
$ sudo systemctl restart apache2
```

Vous trouverez de nombreux exemples d'utilisation courante (et moins courante) dans <http://httpd.apache.org/docs/2.4/fr/rewrite/>.

Exemple : supposons qu'on a récemment renommé la page `default.html` en `index.html` et que l'on désire que les accès à l'ancienne URL restent compatibles. Cependant, on veut que les utilisateurs de l'ancienne URL ne puissent pas reconnaître que les pages ont été renommées. Pour cela, on utilise les directives :

— `RewriteEngine` qui active ou désactive l'exécution du moteur de réécriture.

- RewriteRule qui définit les règles pour le moteur de réécriture en utilisant des expressions rationnelles compatible perl.

```
$ sudo vim /etc/apache2/sites-available/www.bts-sn.lan.conf
```

```
<VirtualHost www.bts-sn.lan:80>
ServerName www.bts-sn.lan
DocumentRoot /srv/www/html/www.bts-sn.lan
<Directory "/srv/www/html/www.bts-sn.lan">
    ...
    RewriteEngine On
    RewriteRule ^default\.html$ index.html
    # réécriture qui permet d'accéder aux pages sans avoir à préciser l'extension
    # page -> page.html
    RewriteCond %{SCRIPT_FILENAME} !-d
    RewriteRule ^([\.\.]+)$ $1.html [NC,L]
</Directory>
</VirtualHost>
```

Question 12. Tester les deux réécritures d'URL mises en oeuvre.

Module PHP

Il faut installer le paquetage libapache2-mod-php5 pour activer le support PHP dans Apache.

```
$ sudo apt-get install libapache2-mod-php
Lecture des listes de paquets... Fait
Construction de l'arbre des dépendances
Lecture des informations d'état... Fait
Les paquets supplémentaires suivants seront installés :
  libapache2-mod-php7.2 libsodium23 php-common php7.2-cli php7.2-common php7.2-json php7.2-
  opcache php7.2-readline
Paquets suggérés :
  php-pear
Les NOUVEAUX paquets suivants seront installés :
  libapache2-mod-php libapache2-mod-php7.2 libsodium23 php-common php7.2-cli php7.2-common
  php7.2-json php7.2-opcache php7.2-readline
0 mis à jour, 9 nouvellement installés, 0 à enlever et 8 non mis à jour.
Il est nécessaire de prendre 3 998 ko dans les archives.
Après cette opération, 17,5 Mo d'espace disque supplémentaires seront utilisés.
...
```

On peut constater que le module PHP a été installé et activé :

```
$ ls -l /etc/apache2/mods-enabled/php*
lrwxrwxrwx 1 root root 29 sept. 10 13:39 /etc/apache2/mods-enabled/php7.2.conf -> ../mods-
  available/php7.2.conf
lrwxrwxrwx 1 root root 29 sept. 10 13:39 /etc/apache2/mods-enabled/php7.2.load -> ../mods-
  available/php7.2.load

$ sudo cat /etc/apache2/mods-enabled/php7.2.load
LoadModule php7_module /usr/lib/apache2/modules/libphp7.2.so
```

```
$ sudo ls -l /usr/lib/apache2/modules/libphp7.2.so
-rw-r--r-- 1 root root 4755752 août 12 19:34 /usr/lib/apache2/modules/libphp7.2.so
```

Pour vérifier que le moteur PHP est fonctionnel, on va créer un script `info.php` à la racine du site `serveur.bts-sn.lan` :

```
$ sudo vim /srv/www/html/serveur.bts-sn.lan/info.php
```

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

Et on teste :

```
$ lynx http://serveur.bts-sn.lan:8080/info.php
```

Question 13. Installer le module PHP. Puis, identifier la version de PHP actuellement installée à partir de l'exécution de la fonction `phpinfo()`.

Ajouter la règle de réécriture d'URL suivante :

```
<Directory "/srv/www/html/serveur.bts-sn.lan">
RewriteEngine On
# page-x.html -> page.php?id=x
RewriteRule ^page-([0-9]+)\.html$ /page.php?id=$1 [L]
</Directory>
```

Question 14. Faire fonctionner le script `page.php` fourni ci-dessous avec la configuration de réécriture d'URL réalisée ci-dessus.

```
<?php
// Script : page.php

// Récupère le paramètre id passé dans l'URL
echo "<h1>Bienvenue sur la page n°".$_REQUEST["id"]."<br />";
echo "<p>En construction ...</p>";

// Affiche la version de PHP : la constante PHP_VERSION contient la version courante de PHP
echo "<p>Version de PHP : ".$_PHP_VERSION."</p>";
?>
```