Installer et configurer un serveur web apache2 sur Debian 12

Préambule :

On s'intéresse dans ce tuto à l'installation d'un serveur web Apache2 sur un serveur Debian 12. Pour commencer ; installez et configurez un serveur Debian en vous aidant si besoin du document « Installer et configurer un serveur Debian 12 ».

Installation d'apache

On commence par vérifier que notre serveur est à jour :

```
administrateur@debian:~$  sudo apt update && sudo apt upgrade
```

Puis on installe le service apache2 :

```
administrateur@debian:~$ sudo apt install apache2
```

Ensuite, on ajuste les règles du firewall (ufw) :

```
administrateur@debian:~$ sudo ufw app list
```

On constate qu'ufw possède un pré-réglage pour WWW, WWW Cache, WWW Full et WWW Secure

- WWW : ouvre le port 80 qui correspond à une connexion normale, non cryptée
- WWW Cache : ouvre le port 8080 parfois utilisé par le cache http et les proxy web
- WWW Full : ouvre les ports 80 et 443
- WWW Secure : ouvre uniquement le port 443 utilisé pour un trafic TLS/SSL cryptée

Nous allons pour le moment uniquement ouvrir le port 80 grâce au pré-réglage WWW :

```
administrateur@debian:~$ sudo ufw allow 'WWW'
Rule added
Rule added (v6)
administrateur@debian:~$ sudo ufw status
Status: active

To                         Action      From
--                         ------      ----
22                         ALLOW       Anywhere
WWW                        ALLOW       Anywhere
22 (v6)                    ALLOW       Anywhere (v6)
WWW (v6)                   ALLOW       Anywhere (v6)
administrateur@debian:~$
```

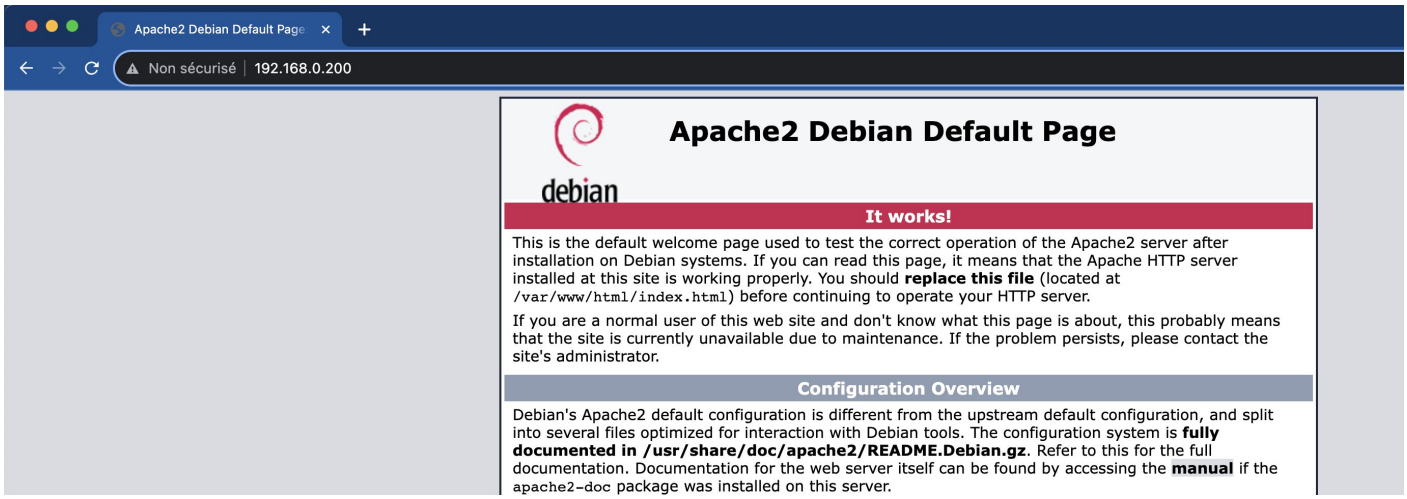
On vérifie ensuite que le service apache2 est bien actif :

```
administrateur@debian:~$ sudo systemctl status apache2
● apache2.service - The Apache HTTP Server
     Loaded: loaded (/lib/systemd/system/apache2.service; enabled; preset: enabled)
     Active: active (running) since Mon 2023-10-09 09:04:07 +04; 9min ago
       Docs: https://httpd.apache.org/docs/2.4/
```

```
  Main PID: 1076 (apache2)
    Tasks: 55 (limit: 4645)
   Memory: 9.4M
      CPU: 167ms
   CGroup: /system.slice/apache2.service
           ├─1076 /usr/sbin/apache2 -k start
           ├─1077 /usr/sbin/apache2 -k start
           └─1078 /usr/sbin/apache2 -k start

oct. 09 09:04:07 debian systemd[1]: Starting apache2.service - The Apache HTTP Server...
oct. 09 09:04:07 debian systemd[1]: Started apache2.service - The Apache HTTP Server.
administrateur@debian:~$
```

On constate que le service est actif et qu'il est paramétré pour se lancer au démarrage de la machine. Nous allons tester notre serveur en ouvrant la page par défaut. Vérifiez l'adresse IP de votre serveur web en lançant ip -a puis ouvrez un navigateur web et mettez l'URL « http://ip-serveur »



Si vous n'arrivez pas à visualiser cette page, vérifiez la connectivité entre votre serveur et la machine du navigateur.

Gérer le processus apache2

Pour piloter le processus apache, vous pouvez utiliser la commande « systemctl <option> apache2 » où <option> peut prendre comme valeur :

- stop : arrêt du serveur web
- start : démarrage du serveur web
- restart : arrêt et redémarrage du serveur web
- reload : après modification de la configuration, relaod force apache à recharger les fichiers de configuration sans fermer les connexions déjà existantes
- disable : si vous ne voulez pas que le serveur web se lance automatiquement au démarrage du serveur
- enable : si vous souhaitez que le serveur web se lance automatiquement au démarrage du serveur. C'est l'option par défaut à l'installation.

Hébergement mutualisé avec apache (les hôtes virtuels)

Apache est un serveur web puissant. Il permet d'héberger plusieurs sites web et il n'est donc pas nécessaire d'associer un serveur à un seul site. Pour héberger plusieurs site, nous allons créer des hôtes virtuels pour chacun d'entre eux.

Nous allons créer un site qui répondra à « site1.btssio.lan »

Par défaut, apache possède un « hôte virtuel » qui distribue les fichiers situés dans « /var/www/html ». Pour héberger d'autres sites, nous allons créer une nouvelle structure dans « /var/www »pour Site1. /var/www/html restera le répertoire par défaut pour les clients qui formuleront des requêtes vers un site qui ne peuvent aboutir.

```
administrateur@debian:~$ sudo mkdir –p /var/www/site1
administrateur@debian:~$ sudo chown –R $USER:$USER /var/www/site1
administrateur@debian:~$ sudo chmod –R 755 /var/www/site1
```

On crée ensuite un fichier nommé index.html dans le dossier créé :

```
administrateur@debian:~$ sudo nano /var/www/site1/index.html
<html>
        <head>
                <title>Bienvenue sur SITE1.BTSSIO.LAN!</title>
        </head>
        <body>
                <h1>Bravo! l'hote virtuel SITE1.BTSSIO.LAN fonctionne correctement</h1>
        </body>
</html>
administrateur@debian:~$
```

Il faut maintenant créer un fichier de configuration pour notre nouvel hôte virtuel :

```
administrateur@debian:~$ sudo nano /etc/apache2/sites-available/site1.conf
<VirtualHost    *:80>
        ServerAdmin webmaster@localhost
        ServerAlias site1.btssio.lan
        DocumentRoot          /var/www/site1
        ErrorLog        ${APACHE_LOG_DIR}/error.log
        CustomLog     ${APACHE_LOG_DIR}/access.log combined
</VirtualHost>

administrateur@debian:~$
```

On autorise ensuite cette configuration avec a2ensite :

```
administrateur@debian:~$ sudo a2ensite site1.conf
```

On désactive le site par défaut :

```
administrateur@debian:~$ sudo a2dissite 000-default.conf
```

On teste ensuite la configuration :

```
administrateur@debian:~$ sudo apache2ctl configtest
Syntax OK
administrateur@debian:~$
```

Et on relance apache :

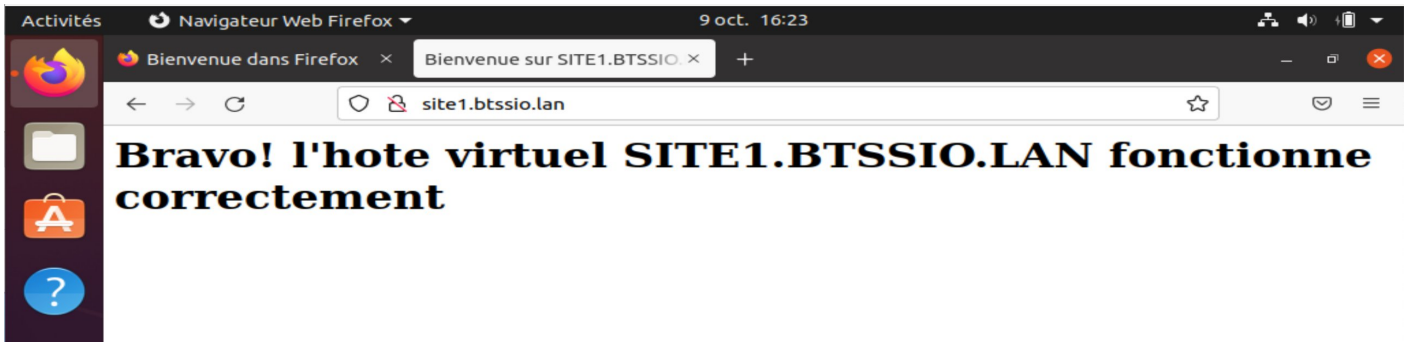
```
administrateur@debian:~$ sudo systemctl restart apache2
```

On va maintenant tester notre nouveau site :

Il faut une machine avec un navigateur web. Pour les besoins de nos tests, on utilise une machine Ubuntu avec environnement graphique et on prend soin au préalable de modifier le fichier /etc/hosts pour que la machine puisse associer site1.btssio.lan à l'adresse IP de notre serveur.

Bien entendu, si vous possédez un serveur DNS sur lequel vous pouvez créer un enregistrement, c'est beaucoup plus facile et n'importe quelle machine aura accès à la résolution IP.

Depuis un navigateur, on teste notre site avec l'URL http://site1.btssio.lan



On peut alors refaire ces manipulations pour un nouvel hôte nommé site2 et le tester à son tour.

Notre serveur distribue maintenant ceux sites web grâce aux hôtes virtuels :

- site1.btssio.lan
- site2.btssio.lan

Sites dynamiques

On va installer un environnement php et un serveur MariaDB pour compléter notre serveur apache2. On crée ainsi un environnement de type LAMP (Linux-Apache-MariaDB-Php).

Installation de MariaDB

Pour installer MariaDB on lance la commande :

```
administrateur@debian:~$ sudo apt install mariadb-server
```

Puis on sécurise notre serveur en lançant :

```
administrateur@debian:~$ sudo mysql_secure_installation
```

Il faut ensuite répondre aux questions :

- Current password for root : attention, il est fait ici référence à l'utilisateur root de mariaDB et non du super utilisateur root du système. Pour l'heure il n'y a pas de mot de passe pour root.
- Switch to unix_socket authentication : no
- Change the root password : Y (choisir un mot de passe pour root, de préférence différent de celui utilisé pour l'utilsateur root du système linux)
- Remove anonymous users : Y
- Disallow root login remotely : Y
- Remove test database and access to it : Y
- Reload privilege tables now : Y

On peut alors tester la connexion à notre serveur. Vous pouvez utiliser la commande mysql ou mariadb. (exit ; pour vous déconnecter) :

- administrateur@debian:~$ sudo mariadb

  En utilisant l'instruction sudo, le système valide votre authentification auprès de mariadb. Vous n'avez donc pas de mot de passe à entrer.

- administrateur@debian:~$ mariadb –uroot -p

  Si vous n'utilisez pas l'instruction sudo, il faut alors vous authentifier auprès de mariadb. La commande est utilisée avec deux option : –uroot signifie que vous cherchez à vous connecter en tant qu 'utilisateur root (au sens de mariadb, pas au sens du système linux). –p signifie que mariadb va vous demander le mot de passe pour valider l'authentifiaction (il faudra utiliser le mot de passe défini lors de la procédure mysql_secure_installation).

```
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 47
Server version: 10.11.4-MariaDB-1~deb12u1 Debian 12

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]>
```

Installation de PHP

Il faut installer les packages suivants

```
administrateur@debian:~$ sudo apt install php libapache2-mod-php php-mysql
```

Puis vérifier la version installée :

```
administrateur@debian:~$ php –v
PHP 8.2.7 (cli) (built: Jun  9 2023 19:37:27) (NTS)
Copyright (c) The PHP Group
Zend Engine v4.2.7, Copyright (c) Zend Technologies
    with Zend OPcache v8.2.7, Copyright (c), by Zend Technologies
administrateur@debian:~$
```

On vérifie que le serveur cherchera le fichier index.php au même titre que les autres extensions. Ouvrir le fichier dir.conf et vérifiez que index.php est bien présent dans la ligne DirectoryIndex.

```
administrateur@debian:~$ sudo nano /etc/apache2/mods-enabled/dir.conf
DirectoryIndex index.html index.cgi index.pl index.php index.xhtml index.htm
```

On teste notre configuration :

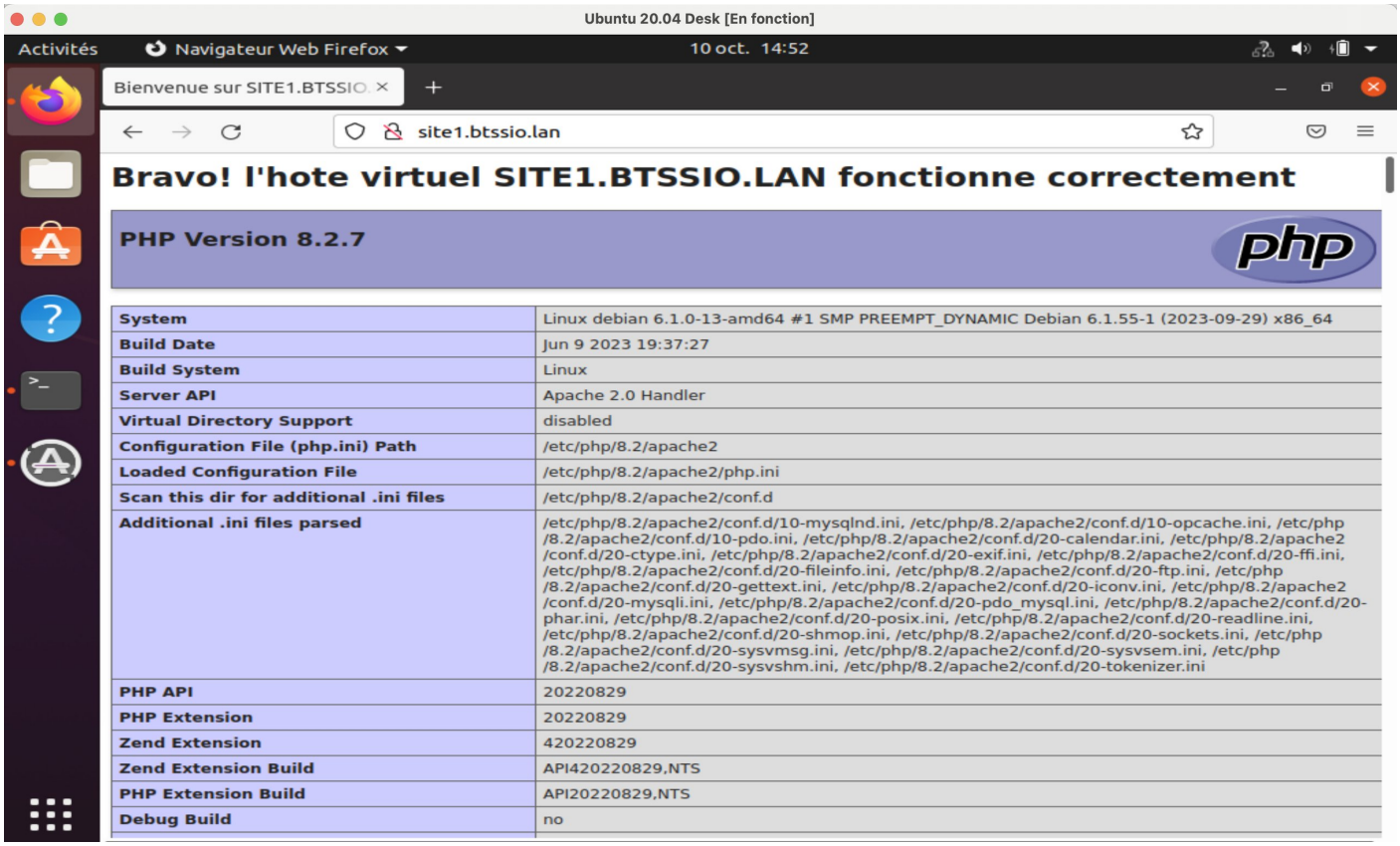
Modifions le fichier index.html de site1 et appelons-le index.php :

```
administrateur@debian:~$ sudo mv /var/www/site1/index.html /var/www/site1/index.php
administrateur@debian:~$ sudo nano /var/www/site1/index.php
<html>
        <head>
                <title>Bienvenue sur SITE1.BTSSIO.LAN!</title>
```

```
        </head>
        <body>
                <h1>Bravo! l'hote virtuel SITE1.BTSSIO.LAN fonctionne correctement</h1>
        </body>
</html>
<?php
phpinfo();
?>
administrateur@debian:~$
```

Retournez maintenant sur un navigateur et accédez à http://site1.btssio.lan

Site1 fonctionne et la page d'information phpinfo() est bien intégrée à la page d'accueil.



System	Linux debian 6.1.0-13-amd64 #1 SMP PREEMPT_DYNAMIC Debian 6.1.55-1 (2023-09-29) x86_64
Build Date	Jun 9 2023 19:37:27
Build System	Linux
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php/8.2/apache2
Loaded Configuration File	/etc/php/8.2/apache2/php.ini
Scan this dir for additional .ini files	/etc/php/8.2/apache2/conf.d
Additional .ini files parsed	/etc/php/8.2/apache2/conf.d/10-mysqlnd.ini, /etc/php/8.2/apache2/conf.d/10-opcache.ini, /etc/php/8.2/apache2/conf.d/10-pdo.ini, /etc/php/8.2/apache2/conf.d/20-calendar.ini, /etc/php/8.2/apache2/conf.d/20-ctype.ini, /etc/php/8.2/apache2/conf.d/20-exif.ini, /etc/php/8.2/apache2/conf.d/20-ffi.ini, /etc/php/8.2/apache2/conf.d/20-fileinfo.ini, /etc/php/8.2/apache2/conf.d/20-ftp.ini, /etc/php/8.2/apache2/conf.d/20-gettext.ini, /etc/php/8.2/apache2/conf.d/20-iconv.ini, /etc/php/8.2/apache2/conf.d/20-mysqli.ini, /etc/php/8.2/apache2/conf.d/20-pdo_mysql.ini, /etc/php/8.2/apache2/conf.d/20-phar.ini, /etc/php/8.2/apache2/conf.d/20-posix.ini, /etc/php/8.2/apache2/conf.d/20-readline.ini, /etc/php/8.2/apache2/conf.d/20-shmop.ini, /etc/php/8.2/apache2/conf.d/20-sockets.ini, /etc/php/8.2/apache2/conf.d/20-sysvmsg.ini, /etc/php/8.2/apache2/conf.d/20-sysvsem.ini, /etc/php/8.2/apache2/conf.d/20-sysvshm.ini, /etc/php/8.2/apache2/conf.d/20-tokenizer.ini
PHP API	20220829
PHP Extension	20220829
Zend Extension	420220829
Zend Extension Build	API420220829,NTS
PHP Extension Build	API20220829,NTS
Debug Build	no

Echanges sécurisés avec apache2 (https)

Les échanges web au travers le protocole http sont efficaces mais peu fiables car les informations circulent en clair à travers les réseaux. Tant que le site est statique, on peut supposer que les informations qui circulent ne sont pas « essentielles » mais dès que le contenu devient dynamique, il est préférable de passer au protocole https (http over TLS/SSL) pour chiffrer les échanges entre le serveur et les clients. L'exemple de l'authentification est frappant : un identifiant et un mot de passe ne doivent pas circuler en clair sur le réseau.

On va donc s'attaquer à la mise en place du protocole https pour nos hôtes virtuels. Https nécessite l'utilisation d'une paire de clef privée/publique et d'un certificat. Dans notre exemple, le certificat sera auto-signé à l'aide de let's encrypt, c'est à dire que notre serveur livrera lui même le certificat qui permet l'authentification de la clef publique utilisée par https. Pour un serveur publique, il serait préférable de faire appel à un organisme de certification (payant) mais l'auto-signature sera suffisante dans notre cas. Notez néanmoins que l'auto-signature n'est pas une sécurité forte et qu'elle laisse la porte ouverte pour les attaques de type « man in the middle ».

On commence par activer mod_ssl, le module apache qui prend en charge le cryptage SSL :

```
administrateur@debian:~$ sudo a2enmod ssl
[sudo] Mot de passe de administrateur :
Considering dependency setenvif for ssl:
Module setenvif already enabled
Considering dependency mime for ssl:
Module mime already enabled
Considering dependency socache_shmcb for ssl:
Enabling module socache_shmcb.
Enabling module ssl.
See /usr/share/doc/apache2/README.Debian.gz on how to configure SSL and create self-signed certificates.
To activate the new configuration, you need to run:
  systemctl restart apache2
administrateur@debian:~$ sudo systemctl restart apache2
administrateur@debian:~$
```

On crée ensuite le certificat SSL

```
administrateur@debian:~$ sudo openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout /etc/ssl/private/apache-selfsigned.key -out /etc/ssl/certs/apache-selfsigned.crt
```

- `openssl` : outil de ligne de commande utilisé pour créer et gérerles certificats, clés et autres fichiers OpenSSL.
- `req -x509` : pour utiliser la gestion des demandes de signature de certificats (CSR) X.509. X.509 est une norme d'infrastructure de clé publique à laquelle SSL et TLS adhèrent pour la gestion des clés et des certificats.
- `-nodes` : indique à OpenSSL de ne pas utiliser l'option de sécurisation du certificat par une phrase de passe. Apache doit être capable de lire le fichier, sans intervention de l'utilisateur. Une phrase de passe empêcherait que cela se produise, puisqu'il faudrait la saisir après chaque redémarrage.
- `-days 365` : cette option fixe la durée pendant laquelle le certificat sera considéré comme valide. Ici, nous l'avons fixée pour un an. De nombreux navigateurs modernes refusent les certificats dont la durée de validité dépasse un an.
- `-newkey rsa:2048` : cette option précise que nous voulons générer un nouveau certificat et une nouvelle clé en même temps. Nous n'avons pas créé la clé nécessaire pour signer le certificat lors d'une étape précédente, nous devons donc la créer en même temps que le certificat. La partie `rsa:2048` lui demande de fabriquer une clé RSA de 2048 bits.
- `-keyout` : indique à OpenSSL où placer le fichier de clé privée généré
- `-out` : indique à OpenSSL où placer le certificat créé.

On remplit ensuite les champs tels que sont demandés :

```
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
```

```
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:fr
State or Province Name (full name) [Some-State]:reunion
Locality Name (eg, city) []:stjoseph
Organization Name (eg, company) [Internet Widgits Pty Ltd]:btssio
Organizational Unit Name (eg, section) []:btssio
Common Name (e.g. server FQDN or YOUR name) []:site1.btssio.lan              D
Email Address []:
```

Le certificat et les deux clefs sont créés.

Il nous reste à configurer apache pour que l'hôte virtuel site1 utilise SSL :

```
administrateur@debian:~$ sudo nano /etc/apache2/sites-available/site1.conf
```

Avant de tester notre connexion https, il faut autoriser les communications sur le port 443 du serveur (on laissera pour l'instant encore le port 80 pour site2) :

```
administrateur@debian:~$ sudo ufw allow 'WWW Full'
administrateur@debian:~$ sudo systemctl reload apache2
```

Depuis un navigateur web on constate que pour l'URL :

- http://site2.btssio.lan est servi correctement
- http://site1.btssio.lan n'est plus servi et nous obtenons la page par défaut
- https://site1.btssio.lan est servi correctement à travers le protocole https

On peut forcer la redirection du protocole http vers https pour site1. Pour se faire, on ouvre le fichier de configuration :

```
administrateur@debian:~$ sudo nano /etc/apache2/sites-available/site1.conf
<VirtualHost    *:443>
        ServerAdmin webmaster@localhost
        ServerAlias site1.btssio.lan

        DocumentRoot        /var/www/site1
        SSLEngine on
        SSLCertificateFile /etc/ssl/certs/apache-selfsigned.crt
        SSLCertificateKeyFile /etc/ssl/private/apache-selfsigned.key
        ErrorLog        ${APACHE_LOG_DIR}/error.log
        CustomLog    ${APACHE_LOG_DIR}/access.log combined
</VirtualHost>
<VirtualHost    *:80>
        ServerName site1.btssio.lan
        Redirect / https://site1.btssio.lan/
</VirtualHost>
administrateur@debian:~$ sudo apachectl configtest
Syntax OK
administrateur@debian:~$ sudo systemctl reload apache2.service
administrateur@debian:~$
```

On constate alors que l'accès à http://site1.btssio.lan redirige vers https://site1.btssio.lan