



Introduction

A- Installation de apache2

- 1- Mettre à jour la machine
- 2- Renommer la machine apache2
- 3- Installation du paquet apache2
- 4- Test de l'installation
- 5- Comprendre l'arborescence des fichiers d'apache2 Localiser les fichiers
 - a- Explorer le fichier apache2.conf
 - b- le contexte directory
- 6- Les mods d'Apache2 et leur gestion le processus
 - a- Les trois mods apache2 mpm_event, mpm_worker et mpm_perfork
 - b- Activation et désactivation manuelle des mods
 - i- Activation et désactivation manuelle des mod mpm
 - ii- Activation et désactivation de mods mpm avec script
 - iii- Détermination du pid des processus

B- Création d'hôte virtuelle

- 1- Création du répertoire de base et de la page d'accueil index.html
- 2- Création d'hôte **virtuelle** par nom
- 3- Création d'hôte **virtuelle** par adresse IP
- 4- Changement de ports
- 5- Utilisation du PHP dans le site
 - a- Implémentation du mod-PHP
 - b-** Implémentation du FPM PHP

C- Sécurisation d'apache2

- 1- Sécurisation apache en masquant sa version et l'os utilisé.
- 2- Sécurisation par SSL
- 3- Sécurisation des répertoires
 - a. Utilisation des Virtuals host
 - b. Utilisation du fichier .htaccess

Introduction

Apache est le nom de la fondation qui a développé le serveur *Apache httpd*, elle a développé en premier le serveur web httpd mais aussi d'autres projets à la suite.

Apache existe depuis plus de 20 ans c'est le serveur web le plus utilisé au monde. Son principal concurrent est un autre logiciel libre appelé *Nginx*.

Les parts de marché sont donnée par le site suivant : <https://fr.hostadvice.com/marketshare/>

Le site officiel de apache2 est <https://httpd.apache.org/>

A- Installation de apache2

1- Mettre à jour la machine

```
root@ares:~# apt update
```

```
root@ares:~# apt upgrade
```

2- Donner un nom à la machine apache2

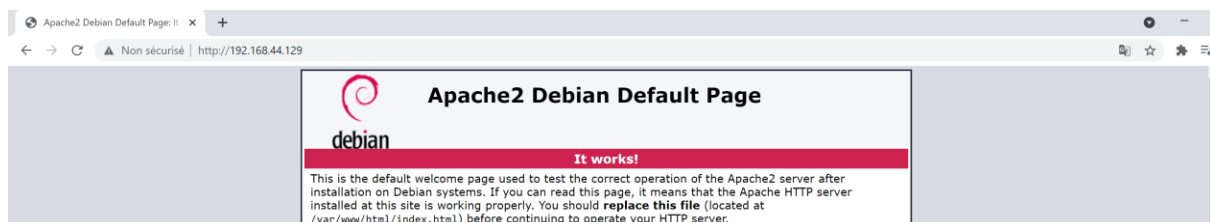
```
root@ares:~# hostnamectl set-hostname apache2
```

3- Installation du paquet apache2

```
root@ares:~# apt install apache2 -y
Lecture des listes de paquets... Fait
Construction de l'arbre des dépendances... Fait
```

4- Test de la connexion apache2

Sur votre machine physique tapez l'adresse ip de votre machine apache



5- Comprendre l'arborescence des fichiers d'apache2 Localiser les fichiers

On va explorer les différents répertoires qui hébergent apache2, avec la commande `whereis` on va identifier ces répertoires.

```
root@apache:~# whereis apache2
apache2: /usr/sbin/apache2 /usr/lib/apache2 /etc/apache2 /usr/share/apache2 /usr/share/man/man8/apache2.8.gz
```

/usr/sbin/apache2 il s'agit ici de l'exécutable dans le répertoire sbin nous avons des binaires système comme son nom l'indique

```
root@apache2:/usr/sbin# ls a2* *apache*
a2disconf a2dismod a2dissite a2enconf a2enmod a2ensite a2query apache2 apache2ctl apachectl
```

/usr/lib/apache2 lib pour bibliothèque

/etc/apache2, est le répertoire où se trouve les fichiers de configuration d'apache2

/usr/share/apache2, des fichiers partagés

/usr/share/man/man8/apache2.8.gz la documentation, obtenu avec la commande man

Le fichier **apache2.conf** nous donne une image de notre arborescence et de la configuration d'apache2.

```
root@apache2:/etc/apache2# cat apache2.conf
# It is split into several files forming the configuration hierarchy outlined
# below, all located in the /etc/apache2/ directory:
#
#   /etc/apache2/
#   |-- apache2.conf
#   |   |-- ports.conf
#   |-- mods-enabled
#   |   |-- *.load
#   |   |-- *.conf
#   |-- conf-enabled
#   |   |-- *.conf
#   |-- sites-enabled
#   |   |-- *.conf
```

On affiche le contenu de notre répertoire

```
root@apache2:/etc/apache2# ls -al
total 88
drwxr-xr-x  8 root root  4096 18 sept. 20:06 .
drwxr-xr-x 78 root root  4096 18 sept. 20:24 ..
-rw-r--r--  1 root root  7224 12 août 13:51 apache2.conf
drwxr-xr-x  2 root root  4096 18 sept. 20:06 conf-available
drwxr-xr-x  2 root root  4096 18 sept. 20:06 conf-enabled
-rw-r--r--  1 root root  1782  8 août 2020 envvars
-rw-r--r--  1 root root 31063  8 août 2020 magic
drwxr-xr-x  2 root root  4096 18 sept. 20:06 mods-available
drwxr-xr-x  2 root root  4096 18 sept. 20:06 mods-enabled
-rw-r--r--  1 root root   320  8 août 2020 ports.conf
drwxr-xr-x  2 root root  4096 18 sept. 20:06 sites-available
drwxr-xr-x  2 root root  4096 18 sept. 20:06 sites-enabled
```

6- Les processus d'Apache et leur gestion le processus

a- Les trois mods apache2 mpm_event, mpm_worker et mpm_perfork

On utilise la commande **ps aux | grep apach[e]** pour afficher les processus d'apache2

a (all user) pour afficher les processus de tout le monde

u (user) pour montrer le propriétaire du processus

x montrer les processus qui ne sont pas attaché à un terminal

Les trois **modules MPM** (event, worker, prefork) déterminent comment Apache gère les connexions. Il ne peut y avoir qu'un seul module MPM actif à la fois.

Le modules MPM "**event**" est démarré par défaut à l'installation d'apache2.

```

root@apache:~# ps aux | grep apach[e]
root      532  0.0  0.2  8572  5040 ?        Ss   09:17   0:02 /usr/sbin/apache2 -k start
www-data  623  0.0  0.3  754740  6064 ?        SL   09:17   0:00 /usr/sbin/apache2 -k start
www-data  624  0.0  0.3  754740  6064 ?        SL   09:17   0:00 /usr/sbin/apache2 -k start
root@apache:~#

root@apache2:~# ps aux --forest | grep apach[e]
root      2591  0.0  0.2  6628  4572 ?        Ss   10:32   0:00 /usr/sbin/apache2 -k start
www-data  2595  0.0  0.3  752748  6304 ?        SL   10:32   0:00 \_ /usr/sbin/apache2 -k start
www-data  2596  0.0  0.5  752748  10384 ?       SL   10:32   0:00 \_ /usr/sbin/apache2 -k start

```

En activant soit le module event ou worker on remarque qu'il y'a 3 processus :

- Le premier processus qui tourne avec le compte root :

```

root      552  0.0  0.4  11496  8528 ?        Ss   09:27   0:00 /usr/sbin/apache2 -k start

```

S = (interruptible sleep) le processus est en exécution et en attente d'un événement Il y a plusieurs threads à l'intérieur de ce processus maître
s = indique que le processus est master

- Les deux autres processus sont dans le même état mais avec un statut qui est en **l** = **(multi-threaded)** ils tournent avec un compte www-data pour des raisons de sécurité, le rôle de ces deux processus répondront aux demandes des clients

```

www-data  750  0.0  0.3  757616  7324 ?        SL   09:27   0:00 /usr/sbin/apache2 -k start
www-data  751  0.0  0.3  757616  7324 ?        SL   09:27   0:00 /usr/sbin/apache2 -k start

```

En lançant le programme plusieurs fois, des processus enfant vont répondre aux appels http des clients en même temps chaque processus a plusieurs threads vont gérer les appels des clients

Apache va charger les nouveaux processus de façon à répondre à une demande qui augmente en même temps chaque processus aura plusieurs threads afin de répondre aux demandes client.

- Si on a un **seul processus** comportant beaucoup de threads ça peut être mal géré.
- Si on a des processus qui sont **single-threaded** on doit en ouvrir d'autres ce qui peut causer une consommation excessive de mémoire Donc apache utilise les deux méthodes, il s'agit d'une **version hybride** qui essaie de gérer au mieux une demande croissante ou fluctuante de pages web.

Au regard ce qui a été dit au-dessus apache peut utiliser 3 mods :

- **Worker** : Ce module permet à chaque processus fils d'Apache de gérer plusieurs connexions dans des "*threads*". Ce mode est performant mais avec des applications web qui ne gèrent pas bien ces threads il peut se montrer moins efficace.
- **Event** : c'est le module le plus récent et le mieux optimisé. C'est une amélioration de mpm_worker qui permet de mieux gérer les connexions **keepalive** (voir apache.conf) qui permettent de faire plusieurs requêtes HTTP à travers une même connexion.

L'avantage de **mpm_events** la version améliorée de **mpm_worker** c'est sa gestion du keepalive.

Un thread spécifique à la gestion du keepalive va permettre de diminuer le nombre de threads qui vont rester en keepalive c'est un des avantages de mpm_events qui permet une optimisation en termes

- D'occupation de process
- D'occupation mémoire

- **Prefork** : Ce module crée un processus fils pour chaque nouvelle connexion, il est fiable mais il est gourmand en mémoire donc en termes de performance ce n'est pas l'Idéal.
Pour vérifier quel mode est chargé pour gérer le connexion clients on va dans **/etc/apache2/mods-enabled**, on constate que le mod mpm_event est démarrée par défaut

```
root@apache2:/etc/apache2# ls mods-enabled/
access_compat.load  authn_file.load  autoindex.load  env.load  mpm_event.load  setenvif.conf
alias.conf          authz_core.load  deflate.conf     filter.load  negotiation.conf  setenvif.load
alias.load          authz_host.load  deflate.load     mime.conf   negotiation.load  status.conf
auth_basic.load     authz_user.load  dir.conf        mime.load   reqtimeout.conf  status.load
authn_core.load     autoindex.conf  dir.load        mpm_event.conf  reqtimeout.load
```

Ou plus simplement avec la commande a2query -M

```
root@www:~# a2query -M
worker
```

Pour arrêter un mode et en démarrer un autre on utilise les scripts créer pour ce but ou manuellement en utilisant les liens symboliques, dans cette démonstration je vais arrêter le module mpm_event et démarrer le module mpm_perfork manuellement après j'utilise les scripts créés à cet effet ce qui peut s'avérer plus simple et pratique.

b- Activation et désactivation des mods mpm

i- Activation et désactivation manuelle des mod mpm

- Je supprime le module mpm_event

```
(root@apache2)~# cd /etc/apache2/mods-enabled/
(root@apache2)~/etc/apache2/mods-enabled# rm mpm_event.conf mpm_event.load
```

- Je redémarre le service apache2 et je vérifie que les processus sont arrêter

```
(root@apache2)~/etc/apache2/mods-enabled# service apache2 restart
Job for apache2.service failed because the control process exited with error code.
See "systemctl status apache2.service" and "journalctl -xe" for details.
(root@apache2)~/etc/apache2/mods-enabled# ps aux --forest | grep apach[e]
```

- Je crée les liens symboliques, je redémarre le service apache2 et je vérifie que les processus sont démarrés


```

[root@apache2]~[/etc/apache2/mods-enabled]
# ln -s ../mods-available/mpm_prefork.conf

[root@apache2]~[/etc/apache2/mods-enabled]
# ln -s ../mods-available/mpm_prefork.load

[root@apache2]~[/etc/apache2/mods-enabled]
# systemctl restart apache2

[root@apache2]~[/etc/apache2/mods-enabled]
# ps aux --forest | grep apach[e]
www-data 2685 0.0 0.0 3432 172 ? Ss 10:32 0:00 /usr/bin/htcacheclean -d 120 -p /var/cache/apach
00M -n
root 3550 0.0 0.2 6344 4296 ? Ss 11:27 0:00 /usr/sbin/apache2 -k start
www-data 3551 0.0 0.1 6604 3644 ? S 11:27 0:00 \_ /usr/sbin/apache2 -k start
www-data 3552 0.0 0.1 6604 3644 ? S 11:27 0:00 \_ /usr/sbin/apache2 -k start
www-data 3553 0.0 0.1 6604 3644 ? S 11:27 0:00 \_ /usr/sbin/apache2 -k start
www-data 3554 0.0 0.1 6604 3644 ? S 11:27 0:00 \_ /usr/sbin/apache2 -k start
www-data 3555 0.0 0.1 6604 3644 ? S 11:27 0:00 \_ /usr/sbin/apache2 -k start

```

ii- Activation et désactivation de mods mpm avec script

Maintenant on va faire l'inverse en désactivant prefork et en activant event mais en utilisant les scripts, on remarquera que c'est plus facile et pratique

```

[root@apache2]~[/etc/apache2/mods-enabled]
# a2dismod mpm_prefork
Module mpm_prefork disabled.
To activate the new configuration, you need to run:
systemctl restart apache2

[root@apache2]~[/etc/apache2/mods-enabled]
# systemctl restart apache2
Job for apache2.service failed because the control process exited with error code.
See "systemctl status apache2.service" and "journalctl -xe" for details.

[root@apache2]~[/etc/apache2/mods-enabled]
# a2enmod mpm_event
Considering conflict mpm_worker for mpm_event:
Considering conflict mpm_prefork for mpm_event:
Enabling module mpm_event.
To activate the new configuration, you need to run:
systemctl restart apache2

[root@apache2]~[/etc/apache2/mods-enabled]
# systemctl restart apache2

[root@apache2]~[/etc/apache2/mods-enabled]
# ps aux --forest | grep apach[e]
www-data 2685 0.0 0.0 3432 172 ? Ss 10:32 0:00 /usr/bin/htcacheclean -d 120 -p /var/cache/apache2/
00M -n
root 3621 0.0 0.2 6628 4676 ? Ss 11:34 0:00 /usr/sbin/apache2 -k start
www-data 3622 0.0 0.3 752748 6316 ? SL 11:34 0:00 \_ /usr/sbin/apache2 -k start
www-data 3623 0.0 0.3 752748 6316 ? SL 11:34 0:00 \_ /usr/sbin/apache2 -k start

```

iii- Détermination du pid des processus

Pour déterminer le pid des processus apache2 on peut utiliser les trois méthodes

```

root@apache2:~# pidof apache2
1556 1555 1554

```

```

root@apache:/var/run/apache2# cat apache2.pid
532

```

```

root@apache2:~# ps aux --forest | grep apach[e]
root 1554 0.0 0.3 11372 7416 ? Ss 10:41 0:00 /usr/sbin/apache2 -k start
www-data 1555 0.0 0.5 757624 11088 ? SL 10:41 0:00 \_ /usr/sbin/apache2 -k start
www-data 1556 0.0 0.5 757624 11088 ? SL 10:41 0:00 \_ /usr/sbin/apache2 -k start

```

B- Création du site

Un serveur peut héberger plusieurs sites web, on peut utiliser deux méthodes pour atteindre ce but en utilisant les Virtuals hosts ou en affectant une adresse IP par site web

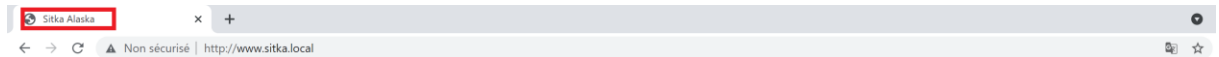
1- Création du répertoire de base et de la page d'accueil index.html

Je crée un répertoire **Sitka** qui va héberger mes pages web, en même temps je copie le fichier index.html du répertoire html dans le répertoire **Sitka**

```
root@apache2:~# mkdir /var/www/sitka/
root@apache2:/var/www# cp html/index.html sitka/
```

Modifier la page de sorte que :

Le titre sur le navigateur soit **Sitka Alaska**



Dans la page html ce qui est indiqué ci-dessous



2- Création de site par nom de domaine

a- Copier le fichier conf du site par défaut d'apache 000-default.conf en sitka.conf

```
root@apache2:/etc/apache2/sites-available# cp 000-default.conf sitka.conf
```

b- Maintenant on va configurer le fichier sitka.conf

La configuration d'un virtual host est comprise dans les balises `<VirtualHost *:80></VirtualHost>`.
*:80 de la balise ouvrante indique que le virtual host utilise le port 80 sur toutes les interfaces.

les options utilisées dans le Virtual host:

- **ServerName**: Il existe un seul **ServerName** par **VirtualHost**, c'est Le nom de l'hôte pour lequel cette configuration sera utilisée
- **ServerAlias**: Il n'y a qu'un **ServerName**, mais on peut compléter par des **ServerAlias**
- **ServerAdmin**: Est l'adresse mail de contact et qui va être affichée sur certains messages d'erreurs
- **DocumentRoot**: L'endroit du stockage des fichiers du site web

Une configuration minimale pourrait s'arrêter là, les autres directives sont des options supplémentaires.

- **CustomLog** : demande de stocker les fichiers de logs au format combined dans un fichier à part plutôt que dans le fichier de logs général
- **ErrorLog** : demande de stocker aussi les logs d'erreur dans un fichier à part mais leur format est fixe
- **<Directory />** : dans la balise **Directory**, vous pouvez définir des règles qui s'appliquent uniquement au contenu d'un répertoire, dans le fichier **apache2.config** on peut trouver des exemples :
 - **ExecCGI**: l'exécution de scripts à l'aide du module CGI est permise

- FollowSymlinks: le serveur va suivre les liens symboliques. Cette option est la seule active par défaut.
- Includes : les inclusions côté serveur à l'aide du module mod_include sont autorisées
- Indexes : si aucun fichier par défaut type index.html n'existe, le module mod_autoindexes va présenter une liste des fichiers et répertoires formatée par Apache
- AllowOverride None : cette directive placée dans <Directory /> indique qu'aucune option ne peut être surchargée par les options qui seraient contenues dans un fichier appelé .htaccess placé dans cette arborescence. On peut préciser l'option qui peut être chargées ou utiliser All pour autoriser toutes les options, dans ce cas, les options du fichier .htaccess auront la priorité sur celles du Virtual Host.

```
root@apache2:/etc/apache2/sites-available# vim sitka.conf
```

```
<VirtualHost *:80>

    ServerName www.sitka.local
    ServerAlias *.sitka.local sitka.local
    ServerAdmin admin@www.sitka.local
    DocumentRoot /var/www/sitka

    ErrorLog ${APACHE_LOG_DIR}/sitka_error.log
    CustomLog ${APACHE_LOG_DIR}/sitka_access.log combined

</VirtualHost>
```

c- Vérification de la syntaxe de mon fichier de configuration sitka.conf

```
root@apache2:~# apachectl -t
AH00558: apache2: Could not reliably determine the server's fully qualified domain name, using 192.168.10.250. Set the 'ServerName' directive globally to suppress this message
Syntax OK
```

d- Activation du site Sitka

```
root@apache2:~# a2ensite sitka.conf
Enabling site sitka.
To activate the new configuration, you need to run:
systemctl reload apache2
```

Je charge ma configuration

```
root@apache2:~# systemctl reload apache2
```

La résolution du nom www.sitka.local doit se faire avec un serveur DNS pour nos test on peut se contenter d'utiliser le fichier hosts, sans la déclaration de www.sitka.local dans un DNS ou le fichier host on ne pourra ni faire un ping ni accéder au site web sur un navigateur



Nom	Modifié le	Type	Taille
hosts	03/11/2021 16:49	Fichier	2 Ko

Avant de faire le test sur un navigateur on va d'abord faire le test du ping


```
PS C:\> ping www.sitka.local

Envoi d'une requête 'ping' sur www.sitka.local [192.168.44.129] avec 32 octets de données :
Réponse de 192.168.44.129 : octets=32 temps<1ms TTL=64
Réponse de 192.168.44.129 : octets=32 temps=1 ms TTL=64
Réponse de 192.168.44.129 : octets=32 temps=1 ms TTL=64
Réponse de 192.168.44.129 : octets=32 temps<1ms TTL=64

Statistiques Ping pour 192.168.44.129:
    Paquets : envoyés = 4, reçus = 4, perdus = 0 (perte 0%),
    Durée approximative des boucles en millisecondes :
        Minimum = 0ms, Maximum = 1ms, Moyenne = 0ms
```

Le ping fonctionne donc on va maintenant tester notre site avec un navigateur



3- Création de site par adresse IP

On crée des adresses virtuelles en rajoutant la configuration suivante dans mon fichier interfaces ici j'ai rajouté deux adresses virtuelles 192.168.44.160 et 192.168.44.161

```
root@apache2:~# vim /etc/network/interfaces
```

```
# The primary network interface
allow-hotplug ens33
iface ens33 inet dhcp

#rajout de cartes virtuelles

up ip addr add 192.168.44.160/24 dev ens33 label ens33:0
down ip addr del 192.168.44.160/24 dev ens33 label ens33:0

up ip addr add 192.168.44.161/24 dev ens33 label ens33:1
down ip addr del 192.168.44.161/24 dev ens33 label ens33:1
```

J'active et je désactive ma carte

```
root@apache2:~# ifdown ens33
```

```
root@apache2:~# ifup ens33
```

Je vérifie avec ip ad que mes deux adresses IP virtuelles sont actives

```
(root@apache2)~# ip ad
# ip ad
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 00:0c:29:8b:b6:95 brd ff:ff:ff:ff:ff:ff
    altname enp2s1
    inet 192.168.44.129/24 brd 192.168.44.255 scope global dynamic ens33
        valid_lft 1735sec preferred_lft 1735sec
    inet 192.168.44.160/24 scope global secondary ens33:0
        valid_lft forever preferred_lft forever
    inet 192.168.44.161/24 scope global secondary ens33:1
        valid_lft forever preferred_lft forever
    inet6 fe80::20c:29ff:fe8b:b695/64 scope link
        valid_lft forever preferred_lft forever
```

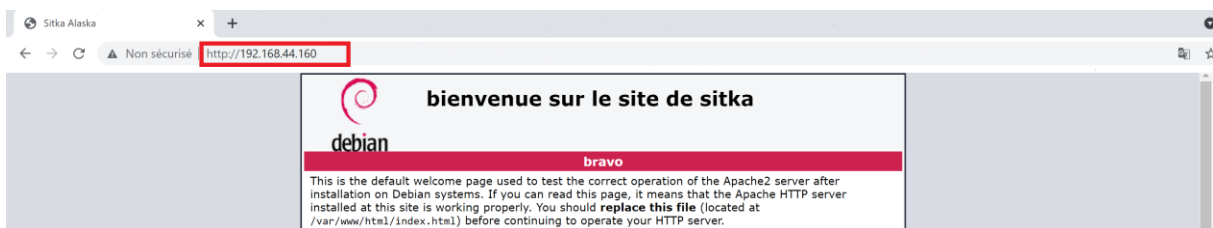
Dans le fichier sitka.conf je précise à mon serveur apache2 qu'au lieu d'écouter sur toutes les adresses il faut maintenant juste écouter sur l'interface **192.168.44.160**

```
<VirtualHost 192.168.44.160:80>
    ServerName www.sitka.local
    ServerAlias *.sitka.local sitka.local
    ServerAdmin admin@www.sitka.local
    DocumentRoot /var/www/sitka

    ErrorLog ${APACHE_LOG_DIR}/sitka_error.log
    CustomLog ${APACHE_LOG_DIR}/sitka_access.log combined
</VirtualHost>
```

```
root@apache2:~# apachectl -t
Syntax OK
```

```
root@apache2:~# apache2ctl graceful
```



4- Changement du port par défaut de notre site

Le serveur web écoute par défaut sur le port 80 en http et 443 en https, on va assigner à notre serveur le port 44

```
Listen 80
Listen 44

<IfModule ssl_module>
    Listen 443
</IfModule>

<IfModule mod_gnutls.c>
    Listen 443
</IfModule>
```

Dans le fichier de configuration du site sitka.conf on va indiquer le port sur le quel apache doit écouter.

```
<VirtualHost 192.168.44.160:44>
    ServerName www.sitka.local
    ServerAlias *.sitka.local sitka.local
    ServerAdmin admin@www.sitka.local
    DocumentRoot /var/www/sitka

    ErrorLog ${APACHE_LOG_DIR}/sitka_error.log
    CustomLog ${APACHE_LOG_DIR}/sitka_access.log combined
</VirtualHost>
```

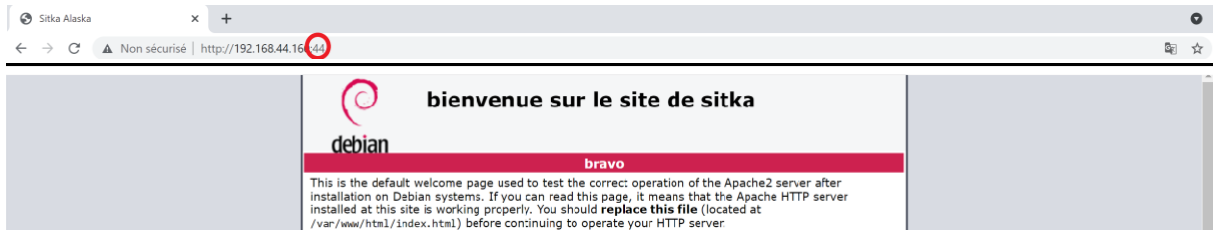
On vérifie la syntaxe

```
root@apache2:~# apachectl -t
Syntax OK
```

On redémarre apache2

```
root@apache2:~# apache2ctl graceful
```

On teste notre nouvelle configuration avec un navigateur.



5- Utilisation du PHP dans le site

Il existe deux moyens pour utiliser PHP dans un site web **mod_php** et **FPM** :

- i- **mod_php** c'est un module qui permet à Apache d'exécuter les scripts PHP. Malheureusement **mod_php** possède de nombreux inconvénients.

Inconvénient de mod_php

- mod_php ne fonctionne qu'avec Apache puisque c'est un module Apache.
- mod_php fonctionne avec mod_prefork qui va générer un processus par requête http en terme de performance ce n'est pas top.
- Avec mod_php, Apache partage le même processus que PHP par conséquent la mémoire utilisée est partagée entre eux, Cela pose des problèmes de sécurité si une personne trouve une faille de sécurité.
- Avec mod_php, on ne peut utiliser qu'une seule version de PHP à la fois.
- Tout fonctionne avec un utilisateur unique : www-data.

- ii- **FPM** (Fast CGI Process Manager) qu'il faut utiliser car plus performant que **mod_php**.

Avantage de FPM

- FPM est un service indépendant d'Apache car il tourne tout seule.
- FPM utilise le protocole Fast CGI pour exécuter les scripts. C'est un protocole qui est géré par plusieurs serveurs web.
- On peut donc avoir plusieurs versions de PHP en même temps. FPM peut gérer plusieurs pools. Chaque pool aura son propre utilisateur et groupe, ses propres permissions et dossiers.
- FPM est capable de travailler avec des sockets IPv4 ou IPv6 ou UDS.
- Il existe plusieurs modes de fonctionnement dynamique de FPM
- FPM s'adapte à la charge en augmentant le nombre de si la charge augmente. Si la charge diminue, les processus sont arrêtés.

a- Implémentation du mod-PHP

On commencera par installer mod-PHP

```
root@apache2:~# apt install libapache2-mod-php
```

Attention à la fin de l'installation on a le message suivant pour nous prévenir que libapache2-mod-php7.4 n'a pas pu activer :

- Le mod mpm_prefork
- Le mod php7.4

Donc il faut activer ces deux mods

```
apache2_switch_mpm prefork: Has been disabled manually, not changing  
libapache2-mod-php7.4: Could not switch to prefork MPM, not enabling PHP 7.4
```

- Activation du mod mpm_prefork

```
root@apache2:~# a2dismod mpm_event
```

```
root@apache2:~# a2enmod mpm_prefork
```

- Activation du mod php7.4

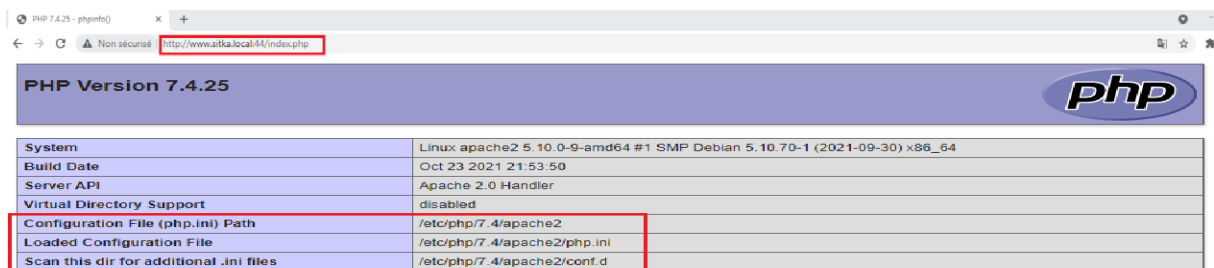
```
root@apache2:~# a2enmod php7.4
```

On redémarre le service apache2

```
root@apache2:/etc/apache2/mods-available# systemctl restart apache2
```

Avant de faire le test on va créer une page index.php, avec le contenu indiqué ci-dessous

```
root@apache2:/var/www/sitka# cat index.php  
<?php  
phpinfo();  
?>
```



PHP Version 7.4.25	
System	Linux apache2 5.10.0-9-amd64 #1 SMP Debian 5.10.70-1 (2021-09-30) x86_64
Build Date	Oct 23 2021 21:53:50
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php/7.4/apache2
Loaded Configuration File	/etc/php/7.4/apache2/php.ini
Scan this dir for additional .ini files	/etc/php/7.4/apache2/conf.d

b- Implémentation du FPM PHP

Tout d'abord il faut désinstaller le paquet libapache2-mod-php7.4 s'il est installé avant d'installer php-fpm

```
root@apache2:~# apt purge libapache2-mod-php7.4 -y
```

```
root@apache2:~# apt autoremove
```

On redémarre le service apache2

```
root@apache2:~# systemctl restart apache2
```

Maintenant on peut commencer l'installation de php-fpm

```
root@apache2:~# apt install php-fpm -y
```

A la fin d'installation on nous demande de démarrer les mods proxy_fcgi et sentenvif ainsi que la conf php7.4-fpm


```
NOTICE: Not enabling PHP 7.4 FPM by default.
NOTICE: To enable PHP 7.4 FPM in Apache2 do:
NOTICE: a2enmod proxy_fcgi setenvif
NOTICE: a2enconf php7.4-fpm
NOTICE: You are seeing this message because you have apache2 package installed.
```

On démarre les mods proxy_fcgi et setenvif

```
root@apache2:~# a2enmod proxy_fcgi setenvif
Considering dependency proxy for proxy_fcgi:
Module proxy already enabled
Module proxy_fcgi already enabled
Module setenvif already enabled
```

On démarre la conf php7.4-fpm

```
root@apache2:~# a2enconf php7.4-fpm
Enabling conf php7.4-fpm.
To activate the new configuration, you need to run:
systemctl reload apache2
```

On rajoute dans le fichier sitka.conf le bloc ci-dessous qui stipule que tout fichier php doit utiliser le socket Unix

```
<FilesMatch \.php$>
    # 2.4.10+ can proxy to unix socket
    SetHandler "proxy:unix:/var/run/php/php7.4-fpm.sock|fcgi://localhost"
</FilesMatch>
```

```
<FilesMatch \.php$>
    # 2.4.10+ can proxy to unix socket
    SetHandler "proxy:unix:/var/run/php/php7.4-fpm.sock|fcgi://localhost"
</FilesMatch>
```

On vérifie quel mod mpm est activé, si c'est le mod prefork qui est activé, alors il faut le désactiver et activer le mod event

```
root@apache2:~# a2query -M
prefork
```

```
root@apache2:~# a2dismod mpm_prefork
```

```
root@apache2:~# a2enmod mpm_event
```

```
root@apache2:~# systemctl reload apache2
```

On vérifie que tous les mods nécessaires pour fpm PHP sont activées

```

root@apache2:~# a2query -m
setenvif (enabled by maintainer script)
env (enabled by maintainer script)
deflate (enabled by maintainer script)
ssl (enabled by site administrator)
authz_host (enabled by maintainer script)
authn_core (enabled by maintainer script)
authz_user (enabled by maintainer script)
status (enabled by maintainer script)
mpm_event (enabled by site administrator)
authn_file (enabled by maintainer script)
filter (enabled by maintainer script)
authz_core (enabled by maintainer script)
autoindex (enabled by maintainer script)
alias (enabled by maintainer script)
negotiation (enabled by maintainer script)
socache_shmcb (enabled by site administrator)
proxy (enabled by site administrator)
auth_basic (enabled by maintainer script)
reqtimeout (enabled by maintainer script)
access_compat (enabled by maintainer script)
dir (enabled by maintainer script)
mime (enabled by maintainer script)
proxy_fcgi (enabled by site administrator)

```

On vérifie que toutes les confs nécessaires pour fpm PHP sont activées

```

root@apache2:~# a2query -c
security (enabled by maintainer script)
serve-cgi-bin (enabled by maintainer script)
sitka_name (enabled by site administrator)
charset (enabled by maintainer script)
localized-error-pages (enabled by maintainer script)
php7.4-fpm (enabled by site administrator)
other-vhosts-access-log (enabled by maintainer script)

```

On fait le test sur un navigateur on remarque que maintenant c'est FPM/Fastcgi qui est activé

PHP 7.4.25 - phpinfo()

Non sécurisé http://www.sitka.local:44/index.php

PHP Version 7.4.25

System	Linux apache2 5.10.0-9-amd64 #1 SMP Debian 5.10.70-1 (2021-09-30) x86_64
Build Date	Oct 23 2021 21:53:50
Server API	FPM/FastCGI
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php/7.4/fpm
Loaded Configuration File	/etc/php/7.4/fpm/php.ini
Scan this dir for additional .ini files	/etc/php/7.4/fpm/conf.d

D- Sécurisation du site

1- Sécurisation apache masquant sa version et l'os utilisé.

Apache envoie par défaut des entêtes HTTP contenant le nom et la version du serveur web ainsi que le système d'exploitation qui héberge apache, ceci peut être problématique car on peut faciliter l'attaque de notre serveur en divulguant ces informations.

En local on peut afficher ces informations avec la commande apt policy apache2

```

root@apache2:~# apt-cache policy apache2
apache2:
  Installé : 2.4.51-1-deb11u1
  Candidat : 2.4.51-1-deb11u1
  Table de version :
  *** 2.4.51-1-deb11u1 500
      500 http://security.debian.org/debian-security bullseye-security/main amd64 Packages
      100 /var/lib/dpkg/status
      2.4.48-3.1+deb11u1 500
      500 http://deb.debian.org/debian bullseye/main amd64 Packages

```

```
root@apache2:~# apachectl -v
Server version: Apache/2.4.51 (Debian)
Server built: 2021-10-07T17:49:44
```

A distance sur une machine linux on peut afficher ces informations avec la commande curl

```
(root@ETANIUM)~# curl -I 192.168.44.129
HTTP/1.1 200 OK
Date: Sun, 07 Nov 2021 10:43:35 GMT
Server: Apache/2.4.51 (Debian)
Last-Modified: Wed, 03 Nov 2021 09:32:45 GMT
ETag: "29cd-5cfd17b19ecc"
Accept-Ranges: bytes
Content-Length: 10701
Vary: Accept-Encoding
Content-Type: text/html
```



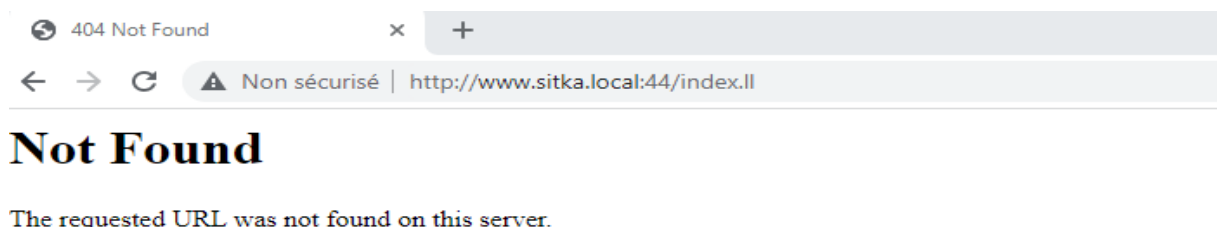
Pour cacher la version d'Apache, il faut changer des paramètres dans le fichier /etc/apache2/conf-enabled/security.conf.

Les paramètres à modifier sont **ServerTokens** et **ServerSignature**, on peut atteindre le même but en rajoutant ces paramètres directement dans le fichier apache2.conf à la fin du fichier.

```
root@apache2:/etc/apache2/conf-enabled# vim security.conf
#ServerTokens OS
ServerTokens Prod
```

```
#ServerSignature On
ServerSignature off
```

```
root@apache2:~# systemctl reload apache2
```



2- Sécurisation par SSL

On vérifie la présence du paquet ssl-cert

```
root@apache:~# dpkg -l ssl-cert
Souhait=inconnU/Installé/suppRimé/Purgé/H=à garder
| État=Non/Installé/fichier-Config/dépaqueté/échec-Config/H=semi-installé/W=attend-traitement-déclen
|/ Err?=(aucune)/besoin Réinstallation (État,Err: majuscule=mauvais)
||/ Nom Version Architecture Description
+++=-----+-----+-----+-----+
ii  ssl-cert 1.0.39 all simple debconf wrapper for OpenSSL
Lines 1-6/6 (END)
```

On active le mode ssl

```
root@apache:/etc/apache2/mods-available# a2enmod ssl
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
```

Création d'un fichier pem (Privacy Enhanced Mail (PEM)) contenant un certificat autosigné et une clé privée.

```
root@apache:~# make-ssl-cert /usr/share/ssl-cert/ssleay.cnf /etc/ssl/private/sitka.pem
```

```
root@apache2:~# make-ssl-cert /usr/share/ssl-cert/ssleay.cnf /etc/ssl/private/sitka.pem
```

Configuration d'un certificat SSL

Veuillez indiquer le nom d'hôte à utiliser dans le certificat SSL.
Ce sera le contenu du champ « commonName » du certificat SSL créé.

Nom d'hôte :

www.sitka.local

<Ok> <Annuler>

Configuration d'un certificat SSL

Veuillez indiquer d'éventuels noms d'hôte supplémentaires à utiliser dans le certificat SSL.
Ce sera le contenu du champ « subjectAltName » du certificat SSL créé.

Des entrées multiples doivent être délimitées par des virgules, sans espaces. Ainsi, pour un serveur web qui utilise plusieurs noms DNS, cette entrée devrait ressembler à :

DNS:www.example.com,DNS:images.example.com

Exemple plus complexe comportant un nom d'hôte, un identifiant web (« WebID »), une adresse électronique et une adresse IPv4 :

DNS:example.com,URI:http://example.com/joe#me,email:me@example.com,IP:192.168.7.3

Nom(s) supplémentaire(s) :

URI:http://sitka.local, IP:192.168.44.129

<Ok> <Annuler>

On vérifie la création du fichier pem

```
root@apache2:/etc/ssl/private# ls
c8c3824d.0 sitka.pem ssl-cert-snakeoil.key
```

En affichant sitka.pem on se rend compte s'aperçoit qu'il possède un certificat et une clé privée

La deuxième méthode est l'utilisation des fichiers `.htaccess` afin de définir certains éléments de configuration dans un répertoire. Apache doit relire ces fichiers à chaque requête, ce qui peut avoir un impact négatif sur les performances.

.htaccess est surtout utilisé pour autoriser des utilisateurs non root à gérer certains aspects de notre serveur apache.

a- Utilisation des Virtuals host

On crée le fichier où les mots de passe vont être stockés, en même temps on crée le premier utilisateur que j'appelle kaiser

```
root@apache2:~# htpasswd -c /etc/apache2/.password kaiser
New password:
Re-type new password:
Adding password for user kaiser
```

Je crée le deuxième utilisateur cesar, en enlevant l'option -c pour ne pas écraser le fichier .password

```
root@apache2:~# htpasswd /etc/apache2/.password cesar
New password:
Re-type new password:
Adding password for user cesar
```

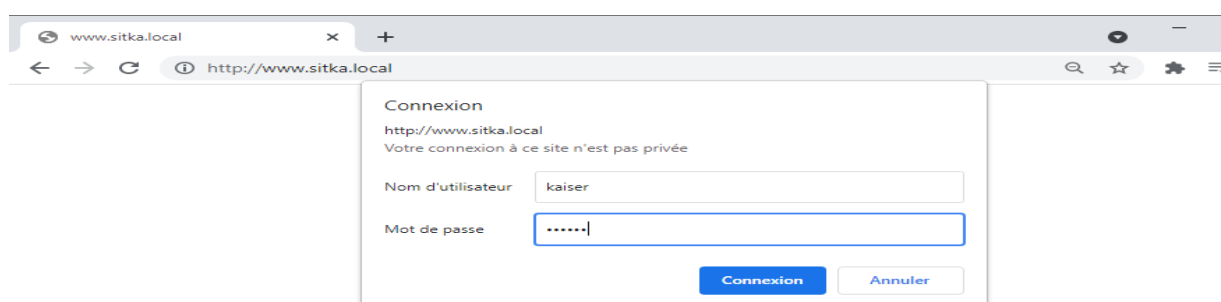
On vérifie la création des comptes utilisateur

```
root@apache2:/etc/apache2# vim .password

kaiser:$apr1$I2WvrU1g$ixe5wt6ZOY44nATtFDV8p/
cesar:$apr1$i0zRx0oi$ullh9Qw7WvqEQWU/waEwH1
```

On rajoute les directives directory dans les virtuels host

```
<Directory /var/www/sitka>
    AuthType Basic
    AuthName "utilisateur authentifié"
    AuthBasicProvider file
    AuthUserFile "/etc/apache2/.password"
    # Require ip 192.168.44.1
    Require valid-user
</Directory>
```



b- Utilisation du fichier .htaccess

Tout d'abord il faut changer directive `AllowOverride None` à

`AllowOverride all`

Ceci autorise l'utilisation d'un fichier .htaccess dans le répertoire /var/www/

```
root@apache2:/etc/apache2# vim apache2.conf |  
  
<Directory /var/www/>  
    Options Indexes FollowSymLinks  
    AllowOverride all  
    Require all granted  
</Directory>
```

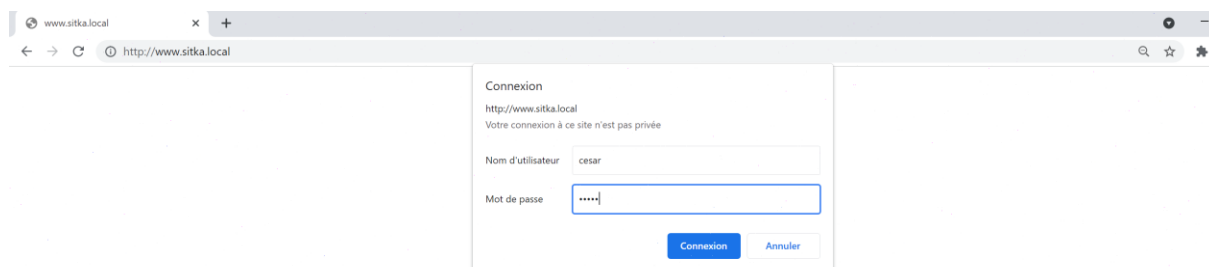
Création d'un fichier .htaccess avec le contenu comme indiqué ci-dessous

```
root@apache2:/var/www/sitka# vim .htaccess |  
  
AuthType Basic  
AuthName "Authentication user"  
AuthUserFile /etc/apache2/.htpasswd  
Require valid-user
```

Je redémarre mon service apache sans déconnecter mes utilisateurs

```
root@apache2:~# apachectl graceful
```

Et enfin on fait le test avec un navigateur



Notre page web s'affiche

