

### Introduction

A- **Installation de apache2**

1. Mettre à jour la machine
2. Renommez la machine apache2 3- Installation du paquet apache2 4- Test de l’installation
3. Comprendre l’arborescence des fichier d’apache2 Localiser les fichiers a- Explorer le fichier apache2.conf

b- le contexte directory

1. Les mods d'Apache2 et leur gestion le processus
   1. Les trois mods apache2 mpm\_event, mpm\_worker et mpm\_perfork
   2. 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
4. Utilisation du PHP dans le site
   1. Implémentation du mod-PHP
   2. Implémentation du FPM PHP C- **Sécurisation d’apache2**

1- Sécurisation apache en masquant sa version et l’os utilisé. 2- Sécurisation par SSL

1. Sécurisation des répertoires
   1. Utilisation des Virtuals host
   2. Utilisation du fichier .htaccess

### Introduction

Apache est le nom d[e](https://www.apache.org/) la fondation qui a develloppé 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

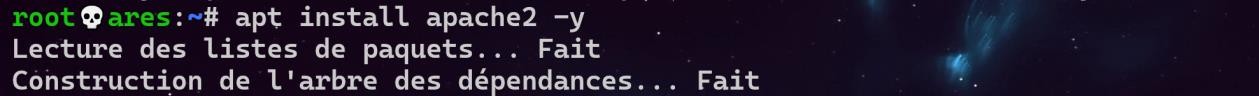
### Mettre à jour la machine



1. **Donner un nom à la ma chine apache2**

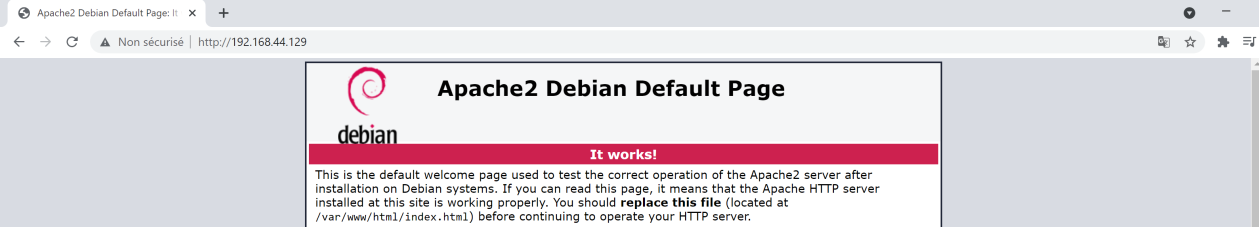


1. Installation du paquet apache2



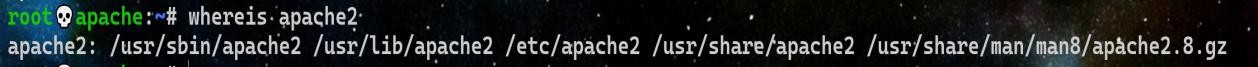
### Test de la connexion apache2

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



### Comprendre l’arborescence des fichier d’apache2 Localiser les fichiers

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



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



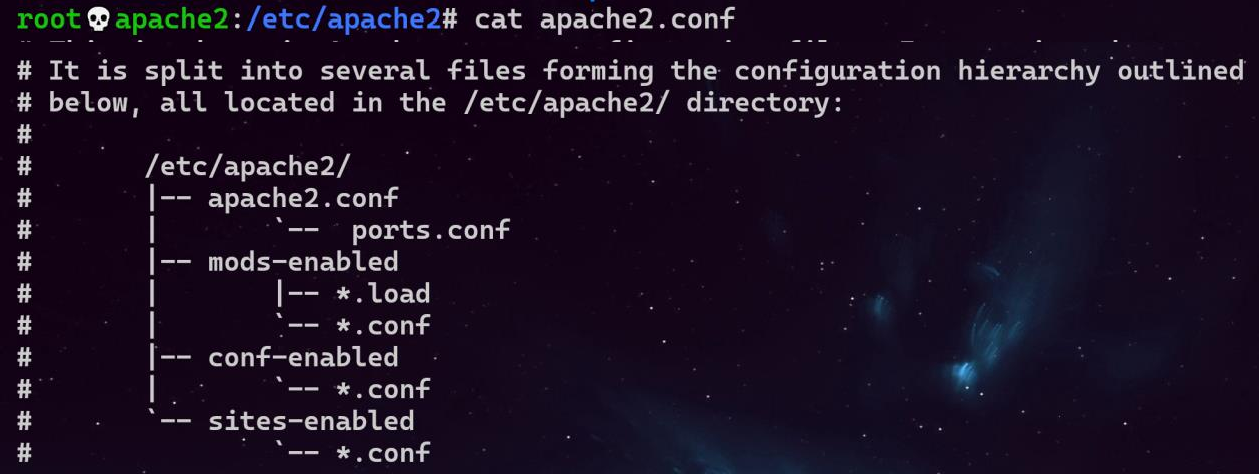
**/usr/ lib/ apache2** lib pour bibliothèque

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

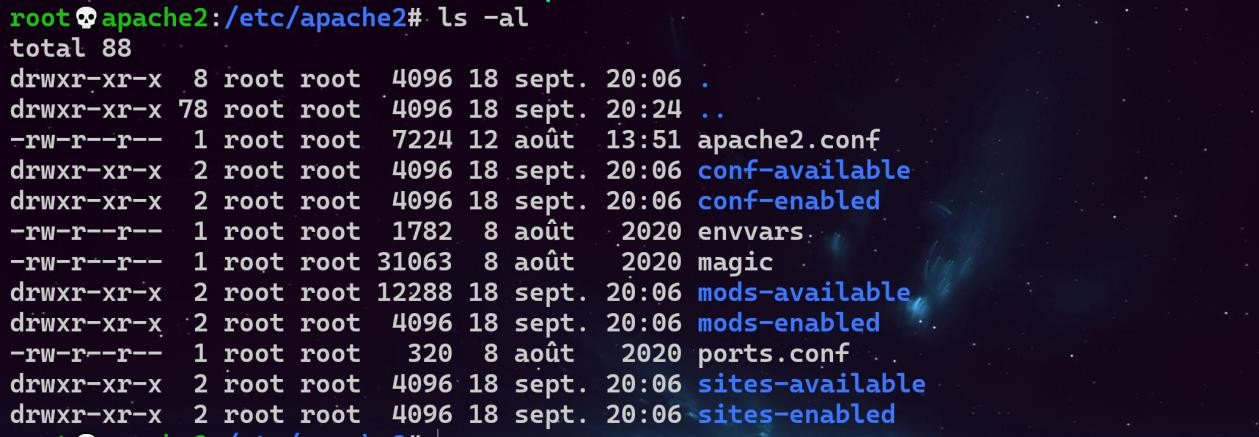
**/usr/ share/apache2**, des fichiers partagés

**/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.



On affiche le contenu de notre répertoire



### Les processus d'Apache et leur gestions le processus

1. Les trois mods apache2 mpm\_event, mpm\_worker et mpm\_perfork

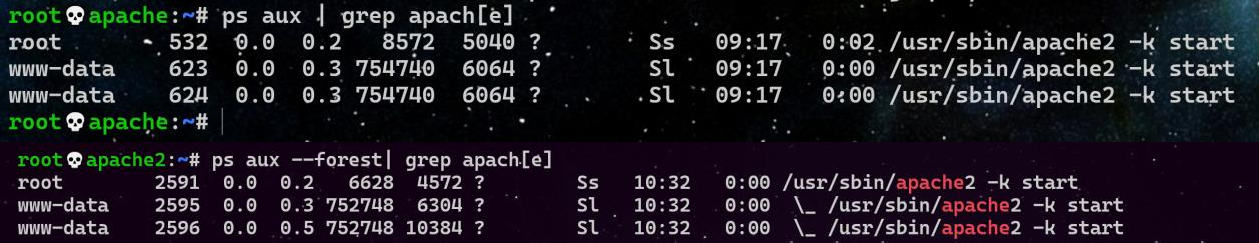
On utilise la commande **ps aux | grep apach[e]** pour afficher les processus d’apache2

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.



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 :

**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



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 processu**s 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.

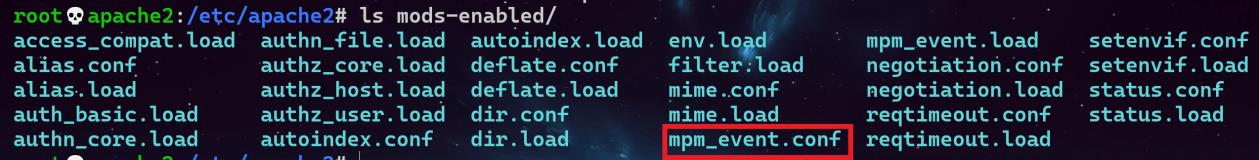
Un thread spécifique à la gestion du keepalive va permettre de diminuer le nombre de threads qui vont rester en keepalive c'est un des avantages de mpm\_events qui permet une optimisation en termes

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’Ideal.

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



Ou plus simplement avec la commande a2query -M



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 crées à cet effet ce qui peut s’avérer plus simple et pratique.

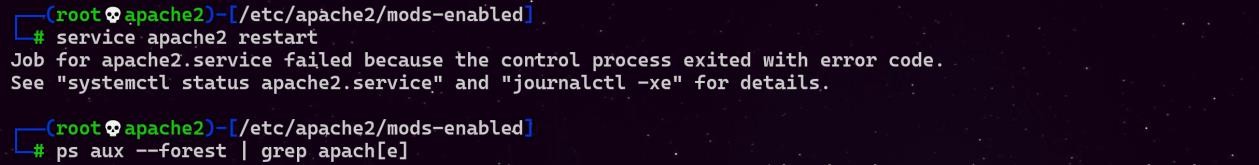
1. Activation et désactivation des mods mpm

i- Activation et désactivation manuelle des mod mpm

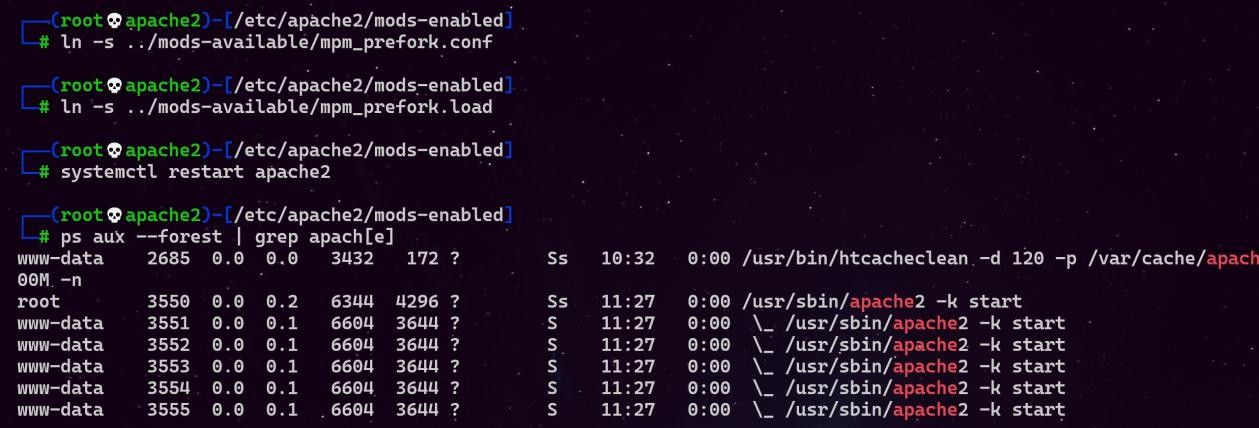
* + Je supprime le module mpm\_event



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

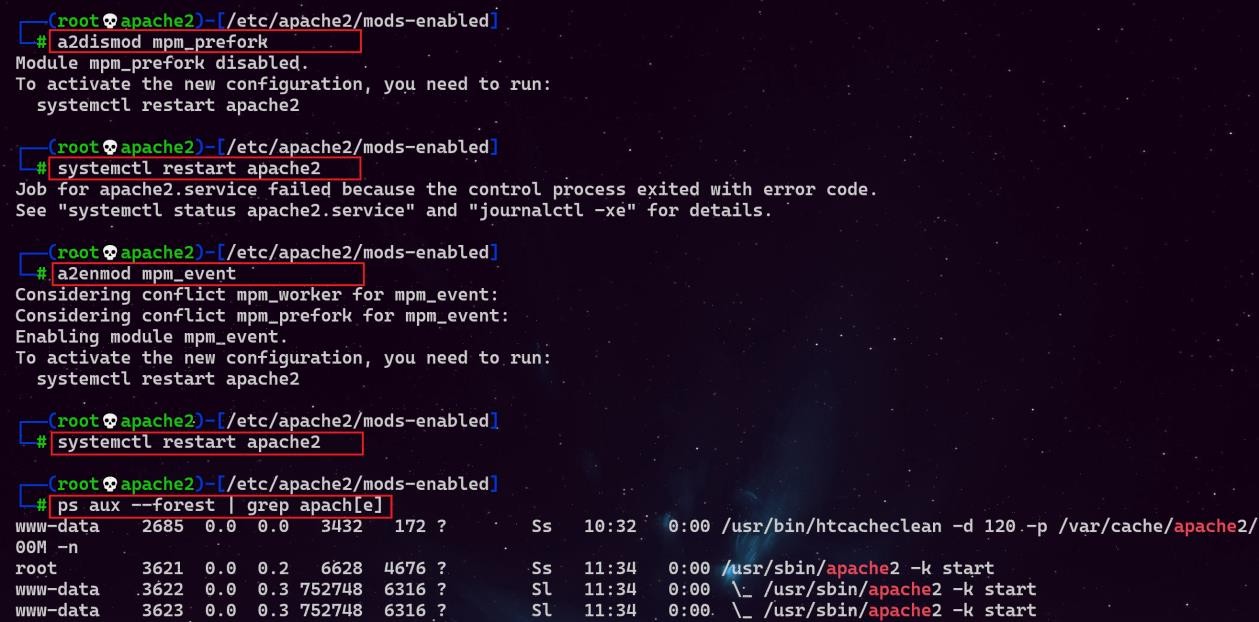


* + Je crée les liens symboliques, je redémarre le service apache2 et je vérifie que les processus sont démarrés



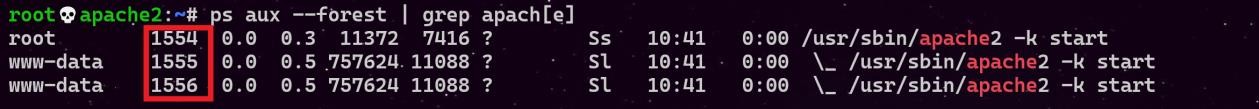
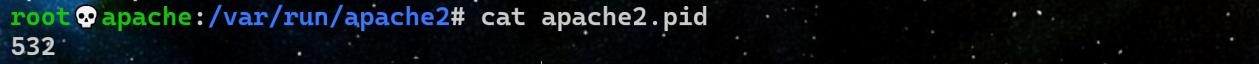
1. Activation et désactivation de mods mpm avec script

Maintenenant on va faire l’inverse en desactivant prefork et en activant event mais en utilisant les scriptes, on remerquera que c’est plus facile et pratique



1. Détermination du pid des processus

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



# 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

### 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**

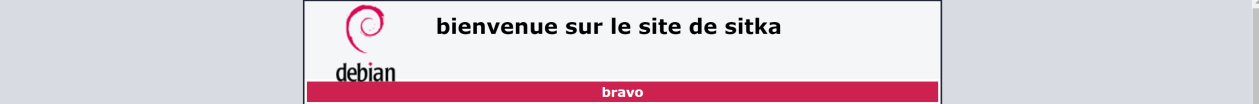


Modifier la page de sorte que :

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



Dans la page html ce qui est indiqué ci-dessous



### Création de site par nom de domaine

* 1. Copier le fichier conf du site par défaut d’apache 000-default.conf en sitka.conf



* 1. 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: ll 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 exmples :

* + 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.
  + Inclues : 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

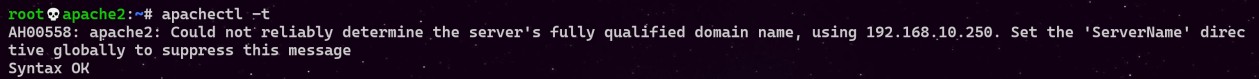
ne peut être surchargée par les options qui seraient contenues dans un fichier appelé

.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

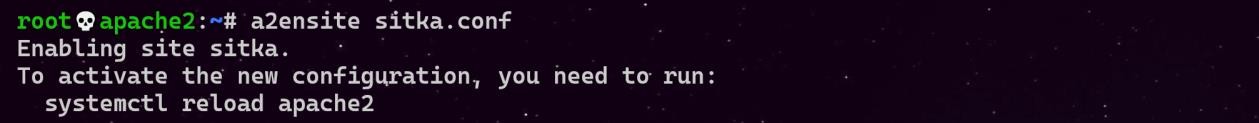
.htaccess auront la priorité sur celles du Virtual Host.



* 1. Vérification de la syntaxe de mon fichier de configuration sitka.conf



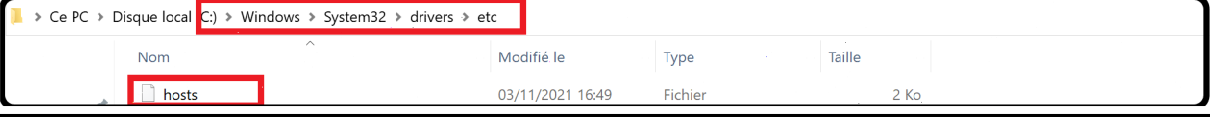
* 1. Activation du site Sitka



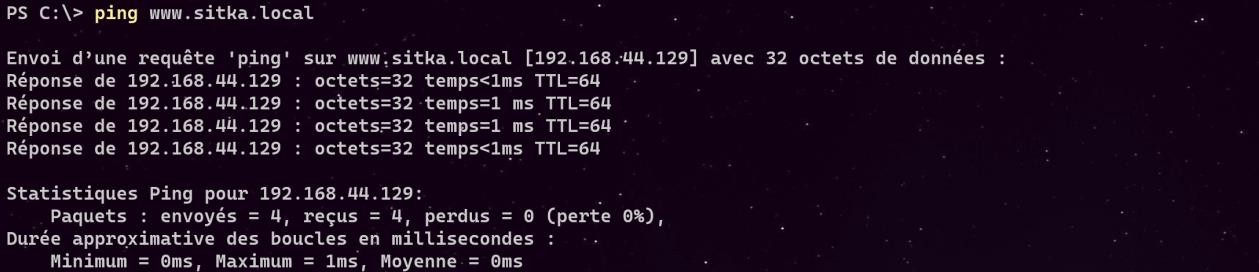
Je charge ma configuration



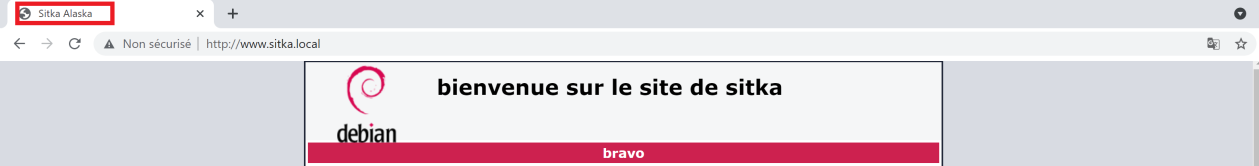
La résolution du nom [www.sitka.local](http://www.sitka.local/) doit se faire avec un serveur DNS pour nos test on peut se contenter d’utiliser le fichier hosts, sans la déclaration de [www.sitka.local](http://www.sitka.local/) dans un DNS ou le fichier host on ne pourra ni faire un ping ni accéder au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



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



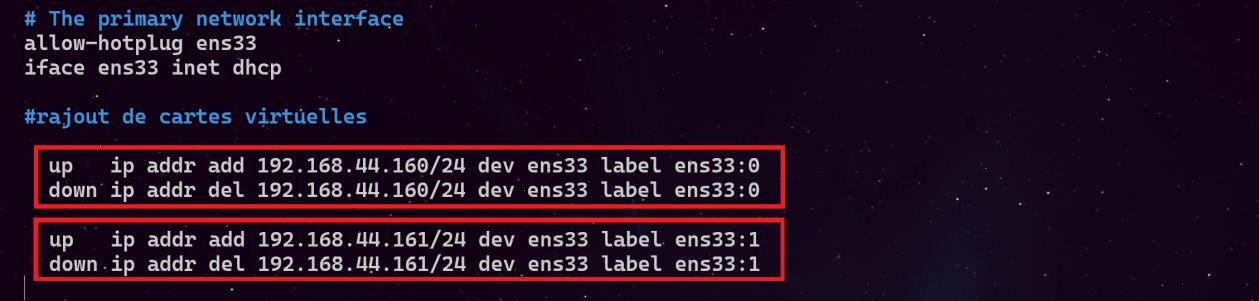
Le ping fonctionne donc on va maintenant tester notre site avec un navigateur



### Création de site par adresse IP

On crée des adresses virtuelles en rajoutant la configuration suivante dans mon fichier

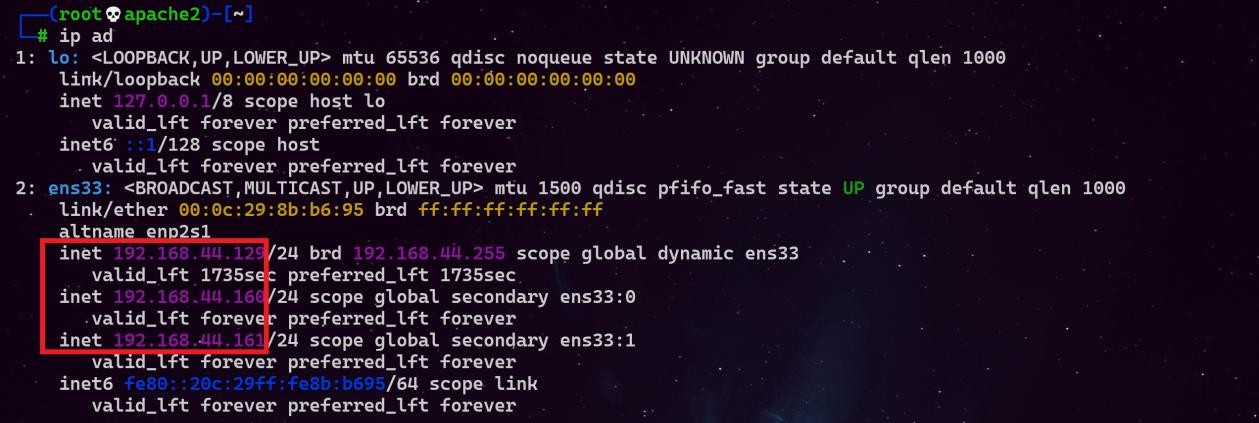
interfaces ici j’ai rajouté deux adresse virtuelles 192.168.44.160 et 192.168.44.161



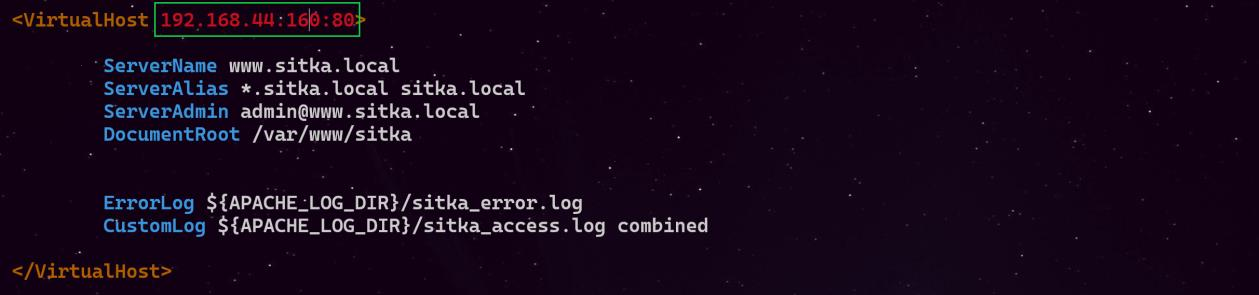
J’active et je désactive ma carte



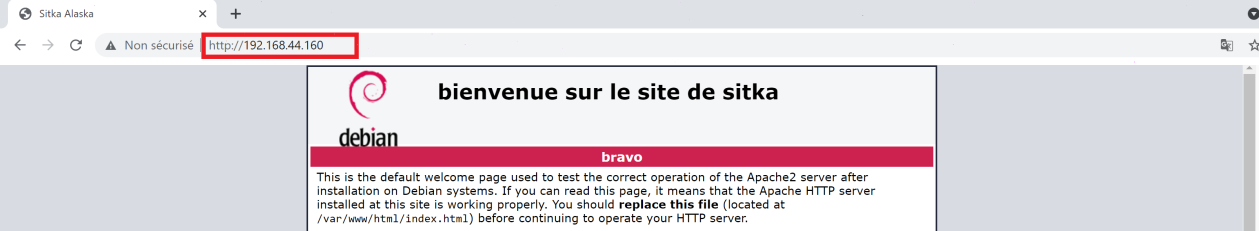
Je vérifie avec ip ad que mes deux adresse IP virtuelles sont actives



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**







### 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



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



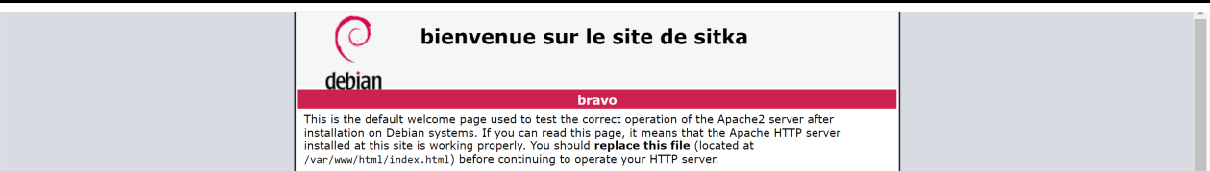
On vérifie la syntaxe



On redémarre apache2



On teste notre nouvelle configuration avec un navigateur.



1. **Utilisation du PHP dans le site**

Il existe deux moyens pour utiliser PHP dans un site web **mod\_php** et **FPM** :

1. **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 conséquence 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.

1. **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 dynamic 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.

### Implémentation du mod-PHP

On commencera par installer 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



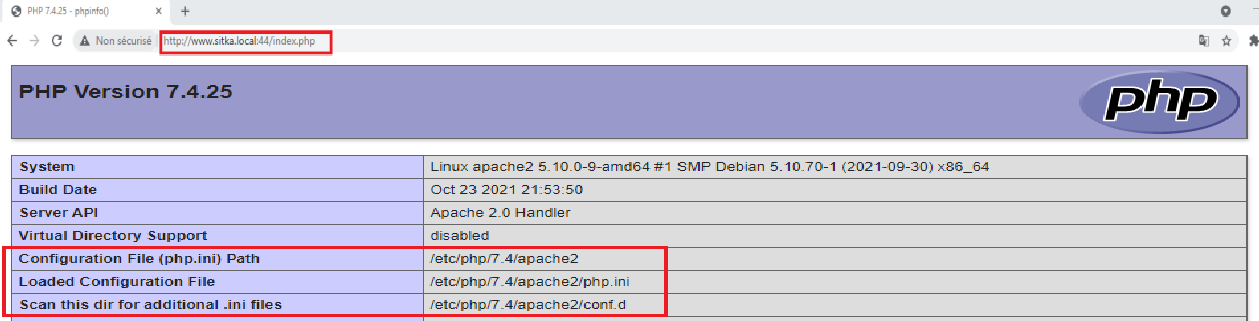
* Activation du mod mpm\_prefork



* Activation du mod php7.4 On redémarre le service apache2

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





### Implémentation du FPM PHP

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



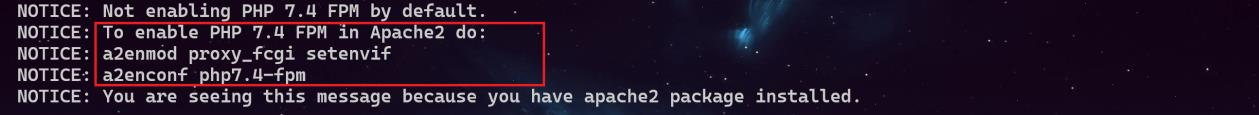
On redémarre le service apache2



Maintenant on peut commencer l’installation de php-fpm

A la fin d’installation on nous demande de démarrer les mods proxy\_fcgi et sentenvif ainsi que la

conf php7.4-fpm



On démarre les mods proxy\_fcgi et sentenvif



On démarre la conf php7.4-fpm



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"

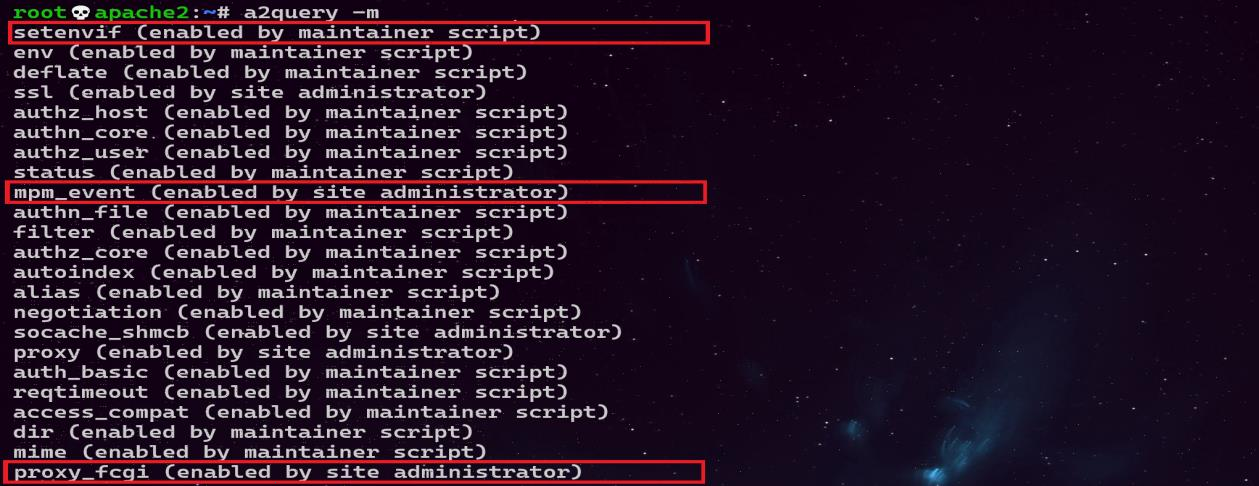
</FilesMatch>

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

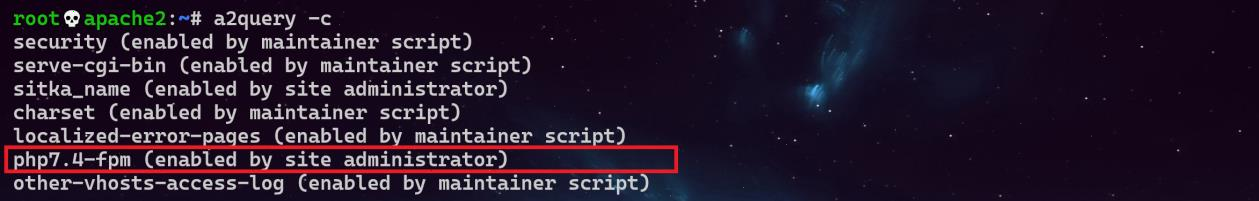
activer le mod event



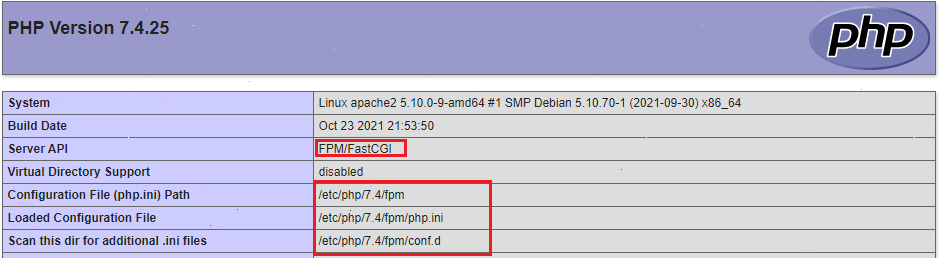
On vérifie que tous les mods nécessaires pour fpm PHP sont activées



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



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



# D- Sécurisation du site

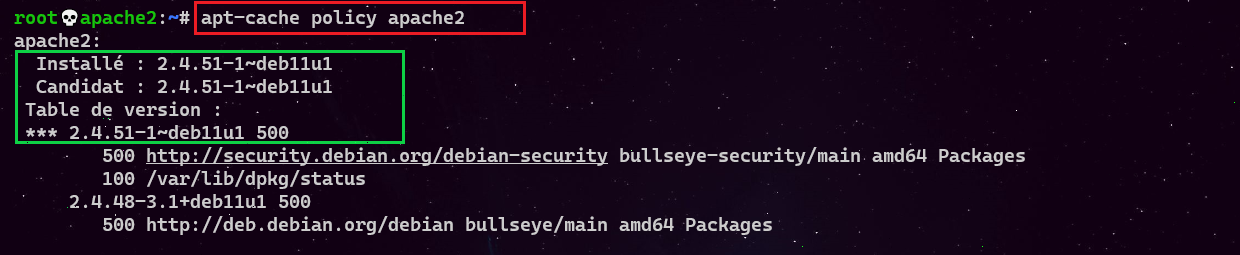
* 1. Sécurisation apache masquant sa version et l’os utilisé.

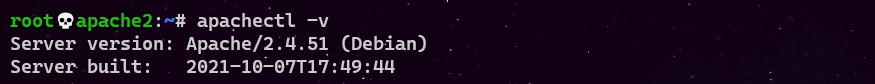
Apache envoie par défaut des entêtes HTTP contenant le nom et la version du serveur web ainsi que

le système d’exploitation qui héberge apache, ceci peut être problématique car on peut faciliter

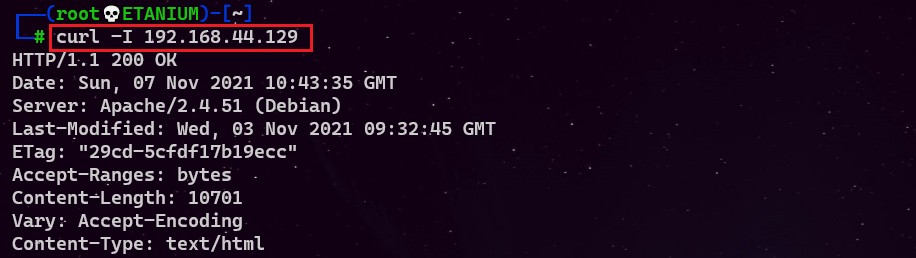
l’attaque de notre serveur en divulguant ces informations.

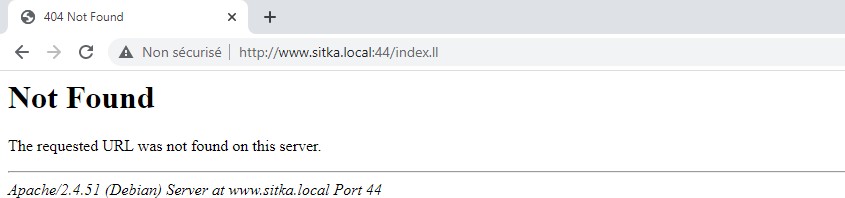
En local on peut afficher ces informations avec la commande apt policy apache2





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

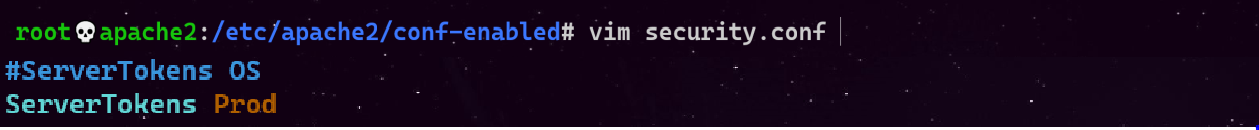


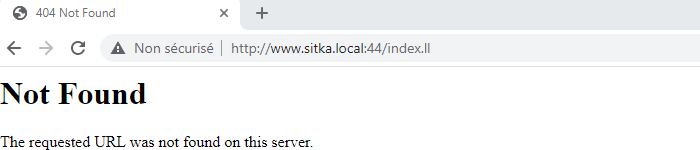


Pour cacher la version d'Apache, il faut changer des paramètres dans le fichier

/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.



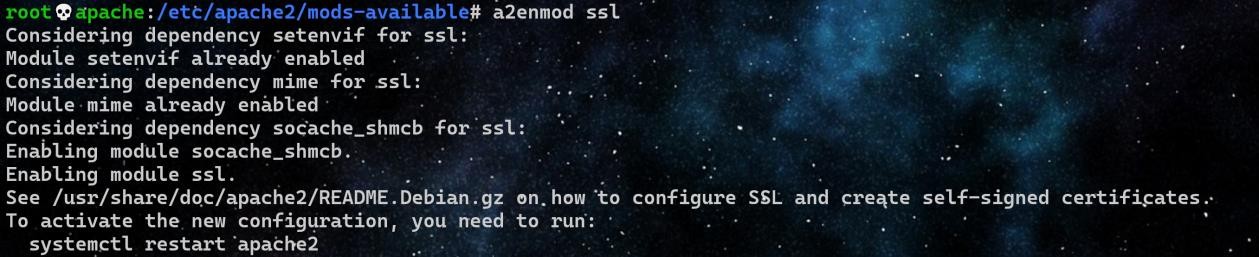


* 1. Sécurisation par SSL

On vérifie la présence du paquet ssl-cert

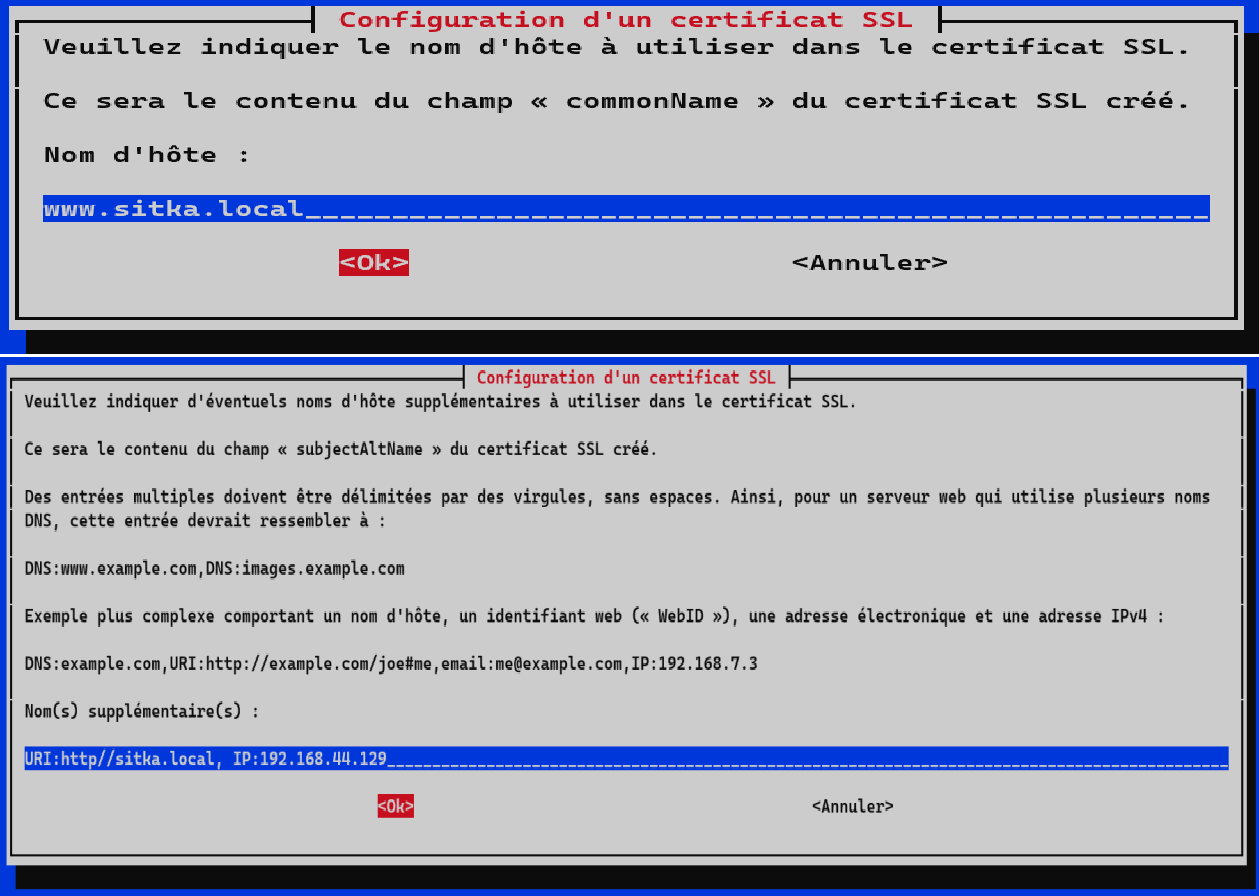


On active le mode ssl



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

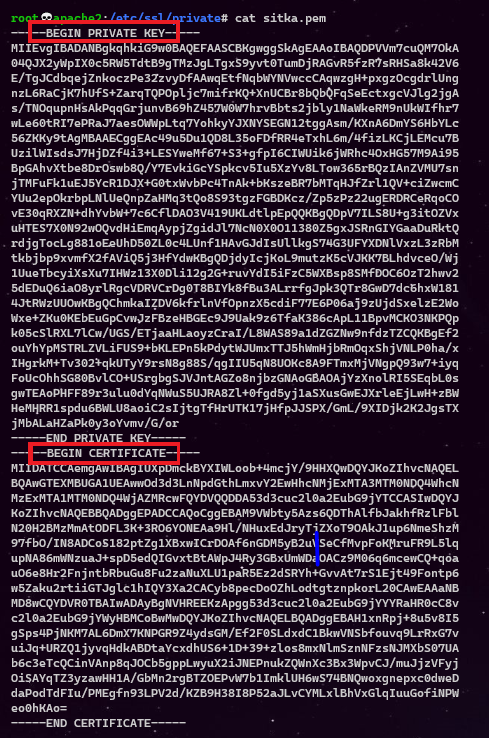
**root**† ¸**apache:~# make-ssl-cert /usr/share/ssl-cert/ssleay.cnf /etc/ssl/private/sitka.pem**



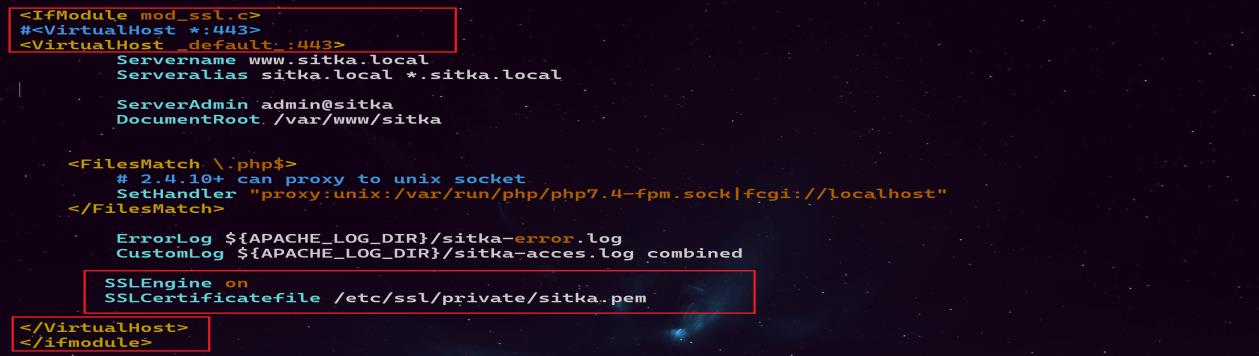
On vérifie la création du fichier pem



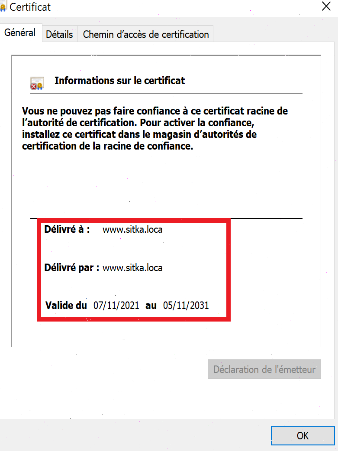
En affichant sitka.pem on se rend compte s’aperçoit qu’il possède un certificat et une clé privé



1. Rajout du ssl dans le fichier de configuration du site dans site-available



- Test de connexion avec ssl sur un navigateur



1. Sécurisation des répertoires

Il existe deux méthodes, en utilisant les Virtuals host cela donne de meilleures performances Cette option nécessite des droits afin d’accéder à la configuration, cette méthode est celle qui est recommandée.

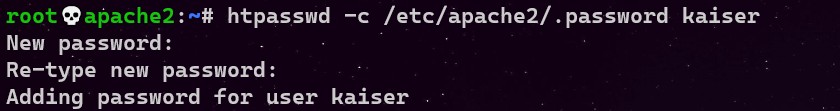
La deuxième méthode est l’utilisation des fichiers .htaccess afin de définir certains éléments de configuration dans un répertoire. Apache doit relire ces fichiers à chaque requête, ce qui peut avoir un impact négatif sur les performances.

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

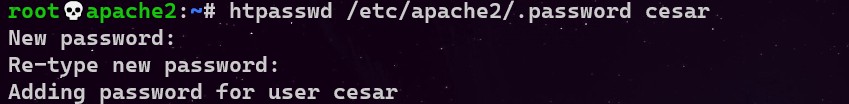
* 1. Utilisation des Virtuals host

On crée le fichier où les mots de passe vont être stockées, en même temps on crée le premier

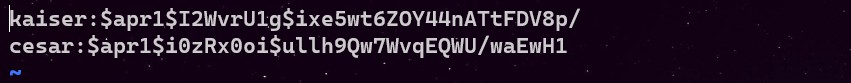
utilisateur que j’appelle kaiser



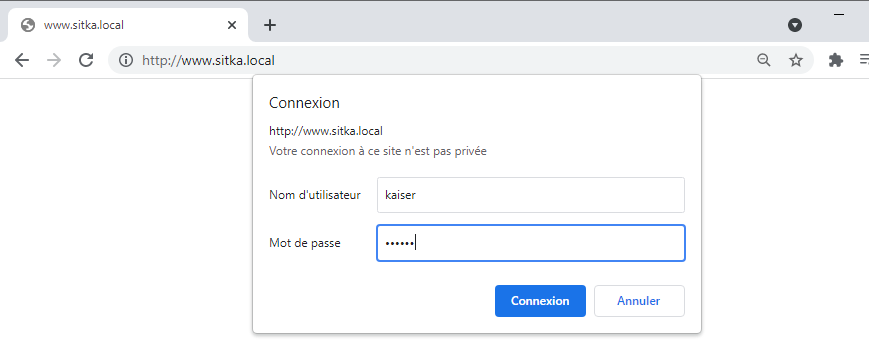
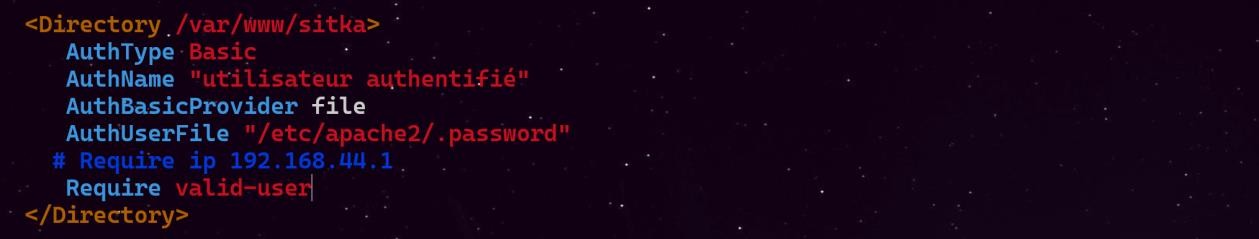
Je crée le deuxième utilisateur cesar, en enlevant l’option -c pour ne pas écraser le fichier . password



On vérifie la création des comptes utilisateur



On rajoute les directive directory dans les virtuels host



* 1. Utilisation du fichier .htaccess

Tout d’abord il faut changer directive à

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



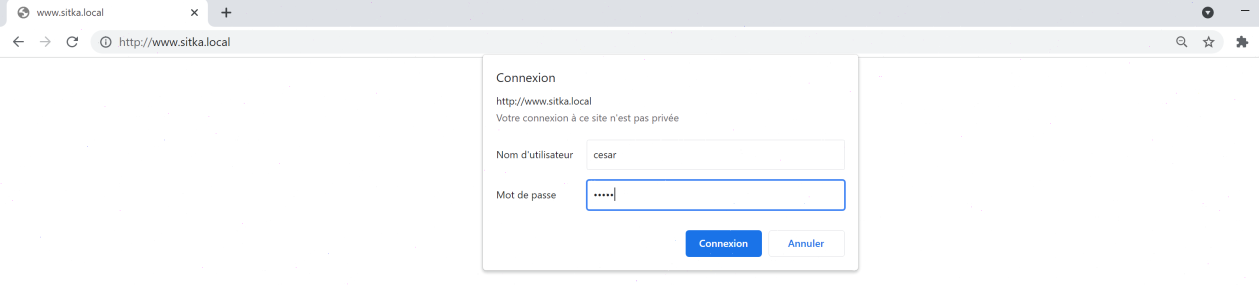
Création d’un fichier .htaccess avec le contenue comme indiqué ci-dessous



Je redémarre mon service apache sans déconnecter mes utilisateurs



Et enfin on fait le test avec un navigateur



Notre page web s’affiche

