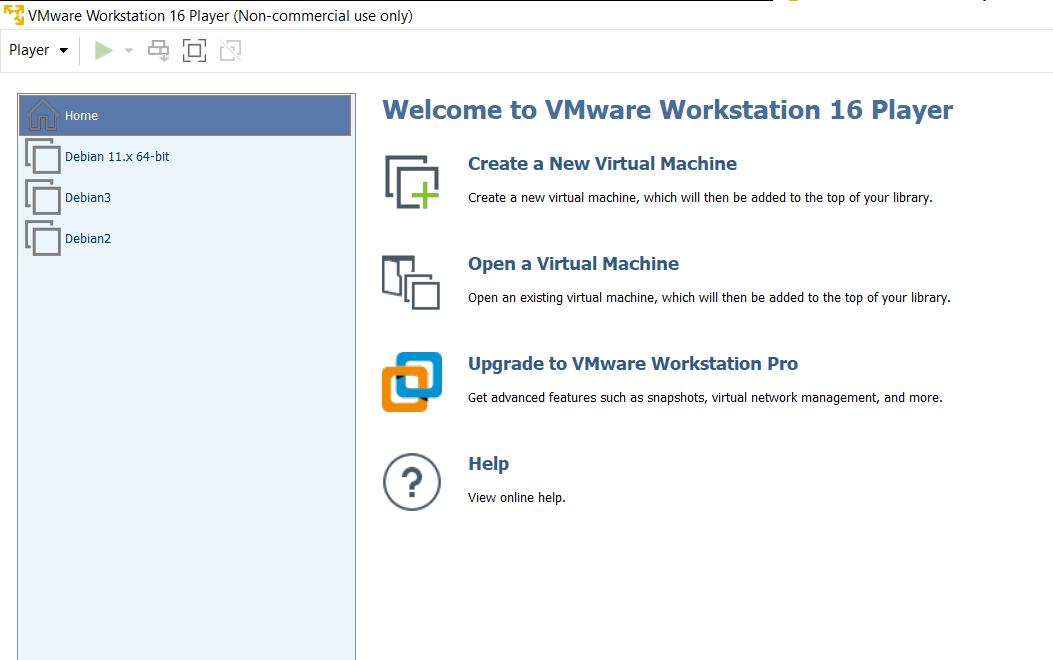
Installons plusieurs VM debian, la principale VM fait office d’hôte pour Apache2 , le serveur web utilisé.

Elle hébergera aussi le service DNS et utilisera DHCP afin de distribuer des IP aux autres VM.

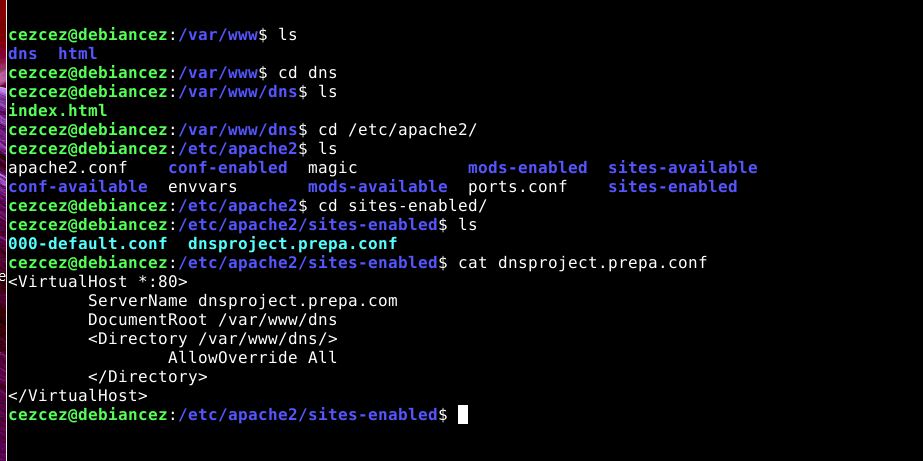


Une fois installée apache2 , créons un répertoire nommé “ dns “ dans /var/www qui contiendra une copie du fichier html de la page web par défaut apache2, nous pouvons la modifier à notre guise.

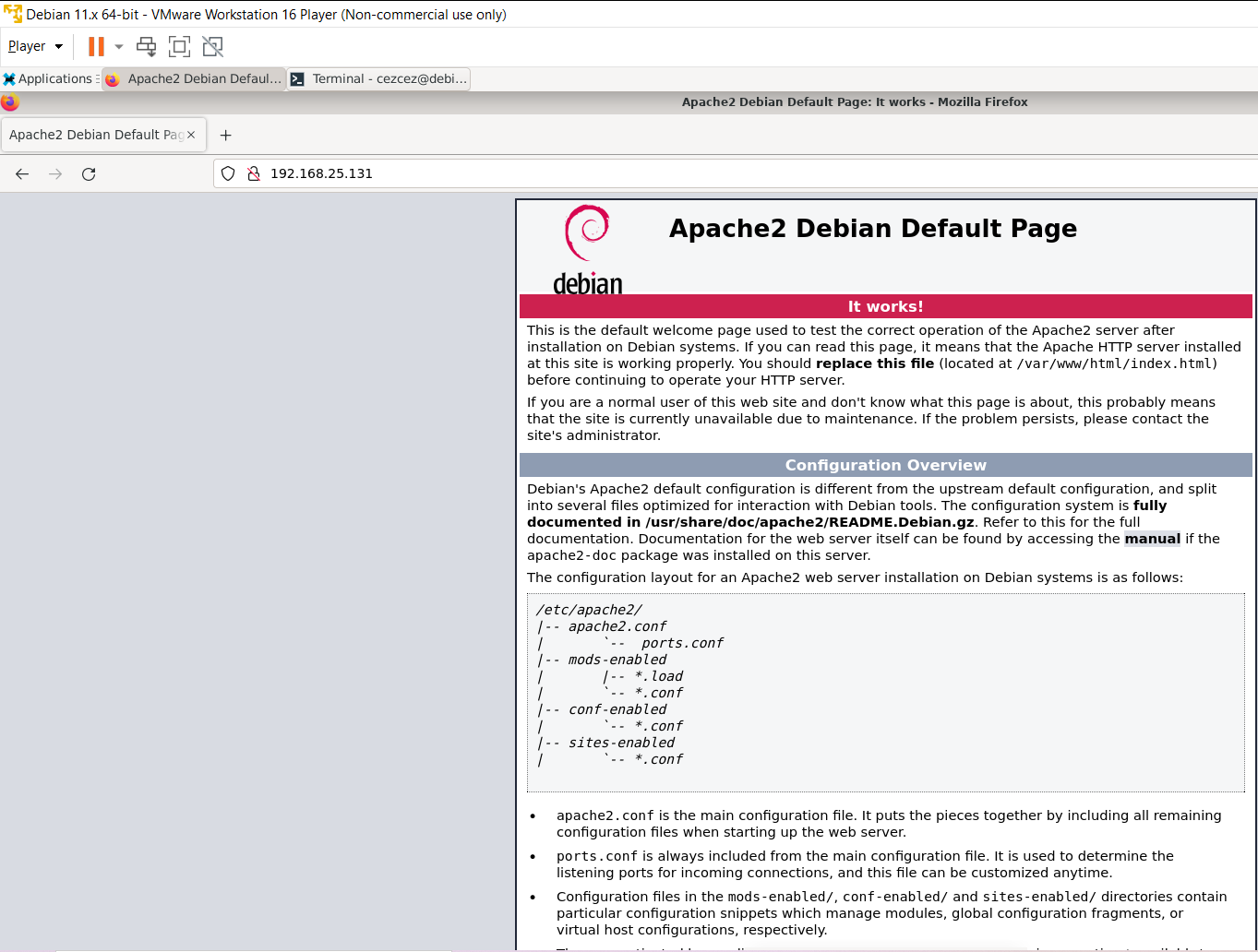
Ensuite il faut créer un fichier de configuration dans sites-available :

-80 pour le port http, nous lui indiquons ensuite ou se trouve le fichier html qui nous intéresse en renseignant l’adresse du répertoire dns créé-

Activons le site avec la commande a2ensite suivi du fichier .conf.



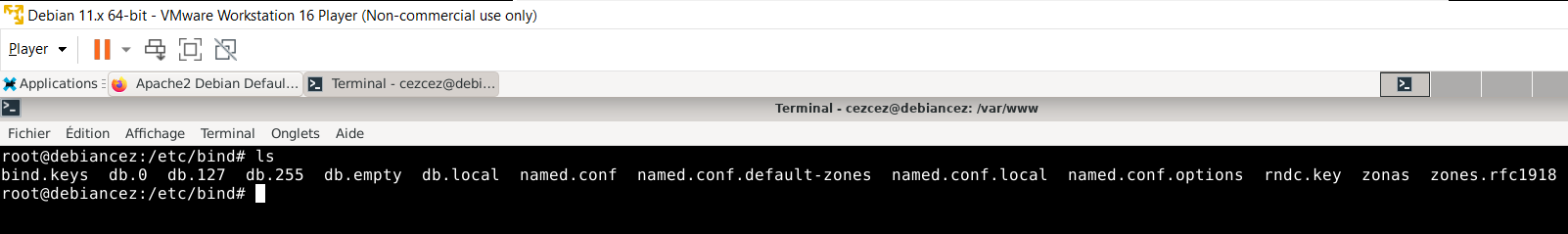
Nous pouvons désormais accéder à la page web à travers un moteur de recherche en renseignant l’ip de notre hôte.



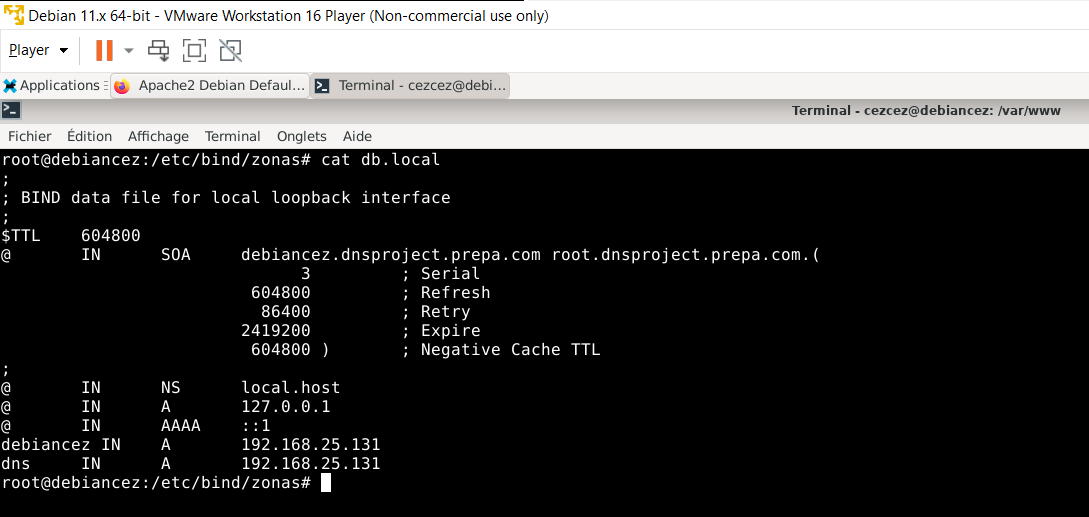
Maintenant nous allons mettre en place un serveur DNS qui fera correspondre l’adresse IP de notre serveur au nom de domaine local suivant “dnsproject.prepa.com”, on pourra donc ping le serveur à travers ce nom.

Pour ce faire, installons bind9.

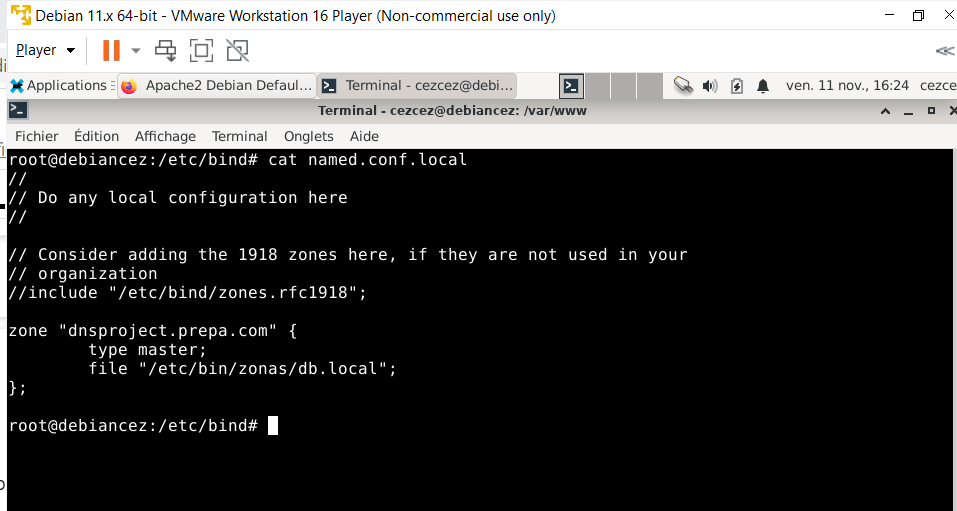
Nous pouvons voir ici les différents fichiers de configuration qu’il possède. Nous créons un nouveau répertoire pour notre site, qui contiendra un fichier de configuration propre à l’adresse DNS voulu pour notre serveur web. Ici il se nomme zonas.



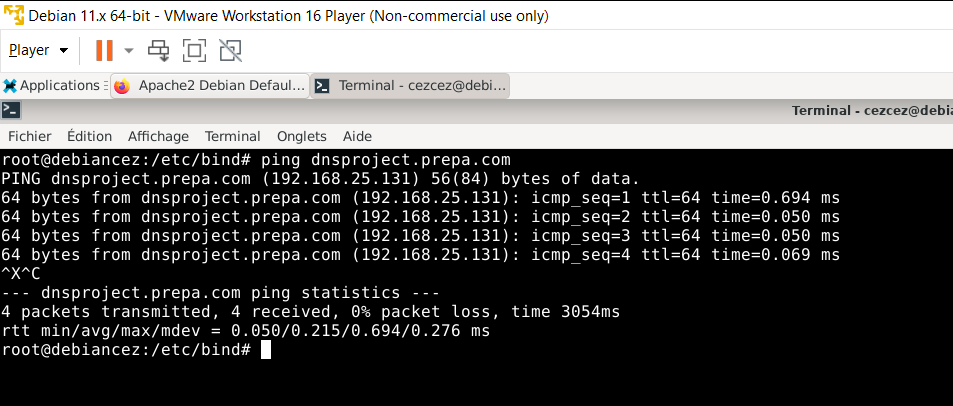
Nous lui copions le fichier db.local de base et ajustons selon l’ip de notre serveur son nom etc…



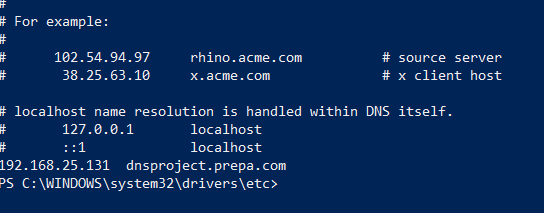
Ensuite dans le fichier named.conf.local nous rajoutons la zone dns et ou la chercher.



Nous pouvons désormais ping le serveur via le nom de domaine :



Pour que la machine hôte puisse ping le nom de domaine ajouter l’adresse dans le fichier hosts :

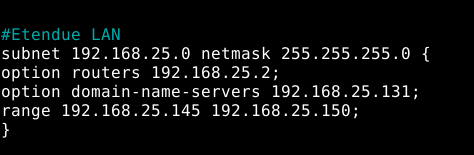


Malheureusement je n’ai pas encore réussi à trouver une manière optimisée pour paramétrer le DNS sur le serveur afin de ne pas changer le fichier hosts manuellement à chaque fois

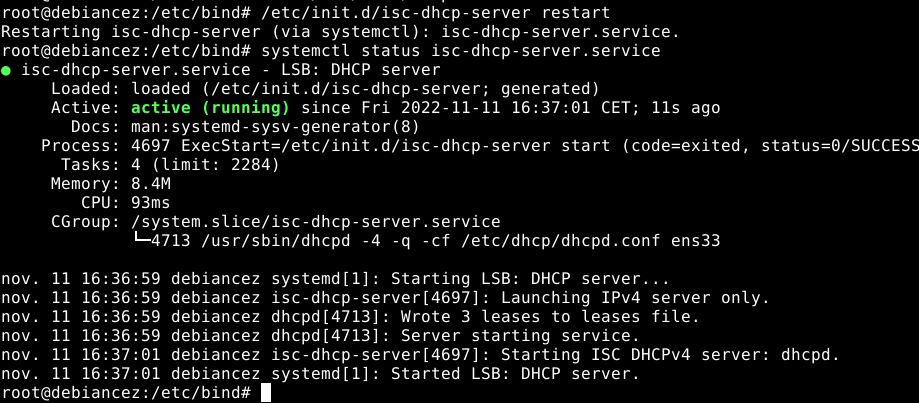
Maintenant nous allons mettre en place un serveur DHCP sur notre serveur central, afin d’attribuer des adresses ip à nos autres VM sur le serveur local.

Installons isc-dhcp-server et éditons le fichier dhcpd.conf ( nano /etc/dhcpd/dhcpd.conf )

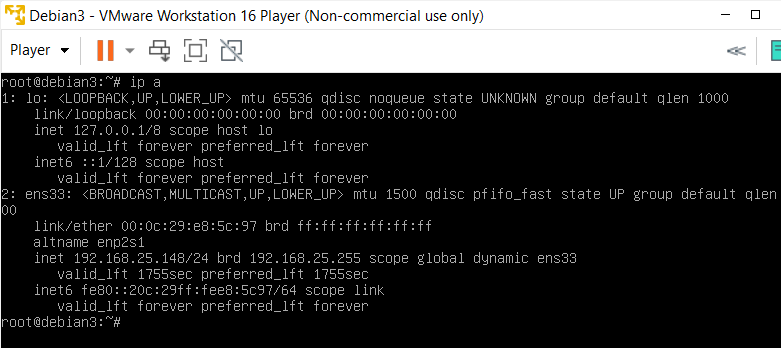
Nous ajoutons ces lignes suivantes comprenant le sous réseau et le routeur, l’indication du DNS correspondant à l'IP de notre serveur principal et la range des adresses IP voulue afin de délimiter une plage d’adresses.

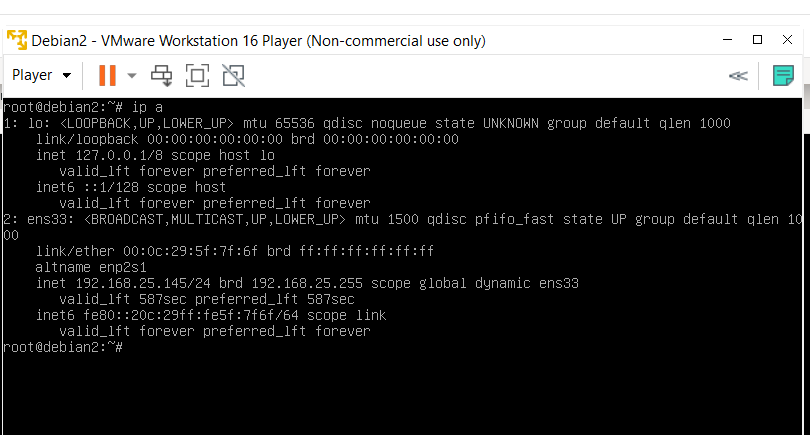


Et puis nous réinitialisons le service dhcp.



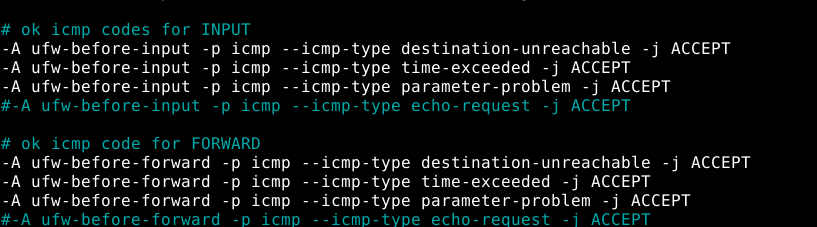
Nous pouvons lancer les deux autres VM et vérifier leur ip pour savoir si le dhcp est bien effectif





les deux ip4, terminent par 145 et 148, nous voulions une attribution d’ip entre 145 et 150, le serveur DHCP fonctionne donc.

Pour mettre en place le pare feu, installer ufw, modifier le fichier before.rules en commentant :



Définition wikipédia : “ Un **serveur web** est soit un [logiciel](https://fr.wikipedia.org/wiki/Logiciel) de service de ressources web (serveur [HTTP](https://fr.wikipedia.org/wiki/Hypertext_Transfer_Protocol)), soit un [serveur informatique](https://fr.wikipedia.org/wiki/Serveur_informatique) ([ordinateur](https://fr.wikipedia.org/wiki/Ordinateur)) qui répond à des requêtes du [World Wide Web](https://fr.wikipedia.org/wiki/World_Wide_Web) sur un réseau public ([Internet](https://fr.wikipedia.org/wiki/Internet)) ou privé ([intranet](https://fr.wikipedia.org/wiki/Intranet))[1](https://fr.wikipedia.org/wiki/Serveur_web#cite_note-1),[2](https://fr.wikipedia.org/wiki/Serveur_web#cite_note-2),[3](https://fr.wikipedia.org/wiki/Serveur_web#cite_note-django-3), en utilisant principalement le protocole HTTP. “

C’est donc grâce aux serveurs web qu’on peut enregistrer des contenus web et assurer leur accessibilité aux utilisateurs de manière sûre.   
Les éléments que l’on aperçoit d’une page sont toujours envoyés sur notre ordinateur depuis un serveur web, pour qu’un site web soit accessible à tout moment, le server web sur lequel il est hébergé doit être connecté à Internet en permanence.

Le terme de serveur Web peut en général se référer à deux choses différentes : soit au **logiciel d’un serveur Web**, soit à **la machine** sur laquelle s’exécute le programme. Lorsqu’il s’agit de la seconde définition, on parle généralement d’hébergeur ou d’hôte (un tel hébergeur peut abriter plusieurs programmes de serveur Web).

Pour visiter un site web, on doit renseigner l’adresse correspondante dans un navigateur, qui se charge d’envoyer une requête au serveur Web, ce dernier lui envoie une réponse ( sous forme de page HTML qui peut être statique ou dynamique )

La transmission est effectuée via le protocole de communication HTTP ( et sa variante sécurisée HTTPS ), qui repose sur le protocole de réseau IP et TCP en très grande majorité.

Évidemment le nombre et la rapidité de traitement des différentes requêtes utilisateur dépendent du matériel et de la charge des flux de données de l’hébergeur ainsi que la complexité des contenus Web.

Même si la transmission de contenus Web est la mission principale des logiciels de serveurs Web, de nombreux programmes proposent des fictions supplémentaires :

* Sécurité : Cryptage des communications entre serveur et client Web via HTTPS
* Identification utilisateur : Authentification HTTP autorisant l’accès à des domaines particuliers du web
* Redirections : Redirection et réécriture d’URL (URL Rewriting grâce au Rewrite-Engine
* Caching (mémoire cache) : Enregistrement intermédiaire de de documents dynamiques pour répondre plus vite aux demandes et prévenir une surcharge du serveur Web
* Attribution de Cookies : Envoi et traitement de cookies HTTP

( Un hébergeur peut être mis en relation avec d’autres programmes que des logiciels de serveur Web. Un client FTP peut par exemple servir au chargement des données ou en tant que base de données de serveur pour des contenus dynamiques ) .

**Distinction** : serveur Web / hébergement Web

* Un serveur Web représente par définition une composante fondamentale d’un hébergement Web.
* L’hébergement Web se caractérise par le stockage et la mise à disponibilité de contenus Web. Cet espace Web est pour ainsi dire un disque dur sur lequel les contenus Web sont stockés.
* Cette mémoire est *liée à un serveur Web*, pour que les contenus en ligne soient rendus disponibles.

Si l’on veut donc faire un hébergement Web autonome, nous devons l’aménager par nous même, pour cela nous aurons besoin d’un programme de serveur Web.

**Différents serveurs web open source :**

**Apache** ( celui que nous utiliserons ) : Il alimente toujours la majorité des sites Web dans le monde pour plusieurs raisons : une licence ouverte, une entrée anticipée ( publiée en 1995, et un déploiement facile de PHP.

* Apache est disponible sur toutes les plateformes - Linux, Windows, MacOs…
* C’est le serveur par défaut pour tous les hébergements partagés CPANEL (<https://fr.wikipedia.org/wiki/CPanel>), ce qui facilite la configuration et le changement de site.
* Des tonnes de fonctionnalités offertes par une large collection de modules
* Configuration par répertoire via .htaccess fichiers.

Et bien d’autres choses : mise à jour régulière, fiable et stable, facile à configurer, grande communauté…

Inconvénients :

* Problèmes de performances sur les sites web avec un énorme trafic.
* Trop d’options de configurations peuvent mener à la vulnérabilité de la sécurité.

**Nginx** : Lorsque Nginx est sorti, cela a fait beaucoup de bruit et les gens se sont éloignés d’Apache, ses améliorations lui permettent de continuellement progresser dans l’usage des utilisateurs notamment grâce à son architecture asynchrone qui permet une gestion de charges élevées.

On peut donc concevoir des systèmes complexes et distribués avec une mise en cache, un proxy et un équilibre de charge puissants.

Mais contrairement à Apache il ne dispose pas de fichiers de configuration par répertoire comme .htacces, chaque modification de configuration doit être effectuée dans le fichier de conf du serveur principal et nécessite donc un rechargement de ce dernier.

**Caddy** : Un des nouveaux frameworks en vogue. Syntaxe similaire et Nginx mais simplifié à l’extrême.

* HTTPS activé par défaut, rien à faire pour l’installation ou le renouvellement des certificats SSL.
* Il sert également les noms DNS publics via HTTPS en utilisant des certificats d’une autorité de certification ACME publique telle que Let’s Encrypt ou ZeroSSL.
* Caddy maintient le renouvellement de tous les certificats gérés et redirige automatiquement HTTP ( port 80 ) vers HTTPS ( port 443 )

Étant donné qu’il est un jeune web serveur, il n’est pas aussi compréhensible que des solutions plus anciennes comme Apache ou Nginx, spécialement quand il y a des règles plus complexes il atteint ses limites .

**Lighttpd :** Construit sur le modèle de gestion des demandes asynchrones et reflète donc essentiellement le fonctionnement de Nginx . Conçu pour surmonter le défi de l’utilisation des ressources dans des environnements à faible mémoire et CPU.

Fonctionne sur un seul thread, donc si une machine est performante, elle ignorera les autres cœurs de processeur.   
Il n’intègre pas des fonctionnalités plus sophistiquées de Nginx telles que le proxy SSL, HTTP / 2, etc.

<https://web.developpez.com/actu/329121/Caddy-l-unique-serveur-web-a-utiliser-HTTPS-automatiquement-et-par-defaut-ecrit-en-langage-Go/>

<https://www.hostinger.fr/tutoriels/quest-ce-quapache-serveur-web-apache#Les_avantages_et_inconvenients_dApache>

<https://fr.wikipedia.org/wiki/Serveur_web>

<http://www.ordinateur.cc/r%C3%A9seaux/Autre-R%C3%A9seaux-informatiques/78896.html>

<https://geekflare.com/fr/open-source-web-servers/>

<https://www.ionos.fr/digitalguide/serveur/know-how/serveur-web-definition-bases-astuces-et-logiciels/>

<https://lkml.iu.edu/hypermail/linux/kernel/9608/0191.html>

<https://guide.ubuntu-fr.org/server/httpd.html>