DDWS

Job 1

Si on souhaite installer SSH, la commande est **sudo apt install openssh-server**.

Job 2

Pour installer apache2 **sudo apt install apache2**

Job 3

Apache est : Open-source et gratuit même pour un usage commercial, il a des mise à jour régulière, correctifs de sécurité réguliers, flexible grâce à sa structure basée sur des modules, Plateforme-Cross (fonctionne sur les serveurs Unix et Windows) et fonctionne avec les sites WordPress.

Mais, il a des problèmes de performances sur les sites web avec un énorme trafic et il a trop d'options de configuration pouvant mener à la vulnérabilité de la sécurité.

Nginx est plus rapide, il sert du contenu statique environ 2,5 fois plus rapidement qu'Apache et il gère mieux le trafic élevé qu'Apache.

Mais il a : des options limitées, une communauté moins développée que celle d'Apache et une moins bonne option pour servir du contenu dynamique qu'Apache.

Tomcat est facile à installer et simple à configurer, il possède des fonctionnalités de sécurité intégrées et il a des options simples pour le déploiement d'applications Web.

Mais, il a : un problème lié à une fuite de mémoire et une prise en charge des clusters qui n'est pas suffisante.

Job 4

Pour pouvoir mettre en place un DNS sur notre serveur Linux qui fera correspondre l'adresse IP de notre serveur au nom de domaine local suivant est : **sudo nano /etc/hosts** et j'y écrit : **127.0.0.1 dnsproject.prepa.com**

Job 5

Un nom de domaine est l'adresse que les visiteurs vont renseigner afin d'accéder à votre site Internet.
Les règles de réservation d'un nom de domaine varient selon la nature du site :

- Domaines géographiques à vocation nationale, selon la localisation géographique de l'entreprise : .fr (France), .de (Allemagne), .it (Italie), .eu (Union européenne)

- Domaines génériques, à vocation internationale : .com (pour les activités commerciales), .net (pour les entreprises), .org (pour les associations ou organisations non gouvernementales, etc.)

Le nom de domaine est attribué à celui qui en demande la réservation en premier. C'est donc la règle du premier arrivé, premier servi qui prévaut.
Pour réserver un nom de domaine, il faut s'adresser à l'organisme gestionnaire qui en a la charge.

JOB 6

Pour pouvoir mettre en place un serveur DNS, il faut installer plusieurs paquets. La commande pour installer plusieurs paquets à la fois est **sudo apt -y install.** Les paquets nécessaires pour ce job sont : bind9, bind9utils et dnsutils.

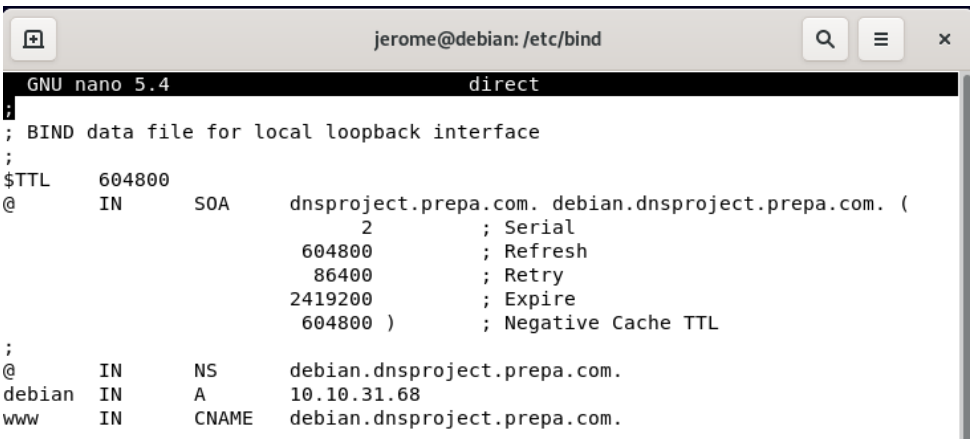
```
jerome@debian:~$ sudo apt -y install apache2 bind9 bind9utils dnsutils ufw isc-d
hcp-server
```

Ensuite, on crée une copie du fichier de configuration **db.local** et on la nomme direct.

```
jerome@debian:/etc/bind$ sudo cp db.local direct
```

J'y écris mon **hostname** qui est **debian**, le **nom de domaine** ici **dnsproject.prepa.com** et l'**IP de mon serveur**, ici **10.10.31.68**.

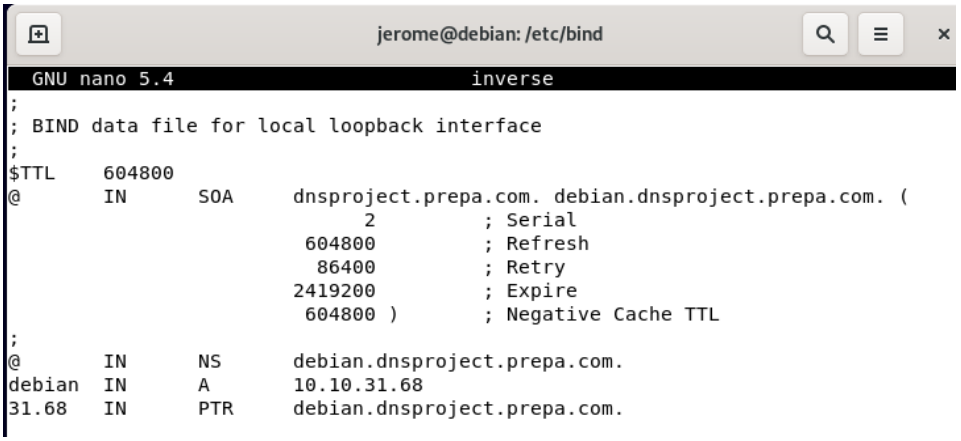
Si on souhaite modifier mon IP j'exécute la commande suivante : **sudo ifconfig nom_carte_réseau IP_souhaité**.
Si la commande est introuvable on l'installe en faisant : **sudo apt install net-tools**.



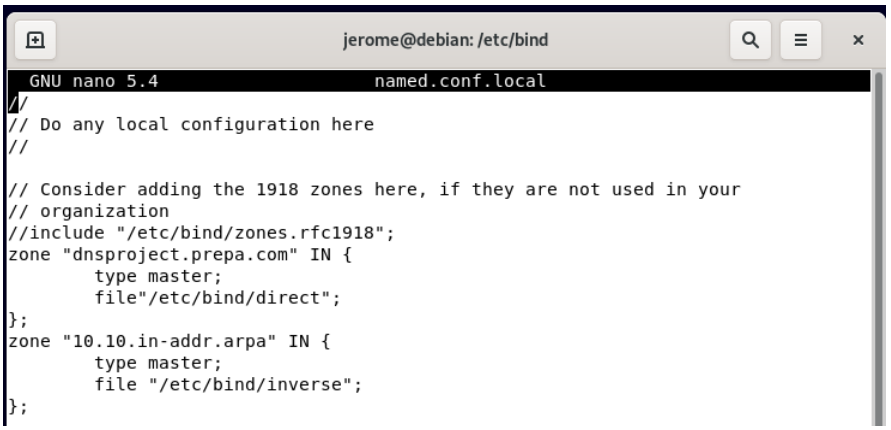
Ensuite on crée le fichier inverse. Pour aller plus vite, on copie le fichier de configuration direct et l'appel inverse.

```
jerome@debian:/etc/bind$ sudo cp direct inverse
```

Dans ce fichier on modifie uniquement la dernière ligne en mettant **Les_deux_derniers_octets_IP IN PTR Hostname.address_DNS.**



On modifie le fichier **named.conf.local** pour qu'il prenne en compte les fichiers de configuration **direct** et **inverse**.



Enfin le dernier fichier à modifier est le fichier **resolv.conf** pour que notre ordinateur puisse prendre en compte notre DNS.

On redémarre le service bind9

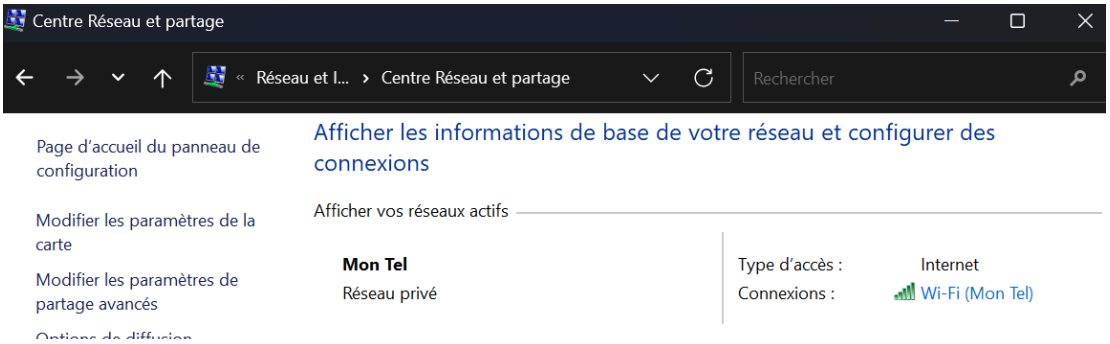
```
jerome@debian:/etc/bind$ sudo systemctl restart bind9
```

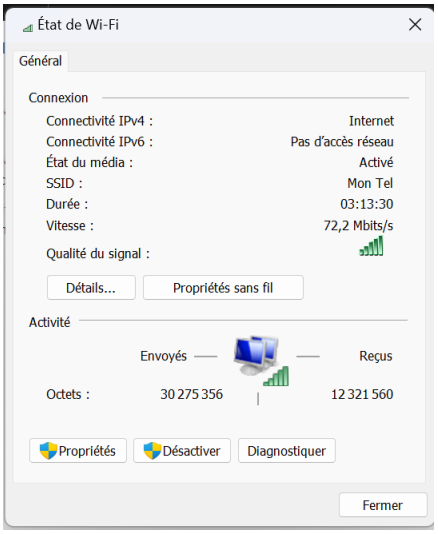
J'exécute la commande **nslookup www** pour vérifier si le serveur DNS y est bien associé.

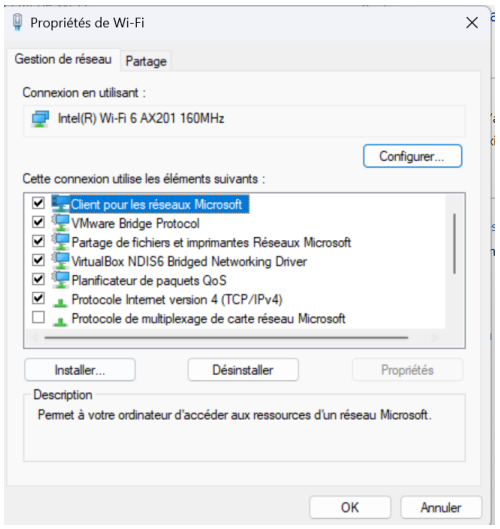
```
jerome@debian:/etc/bind$ nslookup www
Server:         10.10.31.68
Address:        10.10.31.68#53

www.dnsproject.prepa.com        canonical name = debian.dnsproject.prepa.com.
Name:   debian.dnsproject.prepa.com
Address: 10.10.31.68
```

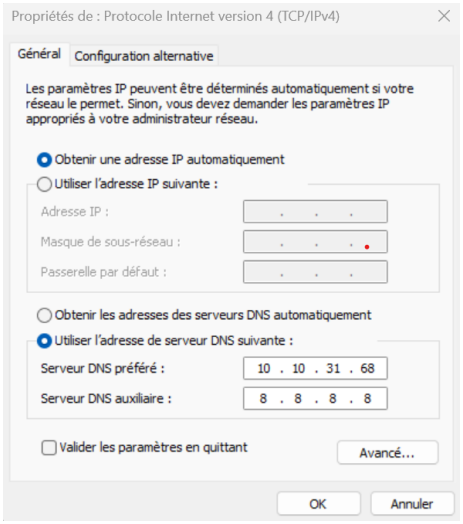
Comme tout est fonctionnel, on peut maintenant passer à la configuration du côté hôte.

On va dans le **panneau de configuration**, puis dans **Réseau et Internet**, dans **wifi**, dans **Propriétés**. Enfin dans **Protocole Internet version 4**, on met l'**adresse IP de notre serveur DNS** dans la section **DNS préféré**. Pour avoir encore accès à internet on met l'IP **8.8.8.8** dans **DNS Auxiliaire**.







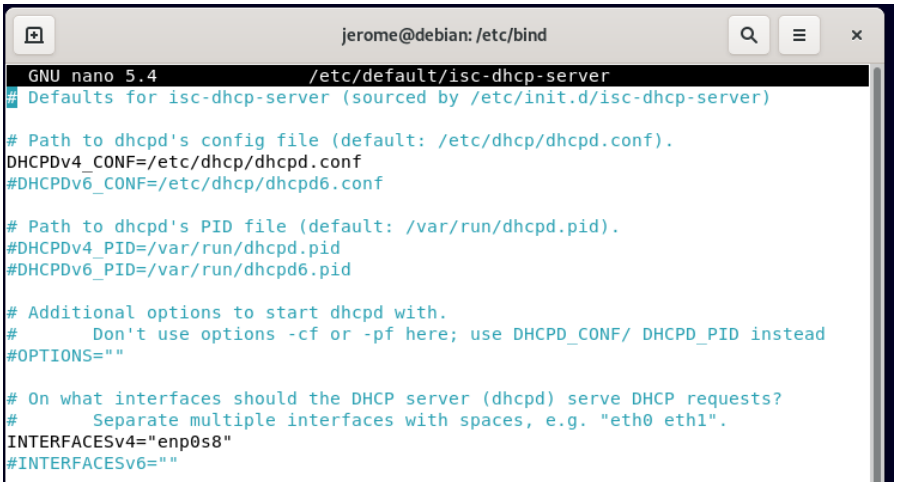


JOB 7

Pour pouvoir faire un serveur DHCP, il faut tout d'abord l'installer. La commande est **sudo apt install isc-dhcp-server**. Nous l'avons installé dans les jobs précédents.

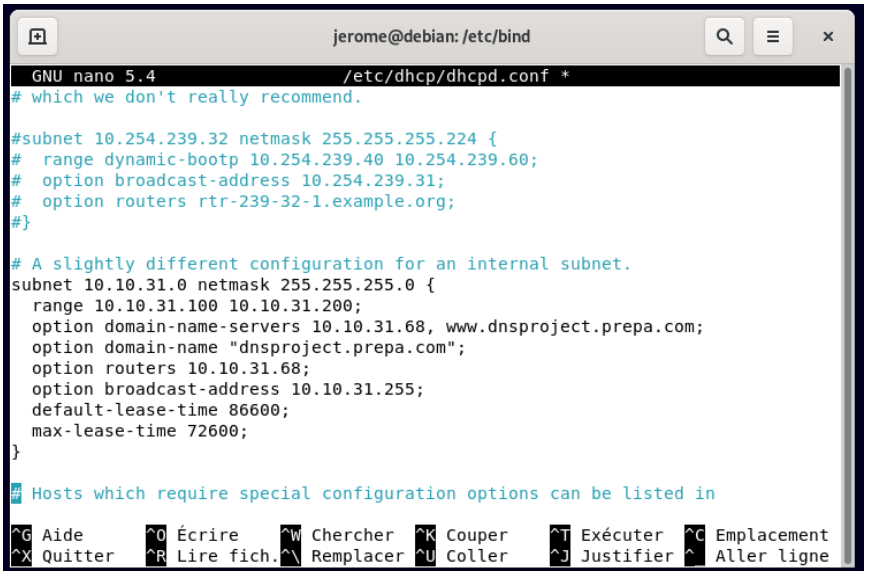
Le premier fichier à modifier est **/etc/default/isc-dhcp-server**. Dans ce dernier, on indique sur quelle carte réseau le serveur doit être configuré.
Pour connaître mon ip et le nom de mes cartes réseaux, on tape **ip a**.

```
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast sta
P group default qlen 1000
    link/ether 08:00:27:a1:0e:55 brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.15/24 brd 10.0.2.255 scope global dynamic noprefixroute en
       valid_lft 73009sec preferred_lft 73009sec
    inet6 fe80::a00:27ff:fea1:e55/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
3: enp0s8: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast sta
P group default qlen 1000
    link/ether 08:00:27:04:b5:8e brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.80/24 brd 192.168.1.255 scope global enp0s8
       valid_lft forever preferred_lft forever
4: enp0s9: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast sta
P group default qlen 1000
    link/ether 08:00:27:49:fd:61 brd ff:ff:ff:ff:ff:ff
    inet 192.168.100.2/24 brd 192.168.100.255 scope global enp0s9
       valid_lft forever preferred_lft forever
```



Ensuite, l'autre fichier à modifier est **/etc/dhcp/dhcpd.conf**
Dans ce dernier on va y indiquer le sous réseau, le netmask, le DNS, le gateway et les paramètres nécessaires au fonctionnement du sous réseau.

```
# option definitions common to all supported networks...
#option domain-name "example.org";
#option domain-name-servers ns1.example.org, ns2.example.org;

#default-lease-time 600;
#max-lease-time 7200;

# The ddns-updates-style parameter controls whether or not the server will
# attempt to do a DNS update when a lease is confirmed. We default to the
# behavior of the version 2 packages ('none', since DHCP v2 didn't
# have support for DDNS.)
#ddns-update-style none;

# If this DHCP server is the official DHCP server for the local
# network, the authoritative directive should be uncommented.
authoritative;
```



Enfin, on redémarre le service

```
jerome@debian:/etc/bind$ sudo systemctl restart isc-dhcp-server.service
```

JOB 9

Pour pouvoir installer ufw, on tape la commande suivante **sudo apt install ufw**. L'installation a été faite dans les jobs précédents.

La configuration initiale est de tout bloquer, puis d'ouvrir les ports selon nos besoins.

```
jerome@debian:~$ sudo ufw default deny incoming
Default incoming policy changed to 'deny'
(be sure to update your rules accordingly)
jerome@debian:~$ sudo ufw default deny outgoing
Default outgoing policy changed to 'deny'
```

Pour activer ufw, il faut taper **sudo ufw enable**. Pour le désactiver c'est **sudo ufw disable**.

Pour pouvoir ajouter des règles à ufw, il faut taper **sudo ufw allow n°_port/tcp**

Ici nos besoins sont : SSH (22) DNS (53), DHCP (67), HTTP/apache2 (80), HTTPS (443).

```
jerome@debian:/etc/apache2$ sudo ufw status
Status: active

To                         Action      From
--                         ------      ----
80/tcp                     ALLOW       Anywhere
53/tcp                     ALLOW       Anywhere
443/tcp                    ALLOW       Anywhere
67/tcp                     ALLOW       Anywhere
22/tcp                     ALLOW       Anywhere
80/tcp (v6)                ALLOW       Anywhere (v6)
53/tcp (v6)                ALLOW       Anywhere (v6)
443/tcp (v6)               ALLOW       Anywhere (v6)
67/tcp (v6)                ALLOW       Anywhere (v6)
22/tcp (v6)                ALLOW       Anywhere (v6)
```

JOB 10

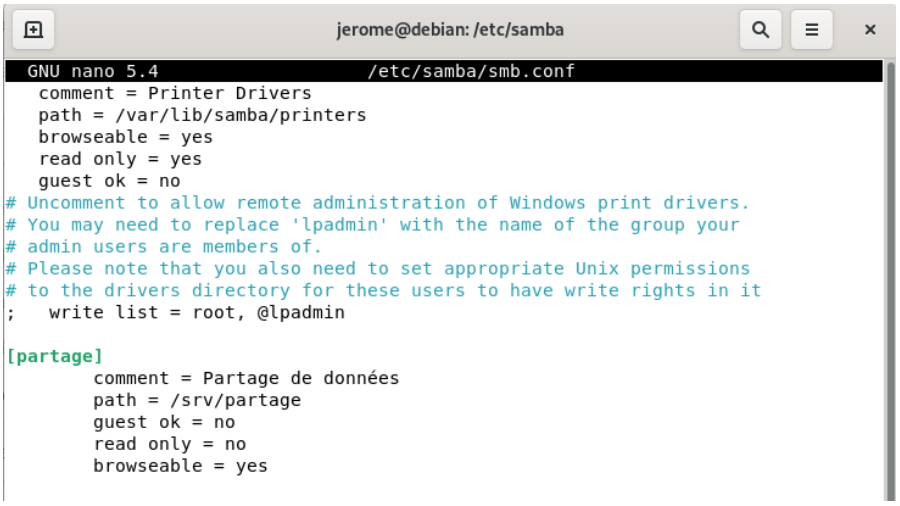
Pour pouvoir créer un dossier partagé, il faut installer samba. Pour cela, la commande est **sudo apt install samba**

```
jerome@debian:~$ sudo apt install -y samba
```

Puis on active le service

```
jerome@debian:~$ sudo systemctl enable smbd
Synchronizing state of smbd service with SysV
```

On édite le fichier **/etc/samba/smb.conf**. On y met les paramètres du dossier : son nom, son chemin d'accès, les permissions, l'accès guest qui correspond à la connexion invité ou anonyme ou encore sa visibilité sur les autres ordinateurs.



On redémarre le service :

```
jerome@debian:/etc/samba$ sudo systemctl restart smbd
```

L'étape qui suit est d'ajouter un utilisateur a samba. Pour cela on va assigner un mot de passe à l'utilisateur par samba. La commande est : **sudo smbpasswd -a nom_utilisateur**.

```
jerome@debian:~$ sudo smbpasswd -a jerome
```

Ensuite il faut créer un groupe ayant un nom qui fait par exemple référence au nom du dossier

```
jerome@debian:~$ sudo groupadd partage
jerome@debian:~$ sudo gpasswd -a jerome partage
```

On crée le dossier qui va être partagé et donne la propriété du dossier au groupe précédemment créé et on lui donne les droits d'écriture et de lecture.

```
jerome@debian:~$ sudo mkdir /srv/partage
jerome@debian:~$ sudo chgrp -R partage /srv/partage
jerome@debian:~$ sudo chmod -R g+rw /srv/partage/
```

Pour finir n'oublions pas d'ajouter samba à ufw

```
jerome@debian:/etc/samba$ sudo ufw allow 139/tcp
Rules updated
Rules updated (v6)
jerome@debian:/etc/samba$ sudo ufw allow 445/tcp
Rules updated
Rules updated (v6)
```

JOB Pour Aller plus Loin

Pour pouvoir installer ssl, on exécute la commande **sudo apt install openssl**.

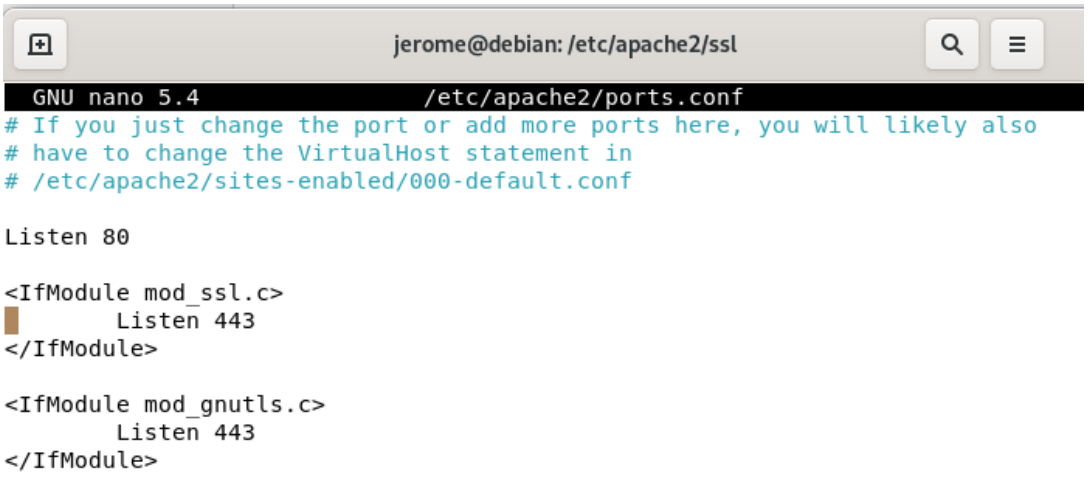
La première étape pour pouvoir rediriger notre DNS en https est de configurer SSL. Pour cela on crée un dossier où on va pouvoir y mettre notre certificat auto-signé.

```
jerome@debian:/etc/apache2$ sudo mkdir ssl
```

On crée mon certificat en tapant la commande : **sudo /usr/sbin/make-ssl-cert /usr/share/ssl-cert/ssleay.cnf /etc/apache2/ssl/apache.pem**

```
jerome@debian:/etc/apache2/ssl$ sudo /usr/sbin/make-ssl-cert /usr/share/ssl-cer
t/ssleay.cnf /etc/apache2/ssl/apache.pem
```

On vérifie si le module ssl est bien pris en charge dans **/etc/apache2/ports.conf**



Le dernier fichier à modifier est **/etc/apache2/sites-available/default-ssl.conf**

On active ssl avec apache en faisant :

```
jerome@debian:/etc/apache2/sites-available$ sudo a2enmod ssl
```

```
jerome@debian:/etc/apache2/sites-available$ sudo a2ensite default-ssl
```

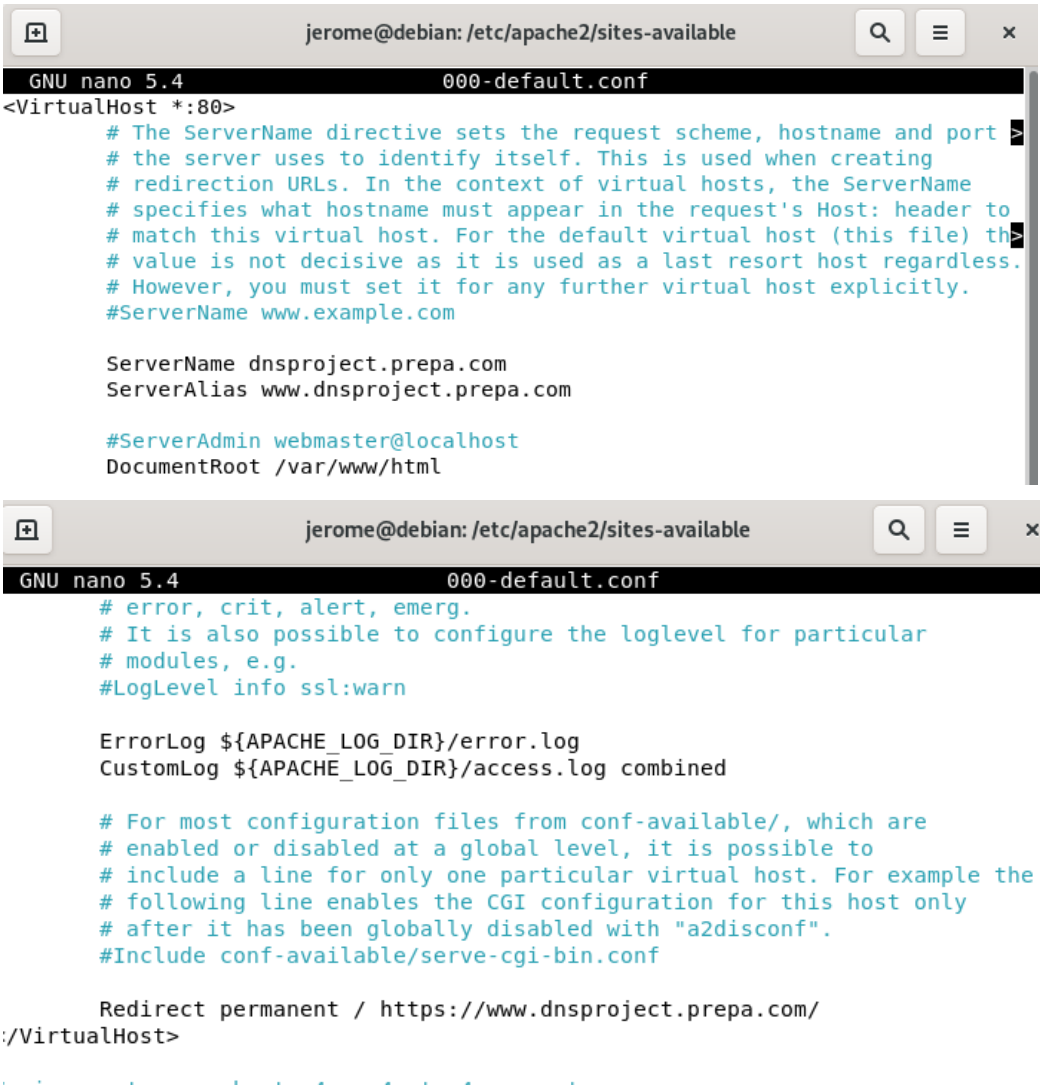
On redémarre le service apache en faisant **sudo systemctl restart apache2**.

Maintenant que ssl est configuré, on peut s'occuper de la redirection.

Tout d'abord on active module rewrite et on redémarre apache

```
jerome@debian:/etc/samba$ sudo a2enmod rewrite
Enabling module rewrite.
To activate the new configuration, you need to run:
  systemctl restart apache2
jerome@debian:/etc/samba$ sudo systemctl restart apache2
```

Pour finir on édite le fichier **/etc/apache2/sites-available/000-default.conf** pour qu'il prenne en compte notre DNS et la redirection en HTTPS.



La différence entre les certificats SSL donnés par des organismes extérieurs et celui auto-signé est que pour celui donné par un organisme extérieur, le certificat est validé par une Autorité de Certification. Alors que pour un certificat auto-signé, aucune approbation n'est nécessaire.
Ce qui conduit pour le certificat auto-signé à l'affichage d'une erreur de sécurité sur tous les navigateurs web et donc qui apparaît comme non sécurisé dans tous les navigateurs.

Documentation faite par Jérôme MUSCAT