

**Ténénan Chevalier**

**GUIDE MISE EN PLACE  
SERVICE WEB/HAPROXY**

**Sous rocky linux 9.6**

## Installation de Nginx & MariaDB

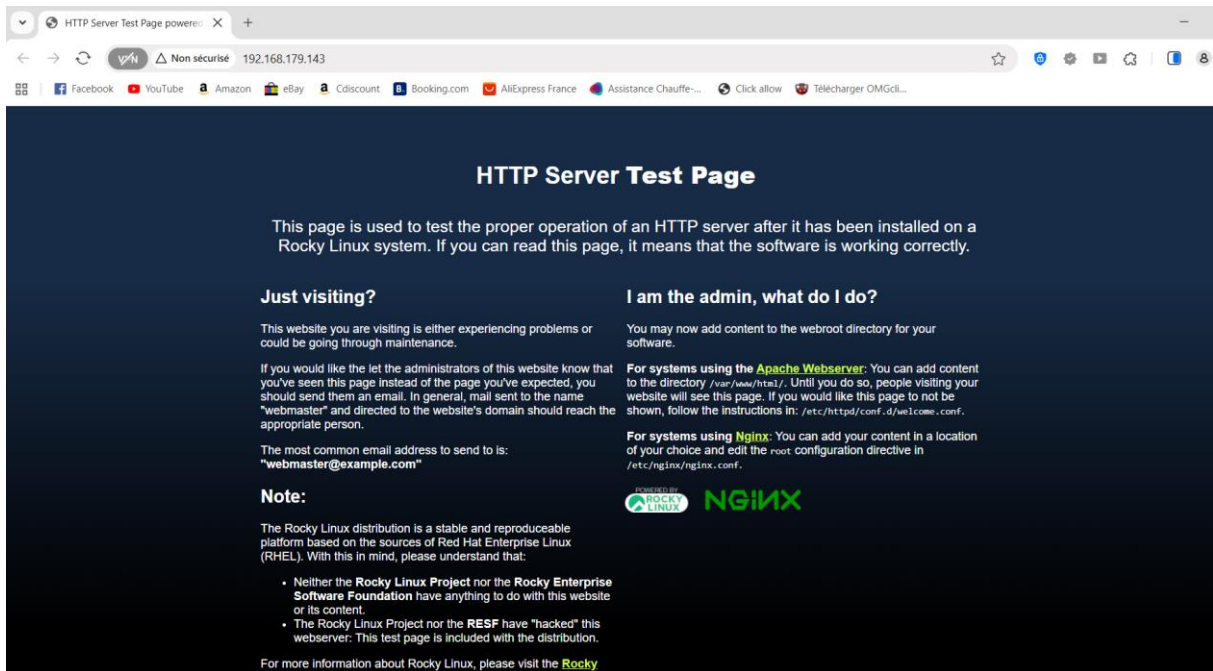
On va commencer par installer nginx, les modules php, mariadb pour la base de données et nano pour faciliter la modification des fichiers. On utilisera la **commande** `dnf install nginx php-fpm mariadb-server php php-mysqld nano`.

```
[root@localhost ~]# dnf install nginx php-fpm mariadb-server php php-mysqld nano
Rocky Linux 9 - BaseOS                               1.7 MB/s | 4.9 MB   00:02
Rocky Linux 9 - AppStream                             8.9 MB/s | 9.7 MB   00:01
Rocky Linux 9 - Extras                               52 kB/s | 16 kB    00:00
Dependencies resolved.
=====
Package                                           Architecture      Version              Repository          Size
=====
Installing:
mariadb-server                                   x86_64            3:10.5.27-1.el9_5.0.2  appstream           9.7 M
nano                                             x86_64            5.6.1-7.el9         baseos              691 k
nginx                                            x86_64            2:1.20.1-24.el9     appstream           36 k
php                                              x86_64            8.0.30-3.el9_6      appstream           8.2 k
php-fpm                                         x86_64            8.0.30-3.el9_6      appstream           1.6 M
php-mysqld                                       x86_64            8.0.30-3.el9_6      appstream           150 k
=====
```

```
[root@localhost ~]# firewall-cmd --permanent --add-service=http
success
[root@localhost ~]# firewall-cmd --reload
success
[root@localhost ~]# firewall-cmd --list-service
cockpit dhcpv6-client http ssh
[root@localhost ~]#
```

```
[root@localhost ~]# sudo systemctl status nginx
● nginx.service - The nginx HTTP and reverse proxy server
   Loaded: loaded (/usr/lib/systemd/system/nginx.service; enabled; preset: disabled)
   Drop-In: /usr/lib/systemd/system/nginx.service.d
            └─php-fpm.conf
   Active: active (running) since Mon 2026-04-06 16:26:41 CEST; 32min ago
     Main PID: 917 (nginx)
       Tasks: 5 (limit: 22909)
      Memory: 7.0M (peak: 7.4M)
         CPU: 32ms
    CGroup: /system.slice/nginx.service
            └─917 "nginx: master process /usr/sbin/nginx"
              └─918 "nginx: worker process"
                └─919 "nginx: worker process"
                  └─920 "nginx: worker process"
                    └─921 "nginx: worker process"

Apr 06 16:26:41 localhost.localdomain systemd[1]: Starting The nginx HTTP and reverse proxy server...
Apr 06 16:26:41 localhost.localdomain nginx[901]: nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
Apr 06 16:26:41 localhost.localdomain nginx[901]: nginx: configuration file /etc/nginx/nginx.conf test is successful
Apr 06 16:26:41 localhost.localdomain systemd[1]: Started The nginx HTTP and reverse proxy server.
```



## Changement du port d'écoute

On modifie le fichier de configuration de Nginx pour changer le port d'écoute de 80 vers 8080. Ça démontre qu'on sait modifier la conf. Ensuite on remet le port 80 pour la suite du TP. Cela permet aussi de tester le serveur sans entrer en conflit avec un autre service qui utiliserait déjà le port 80.

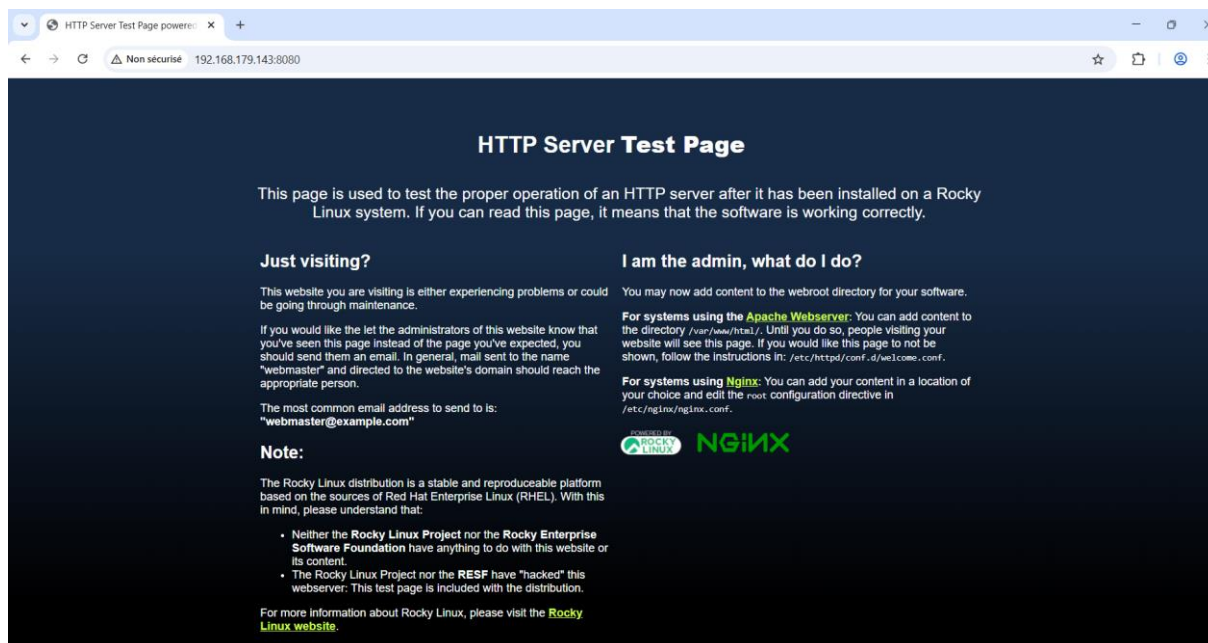
```
# Load modular configuration files from the /etc/nginx/conf.d directory.
# See http://nginx.org/en/docs/nginx_core_module.html#include
# for more information.
include /etc/nginx/conf.d/*.conf;

server {
    listen      80;
    listen     [::]:8080;
    server_name _;
    root       /usr/share/nginx/html;

    # Load configuration files for the default server block.
    include /etc/nginx/default.d/*.conf;
```

```
[root@localhost ~]# grep listen /etc/nginx/nginx.conf | head -5
    listen      8080;
    listen     [::]:8080;
#    listen     443 ssl http2;
#    listen     [::]:443 ssl http2;
[root@localhost ~]#
```

Si j'ouvre le site, je devrai maintenant préciser le port pour accéder à la page. Pour la suite du TP, je rebasculerai sur le port 80.



## Configuration de la base de données :

On commence par allumer de manière permanente le service MariaDB à l'aide de la commande **systemctl enable --now mariadb.service**. Nous pourrions mettre en place MariaDB avec la commande **mysql\_secure\_installation**.

```
[root@localhost //]# mysql_secure_installation

NOTE: RUNNING ALL PARTS OF THIS SCRIPT IS RECOMMENDED FOR ALL MariaDB
SERVERS IN PRODUCTION USE! PLEASE READ EACH STEP CAREFULLY!

In order to log into MariaDB to secure it, we'll need the current
password for the root user. If you've just installed MariaDB, and
haven't set the root password yet, you should just press enter here.

Enter current password for root (enter for none):
OK, successfully used password, moving on...
```

Si une fois l'installation terminée nous ne se sommes pas sur MariaDB, tapez la commande **mysql -u root**.

Une fois dans notre base de données, nous allons créer un site, un utilisateur et nous allons attribuer les

droits sur la base de données à notre utilisateur.

Liste des commandes :

**CREATE DATABASE site\_fr ;**

**CREATE USER 'tenenan'@'localhost' IDENTIFIED BY 'xxxxxxx';**

**GRANT ALL PRIVILEGES ON BDD.\* TO 'tenenan'@'localhost';**

**FLUSH PRIVILEGES;**

**SHOW DATABASES;**

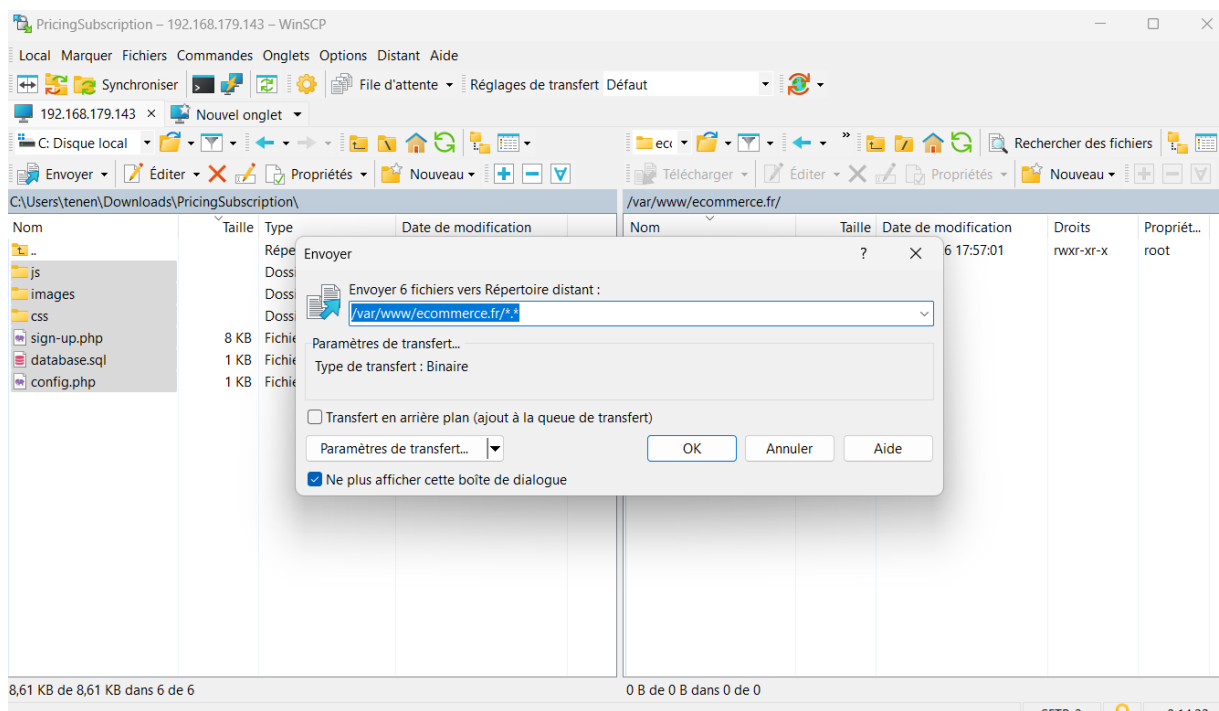
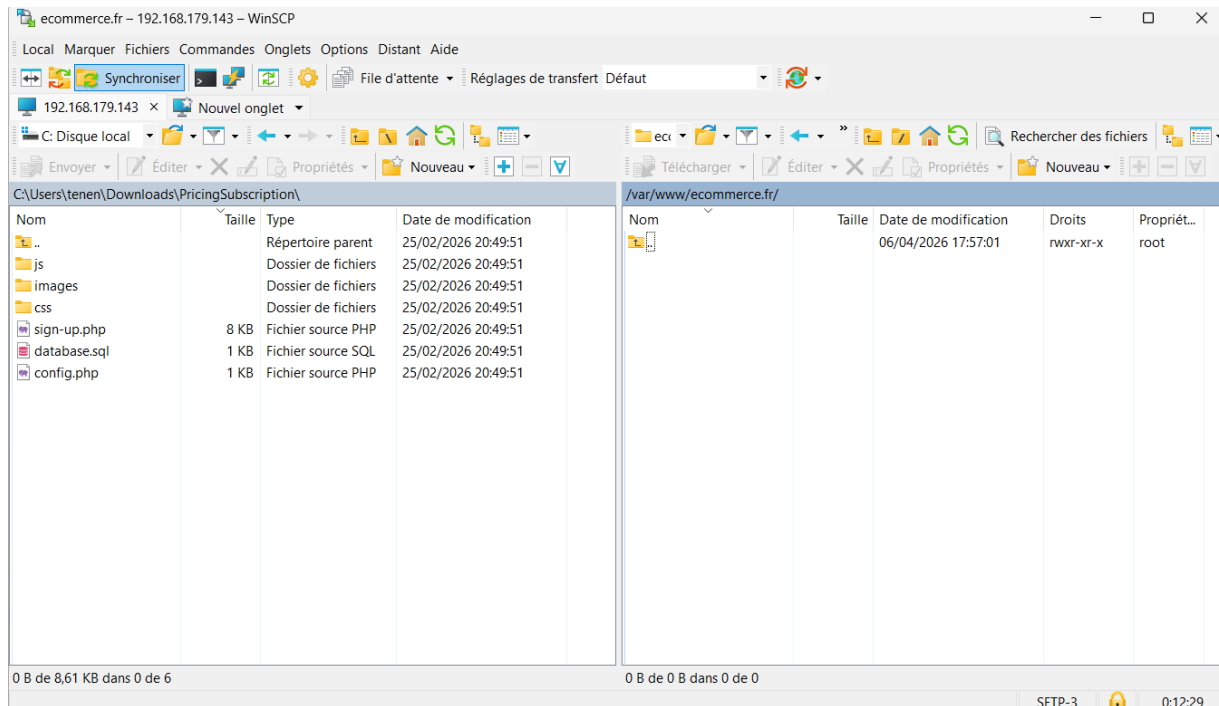
**SHOW TABLES;**

```
[root@localhost ~]# sudo mysql -u root -e "SHOW DATABASES;"
+-----+
| Database           |
+-----+
| information_schema |
| mysql              |
| performance_schema |
| site_fr            |
+-----+
```

```
[root@localhost ~]# sudo mysql -u root -e "SELECT User, Host FROM
mysql.user;"
+-----+-----+
| User          | Host          |
+-----+-----+
| mariadb.sys  | localhost    |
| mysql        | localhost    |
| root         | localhost    |
| tenenan      | localhost    |
+-----+-----+
```

## Mise en place du site FR

Je commence par créer le dossier dans lequel on stockera notre site, on utilisera la commande **mkdir -p /var/www/ecommerce.fr/** puis une fois le dossier créé, je vais y déposer les pages du site à l'aide de **WinSCP**.



Si j'accède à mon dossier avec la commande `cd /var/www/ecommerce.fr` et je peux contrôler que mes fichiers sont bien présents avec la commande `ls`.

```
[root@localhost ~]# cd /var/www/ecommerce.fr
[root@localhost ecommerce.fr]# ls
config.php  css  database.sql  images  js  sign-up.php
[root@localhost ecommerce.fr]#
```

Nous allons modifier les informations du fichier de conf de php pour spécifier un utilisateur qui a accès au dossier `site_fr`. On tapera la commande **nano /etc/php-fpm.d/www.conf**.

```
; Unix user/group of processes
; Note: The user is mandatory. If the group is not set, the default user's group
; will be used.
; RPM: apache user chosen to provide access to the same directories as httpd
user = nginx
; RPM: Keep a group allowed to write in log dir.
group = nginx
```

Nous allons maintenant pouvoir créer un fichier de configuration pour notre site dans lequel on mentionnera le port, le chemin du site, le nom du server ainsi que la page qui apparaît lors de l'accès au site. Nous mentionnerons également les chemins des fichiers de logs et d'autres paramètres PHP. On crée le fichier avec la commande **nano /etc/nginx/conf.d/site\_fr.conf**.

```
server {
    listen 80;
    listen [::]:80;
    root /var/www/ecommerce.fr;
    index index.html index.htm index.nginx-debian.html sign-up.php;
    server_name ecommerce.fr ;
    location ~* \.php$ {
        fastcgi_pass unix:/run/php-fpm/www.sock;
        include fastcgi_params;
        fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
        fastcgi_param SCRIPT_NAME $fastcgi_script_name;
    }
    access_log /var/log/nginx/access_ecommerce.fr.log;
    error_log /var/log/nginx/error_ecommerce.fr.log;
    location / {
        try_files $uri $uri/ =404;
    }
}
```

Dans le cas de notre site, nous allons modifier le fichier config php pour spécifier le nom de la base de données à utiliser ainsi que les identifiants de connexion. On ouvre le fichier avec la commande : **nano /var/www/ecommerce.fr/config.php**.

```
root@localhost:/var/www/ecommerce.fr
GNU nano 5.6.1
?php
["mysql_user"]="tenenan";
["mysql_pass"]="Lmrugby2015";
["hostname"]="localhost";
["mysql_database"]="site_fr";
["data_table"]="registrations";
["paypal_address"]="email@domain.com";
?>
```

Nous retournons ensuite sur MariaDB avec la commande **mysql -u root** puis nous entrons les requêtes suivantes pour créer les tables nécessaires dans la base de données.

```
[root@localhost ecommerce.fr]# mysql -u root
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 21
Server version: 10.5.29-MariaDB MariaDB Server

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

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

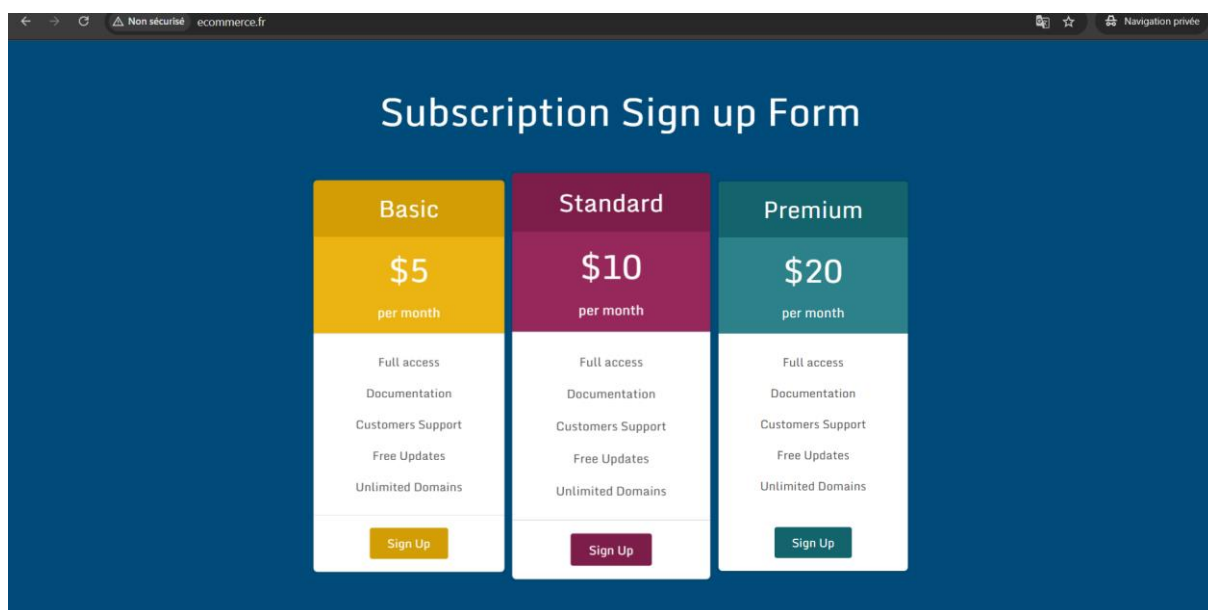
MariaDB [(none)]> USE site_fr;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
MariaDB [site_fr]> CREATE TABLE IF NOT EXISTS `registrations` (
  -> `id` int(11) NOT NULL AUTO_INCREMENT,
  -> `name` varchar(250) NOT NULL,
  -> `email` varchar(250) NOT NULL,
  -> `password` blob,
  -> `phone` varchar(250) NOT NULL,
  -> `plan` enum('basic','standart','premium') DEFAULT NULL,
  -> `date_created` datetime NOT NULL,
  -> `ip` varchar(15) DEFAULT NULL,
  -> UNIQUE (`email`),
  -> PRIMARY KEY (`id`)
  -> ) ENGINE=MyISAM DEFAULT CHARSET=latin1 AUTO_INCREMENT=1 ;
Query OK, 0 rows affected, 1 warning (0.000 sec)
```

Une fois que c'est fait, je peux redémarrer nginx avec la commande **systemctl restart nginx**. Je modifie également mon fichier host pour qu'en entrant ecommerce.fr, l'ip renvoyé est celle de mon serveur.

J'ouvre le blocnote en admin et je vais sur le fichier host.

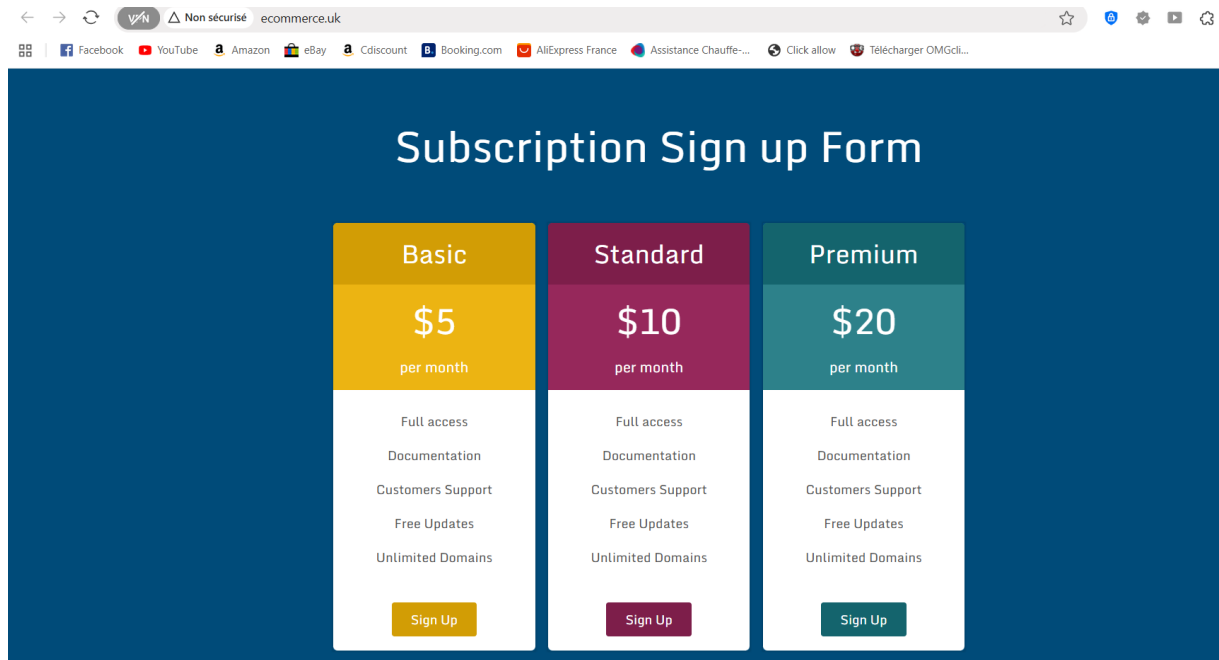
```
# Copyright (c) 1993-2009 Microsoft Corp.
#
# This is a sample HOSTS file used by Microsoft TCP/IP
  for Windows.
#
127.0.0.1      localhost
::1           localhost
192.168.179.143 ecommerce.fr
```



```
[root@localhost ecommerce.fr]# cat /var/www/ecommerce.fr/config.php
sudo mysql -u root -e "SHOW TABLES FROM site_fr;"
<?php
$SETTINGS["mysql_user"]='tenenan';
$SETTINGS["mysql_pass"]='lmrugby2015';
$SETTINGS["hostname"]='localhost';
$SETTINGS["mysql_database"]='site_fr';
$SETTINGS["data_table"]='registrations';
$SETTINGS["paypal_address"]='email@domain.com';
?>+-----+
| Tables_in_site_fr |
+-----+
| registrations    |
+-----+
```

## Mise en place du site UK

Pour la mise en place de ce site, les manipulations sont les mêmes que dans la partie précédente, nous changeons juste le nom des dossiers en remplaçant fr par uk et nous ferons de même pour le nom de domaine, en mettant ecommerce.uk



```
[root@localhost ecommerce.fr]# sudo mysql -u root -e "SHOW DATABASES;"
+-----+
| Database |
+-----+
| information_schema |
| mysql |
| performance_schema |
| site_fr |
| site_uk |
+-----+
```

## Restriction IP

Il est possible de bloquer des IP à la connexion, nous créons un fichier dans lequel je vais répertorier les IP qui ne pourront pas accéder à mon site, j'utilise la commande `nano /etc/nginx/blockip.conf`.

```
[root@localhost ecommerce.fr]# cat /etc/nginx/blockip.conf
deny 192.168.179.1;
```

J'édite ensuite le fichier de conf du site avec la commande :

**nano /etc/nginx/conf.d/site\_fr.conf** pour inclure le fichier créer juste avant.

```
[root@localhost nginx]# nano /etc/nginx/conf.d/site_fr.conf
server {
    listen 80;
    listen [::]:80;
    root /var/www/ecommerce.fr/;
    index index.html index.htm index.nginx-debian.html sign-up.php;
    server_name ecommerce.fr ;
    location ~* \.php$ {
        fastcgi_pass unix:/run/php-fpm/www.sock;
        include fastcgi_params;
        fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
        fastcgi_param SCRIPT_NAME $fastcgi_script_name;
        include blockip.conf;
    }
    access_log /var/log/nginx/access_ecommerce.fr.log;
    error_log /var/log/nginx/error_ecommerce.fr.log;
    location / {
        try_files $uri $uri/ =404;
    }
}
```

Une fois le service redémarrer avec la commande **systemctl restart nginx**, je constate que je n'ai plus accès au site.



## LES LOGS :

Je retrouve l'ensemble de mes logs dans le dossier **cd /var/log/nginx**.

```
[root@localhost ~]# cd /var/log/nginx
[root@localhost nginx]# ls
access.log          access_ecommerce.uk.log  error_ecommerce.fr.log
access_ecommerce.fr.log  error.log                error_ecommerce.uk.log
```



## Test de sécurité

Nous allons mettre en place un test, nous allons envoyer une réponse au formulaire et regarder les trames sur Wireshark.

The screenshot shows the Wireshark interface with the following details:

No.	Time	Source	Destination	Protocol	Length	Info
209	55.198782700	192.168.179.143	192.168.179.1	TCP	66	[TCP Keep-Alive ACK] 80 → 50028 [ACK] Seq=23516
210	55.199006100	192.168.179.143	192.168.179.1	TCP	66	[TCP Dup ACK 41#1] 80 → 58384 [ACK] Seq=1 Ack=1
211	55.199069800	192.168.179.143	192.168.179.1	TCP	66	[TCP Keep-Alive ACK] 80 → 63457 [ACK] Seq=72287
212	57.373026500	192.168.179.1	239.255.255.250	SSDP	179	M-SEARCH * HTTP/1.1
213	60.375458400	192.168.179.1	239.255.255.250	SSDP	179	M-SEARCH * HTTP/1.1
214	60.483990700	192.168.179.1	192.168.179.143	HTTP	799	POST / HTTP/1.1 (application/x-www-form-urlencoded)

Packet 214 details:

```
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,application/javascript;q=0.8,application/signed-exchange;v=b3;q=0.7\r\nReferer: http://ecommerce.fr/\r\nAccept-Encoding: gzip, deflate\r\nAccept-Language: fr-FR,fr;q=0.9,en-US;q=0.8,en;q=0.7\r\n\r\n[Response in frame: 220]\n[Full request URI: http://ecommerce.fr/]\nFile Data: 140 bytes\nHTML Form URL Encoded: application/x-www-form-urlencoded\n  > Form item: "Subscribe" = "1"\n  > Form item: "Plan" = "basic"\n  > Form item: "Price" = "5.00"\n  > Form item: "Name" = "tenenan.chevalier"\n  > Form item: "Email" = "tenenan.chevalier92@gmail.com"\n  > Form item: "Password" = "Motdepasse"\n  > Form item: "Phone" = "0768561835"\n  > Form item: "Terms" = "on"
```

## Mise en place du HTTPS

Les manipulations seront faites sur le site FR, mais seront identiques sur le UK, il faudra remplacer les FR par UK. Je commence par créer un dossier et à restreindre son accès, le dossier stockera la clé du certificat. On fait appel aux commandes :

**mkdir /etc/ssl/private & chmod 700 /etc/ssl/private.**

```
[root@localhost ~]# mkdir /etc/ssl/private\n[root@localhost ~]# chmod 700 /etc/ssl/private
```

Je vais ensuite générer mon certificat à l'aide de commande ci-dessous.

```
openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout
/etc/nginx/ssl/ecommerce.key -out /etc/nginx/ssl/ecommerce.crt
```

Il faudra spécifier quelques informations qui seront visibles sur le certificat.

```
[root@localhost ~]# openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout /etc/nginx/ssl/ecommerce.key -out /etc/nginx/ssl/ecommerce.crt
-----
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) [XX]:FR
State or Province Name (full name) []:Nord
Locality Name (eg, city) [Default City]:Lille
Organization Name (eg, company) [Default Company Ltd]:ecommerce
Organizational Unit Name (eg, section) []:ecommerce
Common Name (eg, your name or your server's hostname) []:ecommerce
Email Address []:tenenan77@gmail.com
-----
[root@localhost ~]#
```

Nous allons maintenant générer les paramètres DH avec la commande

```
openssl dhparam -out /etc/ssl/certs/dhparam.pem 2048.
```

```
[root@localhost ~]# openssl dhparam -out /etc/ssl/certs/dhparam.pem 2048
Generating DH parameters, 2048 bit long safe prime
.+.....
.....
.....+.....
.....
.....+.....
.....
.....+.....
.....
.....+.....
.....
```

Une fois que c'est fait, il faut mettre à jour notre configuration pour avoir une configuration en HTTPS, c'est-à-dire sur le port 443, j'ouvre donc le fichier de conf avec la commande **nano /etc/nginx/conf.d/site\_fr.conf** et rajoute le texte suivant.

```
server {
    listen 443 ssl http2;
    listen [::]:443 ssl http2;
    server_name ecommerce.fr;
    root /var/www/ecommerce.fr;
    index index.html index.htm index.nginx-debian.html sign-up.php;
    ssl_certificate /etc/ssl/certs/nginx-selfsigned.crt;
    ssl_certificate_key /etc/ssl/private/nginx-selfsigned.key;
    ssl_dhparam /etc/ssl/certs/dhparam.pem;
    access_log /var/log/nginx/access_ecommerce.fr.log;
    error_log /var/log/nginx/error_ecommerce.fr.log;
    location / {
        try_files $uri $uri/ =404;
    }
    location ~* \.php$ {
        include fastcgi_params;
        fastcgi_pass unix:/run/php-fpm/www.sock;

        fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
        fastcgi_param SCRIPT_NAME $fastcgi_script_name;
    }
}
```

```

server {
    listen 443 ssl;
    listen [::]:443 ssl;
    ssl_certificate /etc/ssl/certs/nginx-selfsigned.crt;
    ssl_certificate_key /etc/ssl/private/nginx-selfsigned.key;
    ssl_dhparam /etc/ssl/certs/dhparam.pem;

    ssl_protocols TLSv1.2 TLSv1.3;
    ssl_ciphers HIGH:!aNULL:!MD5;

    root /var/www/ecommerce.fr/;
    index index.html index.htm index.nginx-debian.html sign-up.php;
    server_name ecommerce.fr;

    location ~* \.php$ {
        fastcgi_pass unix:/run/php-fpm/www.sock;
        include fastcgi_params;
        fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
        fastcgi_param SCRIPT_NAME $fastcgi_script_name;
        include blockip.conf;
    }

    access_log /var/log/nginx/access_ecommerce.fr.log;
    error_log /var/log/nginx/error_ecommerce.fr.log;

    location / {
        try_files $uri $uri/ =404;
    }
}

```

La commande **nginx -t** va permettre de vérifier que la conf est ok, suite à quoi il faudra redémarrer nginx avec la commande **systemctl restart nginx**.

```

[root@localhost ~]# nginx -t
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginx.conf test is successful
[root@localhost ~]# systemctl restart nginx

```


Il faut également penser à autoriser le port https sur le firewall, nous utiliserons **firewall-cmd --permanent --add-service=https** pour l'ajouter, **firewall-cmd --reload** pour recharger le firewall et **firewall-cmd --list-services** pour vérifier que le service a bien été ajouté.

```

[root@localhost conf.d]# firewall-cmd --permanent --add-service=https
success
[root@localhost conf.d]# firewall-cmd --reload
success
[root@localhost conf.d]# firewall-cmd --list-service
cockpit dhcpv6-client http https ssh
[root@localhost conf.d]#

```

← ↻ 🏠 Non sécurisé <https://ecommerce.fr> ☆ ⚙️ 🔊 ⚙️ Vérifiez q



## Votre connexion n'est pas privée

Les utilisateurs malveillants essaient peut-être de voler vos informations de **ecommerce.fr** (par exemple, les mots de passe, les messages ou les cartes de crédit).

NET:ERR\_CERT\_AUTHORITY\_INVALID

Ce serveur n'a pas pu prouver qu'il s'agit de **ecommerce.fr**. Son certificat de sécurité n'est pas approuvé par le système d'exploitation de votre ordinateur. Cela peut être dû à une mauvaise configuration ou à un utilisateur malveillant qui intercepte votre connexion.

[Continuer vers ecommerce.fr \(non sécurisé\)](#)

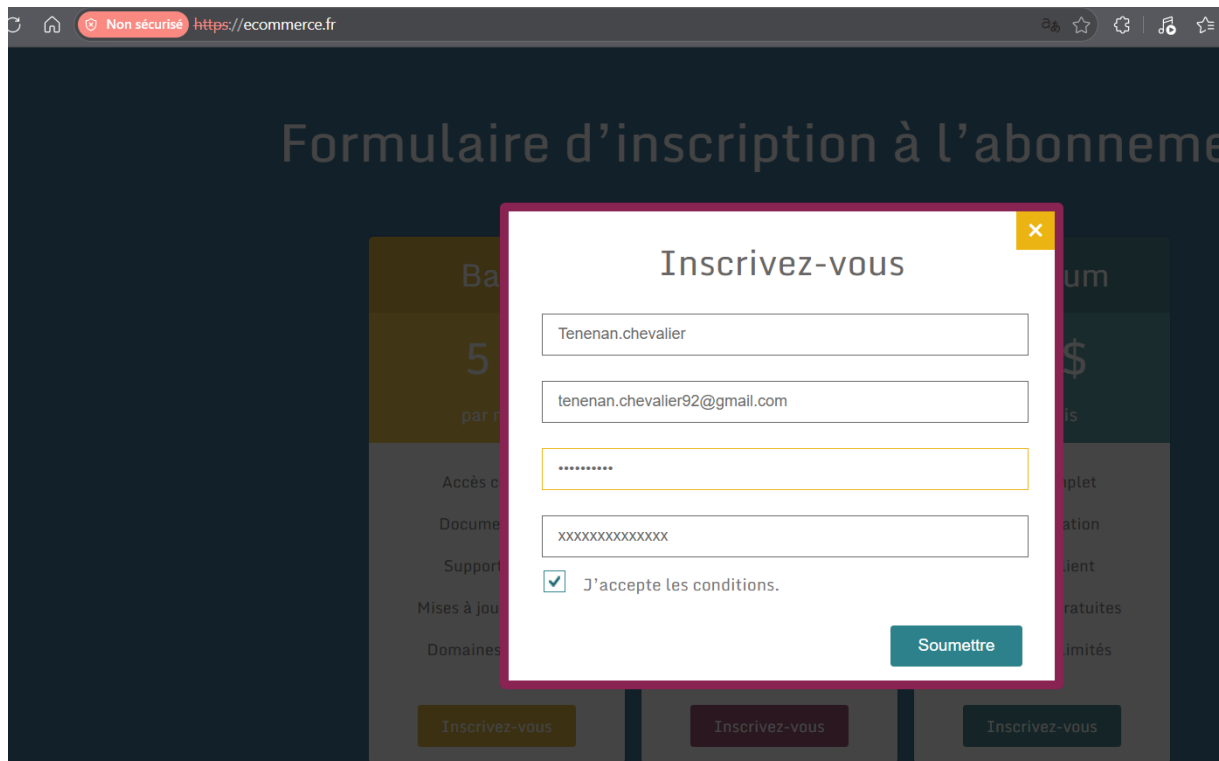
← ↻ 🏠 Non sécurisé <https://ecommerce.fr> Traduit à ☆ ⚙️ 🔊 ⚙️ Vérifiez qu'il s'agit bien

# Formulaire d'inscription à l'abonnement

<b>Base</b>	<b>Standard</b>	<b>Premium</b>
<b>5 \$</b> par mois	<b>10 \$</b> par mois	<b>20 \$</b> par mois
Accès complet Documentation Support client Mises à jour gratuites Domaines illimités	Accès complet Documentation Support client Mises à jour gratuites Domaines illimités	Accès complet Documentation Support client Mises à jour gratuites Domaines illimités
<input type="button" value="Inscrivez-vous"/>	<input type="button" value="Inscrivez-vous"/>	<input type="button" value="Inscrivez-vous"/>

## TEST WIRESHARK

Si on renvoie un formulaire, on voit qu'on est maintenant sur un protocole TLS, qui est chiffré et sécurisé.



\*VMware Network Adapter VMnet8

Fichier Editer Vue Aller Capture Analyser Statistiques Telephonie Wireless Outils Aide

Appliquer un filtre d'affichage ... <Ctrl-/>

No.	Time	Source	Destination	Protocol	Length	Info
104	12.549462000	192.168.179.143	192.168.179.1	TCP	54	443 → 57829 [FIN, ACK] Seq=245 Ack=2137 Win=7001
105	12.550623500	192.168.179.143	192.168.179.1	TLSv1.3	1514	Server Hello, Change Cipher Spec, Application Data
106	12.550630900	192.168.179.143	192.168.179.1	TLSv1.3	66	Application Data
107	12.551227400	192.168.179.1	192.168.179.143	TCP	66	61711 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460
108	12.551354700	192.168.179.143	192.168.179.1	TCP	66	443 → 61711 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0
109	12.555863600	192.168.179.143	192.168.179.1	TCP	1407	[TCP Retransmission] 443 → 63322 [PSH, ACK] Seq=
110	12.556879900	192.168.179.1	192.168.179.143	TCP	54	57829 → 443 [ACK] Seq=2137 Ack=246 Win=131072 Len=0
111	12.556928000	192.168.179.1	192.168.179.143	TCP	66	63322 → 443 [ACK] Seq=3492 Ack=28413 Win=1049600
112	12.556933300	192.168.179.1	192.168.179.143	TCP	54	50686 → 443 [ACK] Seq=1755 Ack=1473 Win=131328 Len=0
113	12.556935200	192.168.179.1	192.168.179.143	TCP	54	61711 → 443 [ACK] Seq=1 Ack=1 Win=131328 Len=0
114	12.557228800	192.168.179.1	192.168.179.143	TLSv1.3	84	Change Cipher Spec, Application Data
115	12.557369100	192.168.179.1	192.168.179.143	TLSv1.3	84	Change Cipher Spec, Application Data
116	12.557622800	192.168.179.1	192.168.179.143	TCP	1514	61711 → 443 [ACK] Seq=1 Ack=1 Win=131328 Len=146
117	12.557622800	192.168.179.1	192.168.179.143	TLSv1.3	348	Client Hello (SNI=ecommerce.fr)
118	12.557724500	192.168.179.1	192.168.179.143	TLSv1.3	815	Application Data
119	12.557755900	192.168.179.1	192.168.179.143	TCP	54	52960 → 443 [FIN, ACK] Seq=2104 Ack=245 Win=1310

```

> Frame 106: Packet, 66 bytes on wire (528 bits), 66 bytes captured on interface
> Ethernet II, Src: VMware_1c:5f:2b (00:0c:29:1c:5f:2b), Dst: VMware_1c:5f:2b (00:0c:29:1c:5f:2b)
> Internet Protocol Version 4, Src: 192.168.179.143, Dst: 192.168.179.1
> Transmission Control Protocol, Src Port: 443, Dst Port: 50686, Seq: 63322, Len: 66
> [2 Reassembled TCP Segments (58 bytes): #105(46), #106(12)]
> Transport Layer Security
0000 00 50 56 c0 00 08 00 0c 29 1c 5f 2b 08 00 45 00  .PV...
0010 00 34 ae 1f 40 00 40 06 a4 c2 c0 a8 b3 8f c0 a8  .4..@.
0020 b3 01 01 bb c5 fe d0 69 43 81 aa 6c d8 ff 50 18  .-...
0030 02 23 68 2b 00 00 d6 84 cc 86 f4 ff 18 20 9a ec  .:ht...
0040 b3 66  .f

```

## Redirection HTTP vers HTTPS

Par sécurité, nous allons activer une redirection pour que quand une personne se connecte en http, elle soit automatiquement redirigée en HTTPS. Il faut modifier le fichier de conf avec la commande `nano /etc/nginx/conf.d/site_fr.conf` et ajouter dans la configuration http la ligne `return 301 https://$host$request_util/;`

```

server {
    return 301 https://$host$request_uti/;
    listen 80;
    listen [::]:80;
    server_name ecommerce.fr;
    return 301 https://$host$request_uri;
}

server {
    listen 443 ssl;
    listen [::]:443 ssl;
    ssl_certificate /etc/ssl/certs/nginx-selfsigned.crt;
    ssl_certificate_key /etc/ssl/private/nginx-selfsigned.key;
    ssl_dhparam /etc/ssl/certs/dhparam.pem;

    ssl_protocols TLSv1.2 TLSv1.3;
    ssl_ciphers HIGH:!aNULL:!MD5;

    root /var/www/ecommerce.fr/;
    index index.html index.htm index.nginx-debian.html sign-up.php;
    server_name ecommerce.fr;

    location ~* \.php$ {
        fastcgi_pass unix:/run/php-fpm/www.sock;
        include fastcgi_params;
        fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
        fastcgi_param SCRIPT_NAME $fastcgi_script_name;
        include blockip.conf;
    }

    access_log /var/log/nginx/access_ecommerce.fr.log;
    error_log /var/log/nginx/error_ecommerce.fr.log;

    location / {
        try_files $uri $uri/ =404;
    }
}

```

Je vérifie ma configuration avec la commande **nginx -t** et une fois qu'elle est ok, je redémarre nginx avec la commande **systemctl restart nginx**.

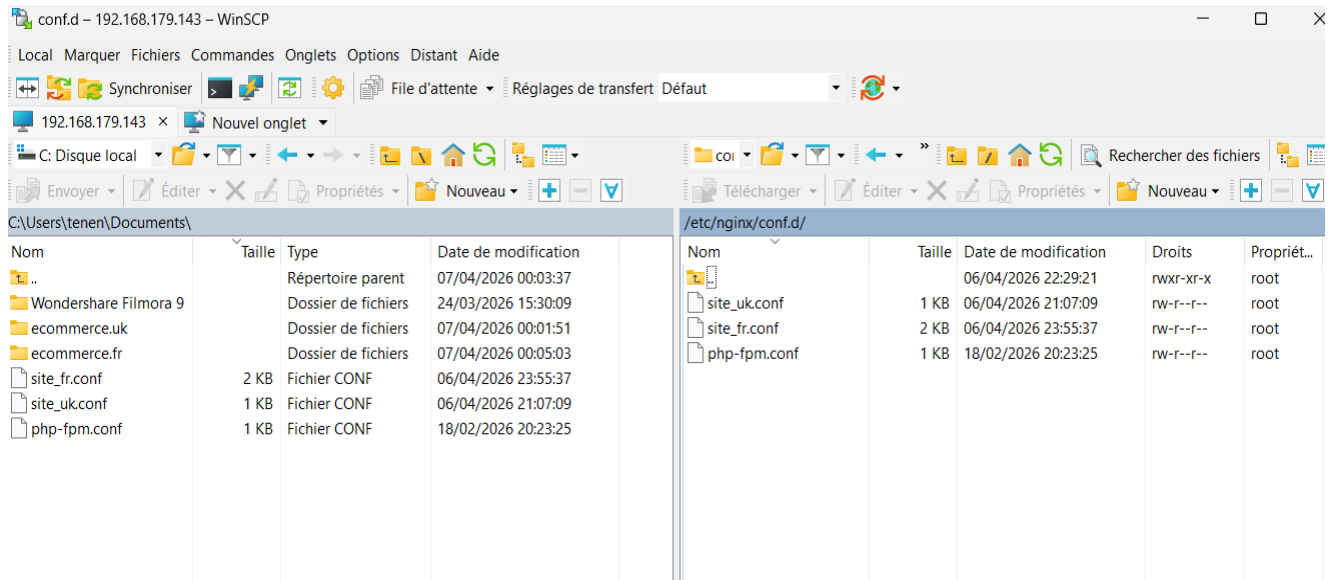
```

[root@localhost conf.d]# nginx -t
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginx.conf test is successful
[root@localhost conf.d]# systemctl restart nginx

```

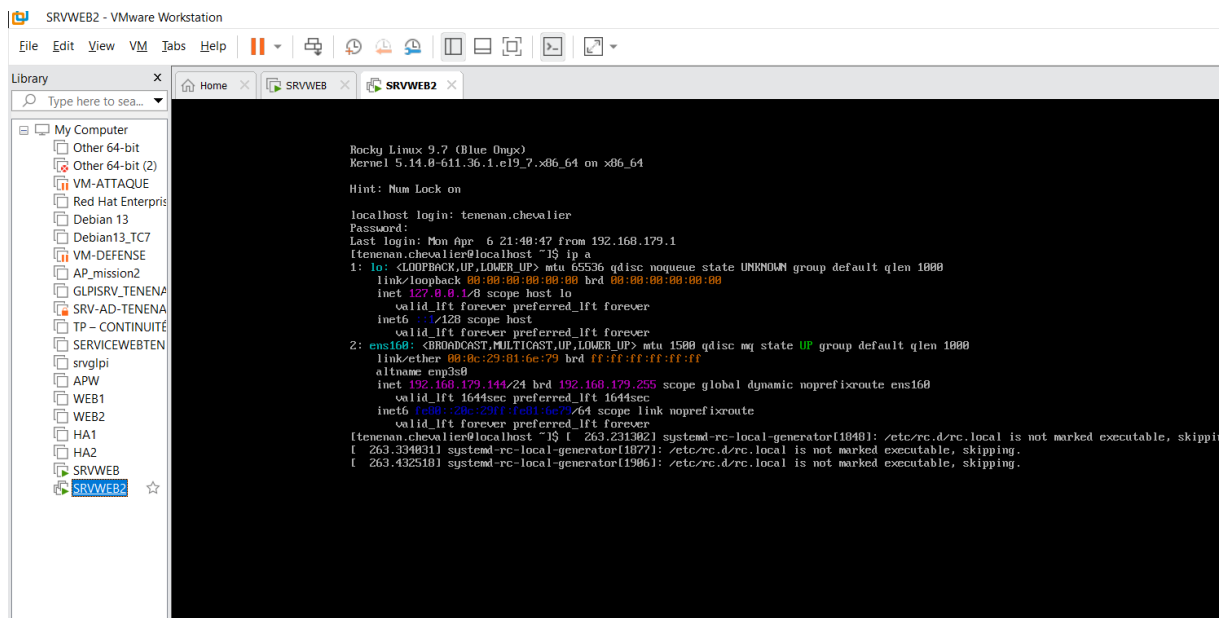
## Mise en place d'un deuxième serveur web

Sur une Rocky linux, une fois nginx installé, avec la commande **dnf install nginx**, je vais copier-coller les fichiers en les plaçant au même endroit sur sur srvweb1. Il faudra copier le fichier de conf, situé dans **/etc/nginx/conf.f/site\_fr.conf** et le dossier contenant le site, au chemin **/var/www/ecommerce.fr**.



Une fois les fichiers copier, je vais éditer le fichier de configuration PHP pour y spécifier l'IP du serveur web 1, car le serveur web 2 utilisera la base de données du serveur web 2.

Je clone le SRVWEB1 au SRVWEB2 :



Je me connecte à l'aide PuTTY au SRVWEB2.

Systemctl status nginx sur le SRVWEB2 :

```
[tenenan.chevalier@localhost ~]$ systemctl status nginx
● nginx.service - The nginx HTTP and reverse proxy server
   Loaded: loaded (/usr/lib/systemd/system/nginx.service; enabled; preset: disabled)
   Drop-In: /usr/lib/systemd/system/nginx.service.d
            └─php-fpm.conf
   Active: active (running) since Tue 2026-04-07 00:15:39 CEST; 4min 7s ago
 Main PID: 1827 (nginx)
   Tasks: 5 (limit: 22909)
  Memory: 5.7M (peak: 5.9M)
     CPU: 23ms
  CGroup: /system.slice/nginx.service
          └─1827 "nginx: master process /usr/sbin/nginx"
            └─1828 "nginx: worker process"
              └─1829 "nginx: worker process"
                └─1830 "nginx: worker process"
                  └─1831 "nginx: worker process"

Apr 07 00:15:39 srvweb2 systemd[1]: Starting The nginx HTTP and reverse proxy server: [OK]
Apr 07 00:15:39 srvweb2 nginx[1825]: nginx: the configuration file /etc/nginx/nginx.conf is not open (yet)
Apr 07 00:15:39 srvweb2 nginx[1825]: nginx: configuration file /etc/nginx/nginx.conf successfully loaded
Apr 07 00:15:39 srvweb2 systemd[1]: Started The nginx HTTP and reverse proxy server: [OK]
lines 1-20/20 (END)
● nginx.service - The nginx HTTP and reverse proxy server
```

Je vais également spécifier le nom et le mot de passe du compte qui aura accès au serveur web 2. J'utilise la commande **nano /var/www/ecommerce.fr/config.php**.

```
[tenenan.chevalier@localhost ~]$ cat /var/www/ecommerce.fr/config.php
<?php
$SETTINGS["mysql_user"]='tenenan';
$SETTINGS["mysql_pass"]='Lmrugby2015';
$SETTINGS["hostname"]='192.168.179.143';
$SETTINGS["mysql_database"]='site_fr';
$SETTINGS["data_table"]='registrations';
$SETTINGS["paypal_address"]='email@domain.com';
?>[tenenan.chevalier@localhost ~]$
```

Il faudra également ouvrir le port 3306 sur le serveur web 1 avec la commande :

**firewall-cmd --permanent --add-port=3306/tcp** et recharger le firewall avec la commande **firewall-cmd --reload**.

```
[root@localhost conf.d]# firewall-cmd --permanent --add-port=3306/tcp
success
[root@localhost conf.d]# firewall-cmd --reload
success
```

Modifier le fichier mariadb avec la commande `nano /etc/my.cnf.d/mariadb-server.cnf` pour autoriser la communication extérieure.

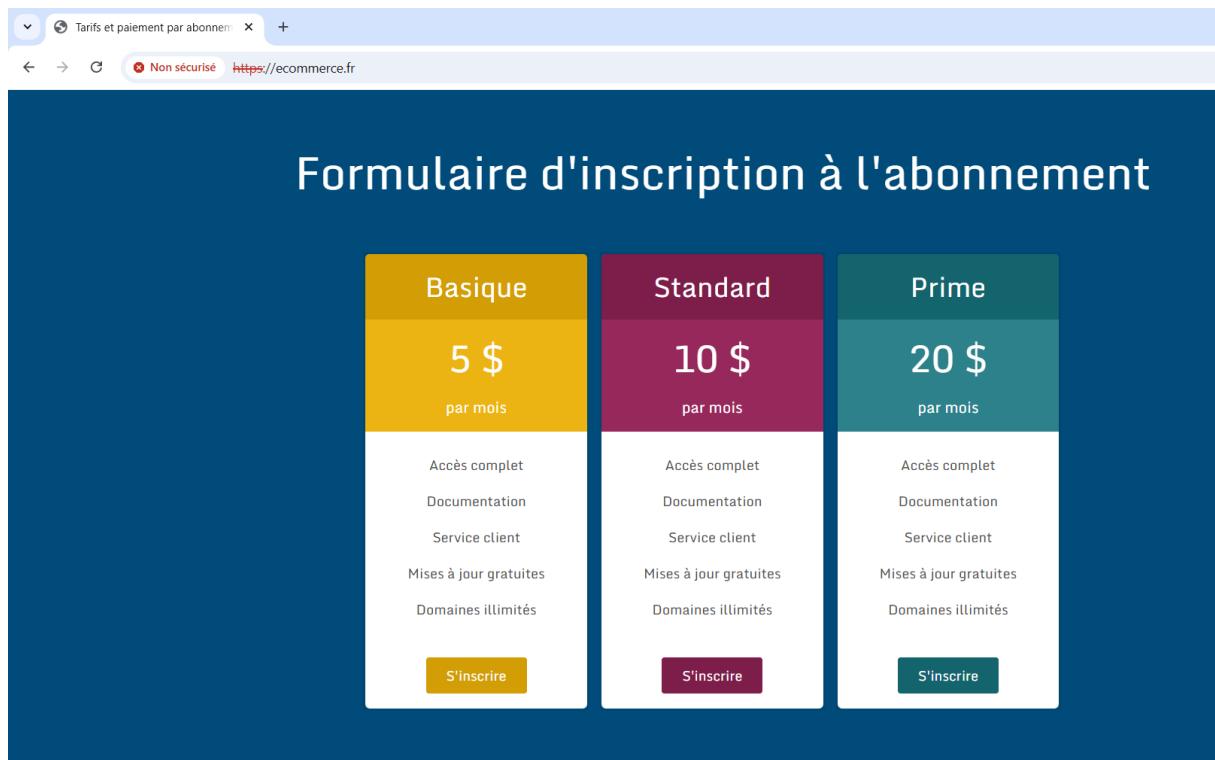
```
[galera]
# Mandatory settings
#wsrep_on=ON
#wsrep_provider=
#wsrep_cluster_address=
#binlog_format=row
#default_storage_engine=InnoDB
#innodb_autoinc_lock_mode=2
#
# Allow server to accept connections on all interfaces.
#
bind-address=0.0.0.0
#
# Optional setting
```

Il faudra aussi créer un autre utilisateur sur la base de données qui aura accès à notre serveur web 2, et lui donner les droits sur les différents sites.

```
[root@localhost ~]# sudo mysql -e "SHOW GRANTS FOR
'tenenan'@'192.168.179.144';"
+-----+
| Grants for tenenan@192.168.179.144 |
+-----+
| GRANT USAGE ON *.* TO `tenenan`@`192.168.179.144` IDENTIFIED BY PASSWORD '*56A54F74ED891816D9D82579969720D4F339A2C9' |
| GRANT ALL PRIVILEGES ON `site_fr`.* TO `tenenan`@`192.168.179.144` |
| GRANT ALL PRIVILEGES ON `site_uk`.* TO `tenenan`@`192.168.179.144` |
+-----+
```

Il faudra remettre en place le HTTPS (voir table des matières). En modifiant l'IP présente dans mon fichier hosts, je remarque les sites sont bien fonctionnels.

```
# Copyright (c) 1993-2009 Microsoft Corp.
#
# This is a sample HOSTS file used by Microsoft TCP/IP for Windows.
#
# This file contains the mappings of IP addresses to host names. Each
# entry should be kept on an individual line. The IP address should
# be placed in the first column followed by the corresponding host name.
# The IP address and the host name should be separated by at least one
# space.
#
# Additionally, comments (such as these) may be inserted on individual
# lines or following the machine name denoted by a '#' symbol.
#
# For example:
#
# 102.54.94.97 rhino.acme.com # source server
# 38.25.63.10 x.acme.com # x client host
#
# localhost name resolution is handled within DNS itself.
# 127.0.0.1 localhost
# ::1 localhost
192.168.145.129 ecommerce.uk
192.168.145.129 ecommerce.fr
```



## Mise en place du serveur HAProxy

Je commence par installer HAProxy avec la commande **yum install haproxy**.

```
assw@localhost:~$ yum install haproxy
Last metadata expiration check: 0:30:06 ago on Tue Apr  7 00:39:24 :
Dependencies resolved.
=====
Package                               Architecture  Version
=====
Installing:
haproxy                                x86_64        2.8.14-1.el9_7.1
```

On va faire une copie du fichier de conf avec la commande **cp /etc/haproxy/haproxy.cfg /etc/haproxy/copie-haproxy.cfg**. On pourra ensuite modifier le fichier avec la commande **nano /etc/haproxy/haproxy.cfg**.

On mentionne dans le fichier de configuration les IP des serveurs dans l'ordre que l'on veut les utiliser en précisant **le port 443 car nos serveurs sont en HTTPS**. Je vais également modifier le **port de frontend pour mettre le 80**, qui correspond au port http. Je passe également du mode http au **mode TCP** pour pouvoir charger mes css correctement.

```
SRV1 x HAPROXY1 x + v
Running scriptlet: haproxy-2.8.14-1.el9_7.1.x86_64 1/1
Verifying      : haproxy-2.8.14-1.el9_7.1.x86_64 1/1

Installed:
haproxy-2.8.14-1.el9_7.1.x86_64

Complete!
[root@localhost tenenan.chevalier]# cp /etc/haproxy/haproxy.cfg /etc/haproxy/copie-haproxy.cfg.
[root@localhost tenenan.chevalier]# ls
[root@localhost tenenan.chevalier]# nano /etc/haproxy/haproxy.cfg
[root@localhost tenenan.chevalier]# systemctl status haproxy
● haproxy.service - HAProxy Load Balancer
   Loaded: loaded (/usr/lib/systemd/system/haproxy.service; enabled; preset: disabled)
   Active: active (running) since Tue 2026-04-07 01:38:49 CEST; 3min 13s ago
   Process: 2785 ExecStartPre=/usr/sbin/haproxy -f $CONFIG -f $CFGDIR -c -q $OPTIONS (code=exited, status=0/SUCCESS)
  Main PID: 2787 (haproxy)
    Status: "Ready."
     Tasks: 5 (Limit: 22909)
  Memory: 44.5M (peak: 45.1M)
     CPU: 358ms
    CGroup: /system.slice/haproxy.service
            └─2787 /usr/sbin/haproxy -Ws -f /etc/haproxy/haproxy.cfg -f /etc/haproxy/conf.d -p /run/haproxy.pid
              └─2789 /usr/sbin/haproxy -Ws -f /etc/haproxy/haproxy.cfg -f /etc/haproxy/conf.d -p /run/haproxy.pid

Apr 07 01:38:49 haproxy systemd[1]: Starting HAProxy Load Balancer...
Apr 07 01:38:49 haproxy haproxy[2787]: [NOTICE] (2787) : New worker (2789) forked
Apr 07 01:38:49 haproxy haproxy[2787]: [NOTICE] (2787) : Loading success.
Apr 07 01:38:49 haproxy systemd[1]: Started HAProxy Load Balancer.
Apr 07 01:40:07 haproxy haproxy[2789]: [ALERT] (2789) : sendmsg()/writev() failed in logger #1: No such file or dire
```

```
backend web_servers
    balance roundrobin
    option httpchk GET /
    server srvweb1 192.168.179.143:443 ssl verify none check
    server srvweb2 192.168.179.144:443 ssl verify none check
```

```
defaults
    log global
    mode tcp
    option tcplog
    option dontlognull
    timeout connect 5000ms
    timeout client 50000ms
    timeout server 50000ms

frontend http_front
    bind *:80
    default_backend web_servers
```

```
frontend http_front
    bind *:80
    default_backend web_servers
```

Nous pourrions également ajouter la configuration pour l'accès à l'interface d'administration en ajoutant le texte ci-dessous dans la partie « Default common » de la conf.

```
listen stats
bind *:8080
mode http
stats enable
stats hide-version
stats uri /stats
stats admin if LOCALHOST
stats auth haproxy:haproxy
```

```
defaults
    log      global
    mode     tcp
    option   tcplog
    option   dontlognull
    timeout  connect 5000ms
    timeout  client  50000ms
    timeout  server  50000ms

frontend http_front
    bind *:80
    default_backend web_servers

backend web_servers
    balance roundrobin
    server srvweb1 192.168.179.143:443 check
    server srvweb2 192.168.179.144:443 check

listen stats
    bind *:8080
    mode http
    stats enable
    stats hide-version
    stats uri /stats
    stats admin if LOCALHOST
    stats auth haproxy:haproxy
```

Nous allons ensuite avec la commande **setsebool -P haproxy\_connect\_any 1** autoriser HAProxy à se connecter vers n'importe quel port distant.

```
[root@localhost haproxy]# setsebool -P haproxy_connect_any 1
```

On va maintenant modifier le fichier de conf rsyslog avec la commande **nano /etc/rsyslog.conf** et nous allons décommenter deux lignes.

```
# rsyslog configuration file

# For more information see /usr/share/doc/rsyslog-*/rsyslog_conf.html
# or latest version online at http://www.rsyslog.com/doc/rsyslog_conf.html
# If you experience problems, see http://www.rsyslog.com/doc/troubleshoot.html

#### GLOBAL DIRECTIVES ####

# Where to place auxiliary files
global(workDirectory="/var/lib/rsyslog")

# Use default timestamp format
module(load="builtin:omfile" Template="RSYSLOG_TraditionalFileFormat")

#### MODULES ####

module(load="imuxsock" # provides support for local system logging (e.g. via logger command)
        SysSock.Use="off") # Turn off message reception via local log socket;
                          # local messages are retrieved through imjournal now.
module(load="imjournal" # provides access to the systemd journal
        UsePid="system" # PID number is retrieved as the ID of the process the journal entry originates from
        FileCreateMode="0644" # Set the access permissions for the state file
        StateFile="imjournal.state") # File to store the position in the journal
#module(load="imklog") # reads kernel messages (the same are read from journald)
#module(load="immark") # provides --MARK-- message capability

# Include all config files in /etc/rsyslog.d/
include(file="/etc/rsyslog.d/*.conf" mode="optional")

# Provides UDP syslog reception
# for parameters see http://www.rsyslog.com/doc/imudp.html
module(load="imudp") # needs to be done just once
input(type="imudp" port="514")

# Provides TCP syslog reception
# for parameters see http://www.rsyslog.com/doc/mtcp.html
#module(load="imtcp") # needs to be done just once
```

On va ensuite pouvoir démarrer les services haproxy et rsyslog avec la commande **systemctl restart haproxy && systemctl restart rsyslog**.

```
[root@haproxy tenenan.chevalier]# systemctl restart haproxy && systemctl restart rsyslog
```

Pour accéder à l'interface graphique et vérifier que tout est ok, je vais autoriser le port http avec la commande **firewall-cmd --permanent --add-service=http** et je vais également autoriser le port 8080 pour la console d'administration avec la commande **firewall-cmd --permanent --add-port=8080/tcp**. Je recharge le firewall avec la commande **firewall-cmd --reload** pour appliquer mes ouvertures de port.

```
[root@haproxy tenenan.chevalier]# firewall-cmd --permanent --add-service=http
Warning: ALREADY_ENABLED: http
success
[root@haproxy tenenan.chevalier]# firewall-cmd --permanent --add-port=8080/tcp
Warning: ALREADY_ENABLED: 8080:tcp
success
```



Une fois le certificat prêt, je modifie le fichier de configuration HAProxy avec la commande **nano /etc/haproxy/haproxy.cfg** et je modifie le port d'écoute en précisant le chemin de mon certificat et j'ajoute une redirection automatique.

```

frontend http_front
  bind *:80
  mode http
  redirect scheme https code 301

frontend https_front
  bind *:443 ssl crt /etc/ssl/haproxy/haproxy.pem
  default_backend web_servers
  
```

<http://ecommerce.fr> est redirigé vers <https://ecommerce.fr> automatiquement.

**HAProxy Statistics Report for pid 3546**

> General process information

pid = 3546 (process #1, nbproc = 1, nbthread = 4)  
 uptime = 0d 0h3m41s, warnings = 0  
 system limits: memmax = unlimited, ulimit-n = 524288  
 maxsock = 524288, maxconns = 262125, reached = 0, maxpipes = 0  
 current conn = 3, current pipes = 0/0, conn rate = 1/sec, bit rate = 6.253 kbps  
 Running tasks: 0/25, idle = 100 %

Legend: active UP, active UP, going down, active DOWN, going up, active or backup DOWN, active or backup DOWN for maintenance (MAINT), active or backup SOFT STOPPED for maintenance, Note: "NOLEAFDRAIN" = UP with loadbalancing disabled

Display options: Scope, Hide DOWN servers, Refresh now, CSV export, JSORL export (schema)

External resources: Primary site, Updates (v2.0), Online manual

Queue		Session rate			Sessions				Bytes		Denied		Errors		Warnings		Status		Server									
Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Conn	Resp	Retr	Redis	OPEN	LastChk	Wght	Act	Bck	Chk	Dwn	Downtime	Thrtle
0	1	-	1	1	1	262	125	1	1	1	12s	0	0	0	0	0	0	0	0	3m42s UP	LOCK in 2ms	1/1	Y	-	0	0	0s	-

Une fois que c'est fait, je peux redémarrer HAProxy avec la commande **systemctl restart haproxy**.

## Mise en place d'un deuxième HAProxy

Pour la mise en place du deuxième HAProxy, on reprendra les étapes de l'installation et de la sécurisation du premier serveur HAProxy.

## Mise en place de PaceMaker - Cluster

Sur chaque serveur, je vais modifier le fichier hosts à l'aide de la commande **nano /etc/hosts**, pour y spécifier les IP de mes différents serveurs ainsi que leurs noms, pour pouvoir faire la résolution.

```

[tenenan.chevalier@localhost ~]$ cat /etc/hosts
127.0.0.1 localhost localhost.localdomain localhost4 localhost4.localdomain4
::1 localhost localhost.localdomain localhost6 localhost6.localdomain6
192.168.179.143 srvweb1
192.168.179.144 srvweb2
192.168.179.145 haproxy
192.168.179.146 haproxy2
  
```

Je vais ensuite activer sur les serveurs HAProxy le répertoire « Highavailability » avec la commande **dnf config-manager --set-enabled highavailability** pour pouvoir ensuite installer pacemaker avec la commande **dnf install pacemaker pcs**.

```
[root@haproxy tenenan.chevalier]# dnf config-manager --set-enabled highavailability
[root@haproxy tenenan.chevalier]# dnf install pacemaker pcs
Rocky Linux 9 - High Availability
Last metadata expiration check: 0:00:01 ago on Tue Apr 7 02:49:47 2026.
Dependencies resolved.
=====
Package                                Architecture      Version
=====
Installing:
pacemaker                               x86_64            2.1.10-1.
pcs                                      x86_64            0.11.10-1.
Installing dependencies:
bzip2                                    x86_64            1.0.8-10.
```

Une fois que c'est fait, j'active le service de manière permanente avec la commande

**systemctl enable --now pcsd**.

```
[root@haproxy tenenan.chevalier]# systemctl enable --now pcsd
Created symlink /etc/systemd/system/multi-user.target.wants/p
[root@haproxy tenenan.chevalier]# |
```

Je vais définir un mot de passe pour le compte « hacluster » avec la **commande passwd hacluster**

```
[root@haproxy tenenan.chevalier]# passwd hacluster
Changing password for user hacluster.
New password:
Retype new password:
Sorry, passwords do not match.
New password:
Retype new password:
passwd: all authentication tokens updated successfully.
[root@haproxy tenenan.chevalier]# |
```

Ensuite, j'autorise sur le firewall le service High-Availability et je recharge ce dernier. On le fera avec les commandes **firewall-cmd --add-service=high-availability --permanent** et **firewall-cmd reload**

```
[root@haproxy tenenan.chevalier]# sudo firewall-cmd --list-services
cockpit dhcpv6-client high-availability http https mysql ssh
```

A cette étape, on effectuera les commandes sur un seul serveur HAProxy uniquement. Je vais faire le lien entre mes deux nœuds (hôtes) et le hacluster avec la commande **pcs auth haproxy1 haproxy2**.

```
[root@haproxy tenenan.chevalier]# pcs host auth haproxy haproxy2 -u hacluster -p Lmrugby2015
haproxy: Authorized
haproxy2: Authorized
```

Une fois que c'est bon, je lance la configuration de mon cluster avec la commande

**pcs cluster setup ha\_cluster haproxy1 haproxy2.**

```
[root@haproxy tenenan.chevalier]# pcs cluster setup ha_cluster haproxy haproxy2
No addresses specified for host 'haproxy', using 'haproxy'
No addresses specified for host 'haproxy2', using 'haproxy2'
Destroying cluster on hosts: 'haproxy', 'haproxy2'...
haproxy2: Successfully destroyed cluster
haproxy: Successfully destroyed cluster
Requesting remove 'pcsd settings' from 'haproxy', 'haproxy2'
haproxy: successful removal of the file 'pcsd settings'
haproxy2: successful removal of the file 'pcsd settings'
Sending 'corosync authkey', 'pacemaker authkey' to 'haproxy', 'haproxy2'
haproxy: successful distribution of the file 'corosync authkey'
haproxy2: successful distribution of the file 'pacemaker authkey'
haproxy2: successful distribution of the file 'corosync authkey'
haproxy2: successful distribution of the file 'pacemaker authkey'
Sending 'corosync.conf' to 'haproxy', 'haproxy2'
haproxy: successful distribution of the file 'corosync.conf'
haproxy2: successful distribution of the file 'corosync.conf'
Cluster has been successfully set up.
```

Je démarre mon cluster avec la commande **sudo pcs cluster start --all** et l'active au démarrage avec la commande **sudo pcs cluster enable --all**.

```
[root@haproxy tenenan.chevalier]# sudo pcs cluster start --all
haproxy: Starting Cluster...
haproxy2: Starting Cluster...
```

```
[root@haproxy tenenan.chevalier]# sudo pcs cluster enable --all
haproxy: Cluster Enabled
haproxy2: Cluster Enabled
```

On pourra vérifier l'état du cluster avec les commandes **sudo pcs cluster status**, **sudo pcs status corosync**, **sudo pcs status cluster**, **sudo pcs status nodes**.

```
[root@haproxy tenenan.chevalier]# sudo pcs cluster status
Cluster Status:
Cluster Summary:
 * Stack: corosync (Pacemaker is running)
 * Current DC: haproxy (version 2.1.10-1.1.el9_7-5693eae) - partition with quorum
 * Last updated: Tue Apr 7 03:28:42 2026 on haproxy
 * Last change: Tue Apr 7 03:24:51 2026 by hacluster via hacluster on haproxy
 * 2 nodes configured
 * 0 resource instances configured
Node List:
 * OnLine: [ haproxy haproxy2 ]

PCSD Status:
 haproxy: Online
 haproxy2: Online
```

```
[root@haproxy tenenan.chevalier]# sudo pcs status corosync
```

Membership information

```
-----
Nodeid      Votes Name
   1         1 haproxy (local)
   2         1 haproxy2
```

```
[root@haproxy tenenan.chevalier]# sudo pcs status nodes
```

Pacemaker Nodes:

```
Online: haproxy haproxy2
Standby:
Standby with resource(s) running:
Maintenance:
Offline:
```

Pacemaker Remote Nodes:

```
Online:
Standby:
Standby with resource(s) running:
Maintenance:
Offline:
```

Je désactive ensuite stonith, pour éviter que tous nos nœuds fournissent le même service ou plus aucun. On redémarrera ensuite le cluster.

**Liste des commandes:**

```
sudo pcs property set stonithenabled=false
```

```
sudo pcs property set no-quorumpolicy=ignore
```

```
sudo pcs cluster stop --all
```

```
sudo pcs cluster start --all
```

```
[root@haproxy tenenan.chevalier]# sudo pcs property set stonith-enabled=false
[root@haproxy tenenan.chevalier]# sudo pcs property set no-quorum-policy=ignore
[root@haproxy tenenan.chevalier]# sudo pcs cluster stop --all
haproxy2: Stopping Cluster (pacemaker)...
haproxy: Stopping Cluster (pacemaker)...
haproxy2: Stopping Cluster (corosync)...
haproxy: Stopping Cluster (corosync)...
[root@haproxy tenenan.chevalier]# sudo pcs cluster start --all
haproxy2: Starting Cluster...
haproxy: Starting Cluster...
[root@haproxy tenenan.chevalier]# |
```

```
[root@haproxy tenenan.chevalier]# crm_verify -L -V
```

En vérifiant les propriétés avec la commande **pcs property**, on voit que c'est bien désactiver.

```
[root@haproxy tenenan.chevalier]# pcs property
Cluster Properties: cib-bootstrap-options
cluster-infrastructure=corosync
cluster-name=mycluster
dc-version=2.1.10-1.1.el9_7-5693eae
have-watchdog=false
no-quorum-policy=ignore
stonith-enabled=false
```

Nous allons maintenant créer une IP virtuelle pour faire le lien entre nos deux serveurs avec la commande **pcs resource create virtual\_ip ocf:heartbeat:IPaddr2 ip=192.168.179.150 cidr\_netmask=24 op monitor interval=30s**.

```
[root@haproxy tenenan.chevalier]# sudo pcs resource create virtual_ip ocf:heartbeat:IPaddr2 ip=192.168.179.150 cidr_netmask=24 op monitor interval=30sudo p
cs resource create virtual_ip
```

On peut vérifier que l'IP a bien été créée avec la commande **pcs status resources** et on peut la voir avec la commande **ip a**. On voit également qu'elle est portée par le serveur haproxy1.

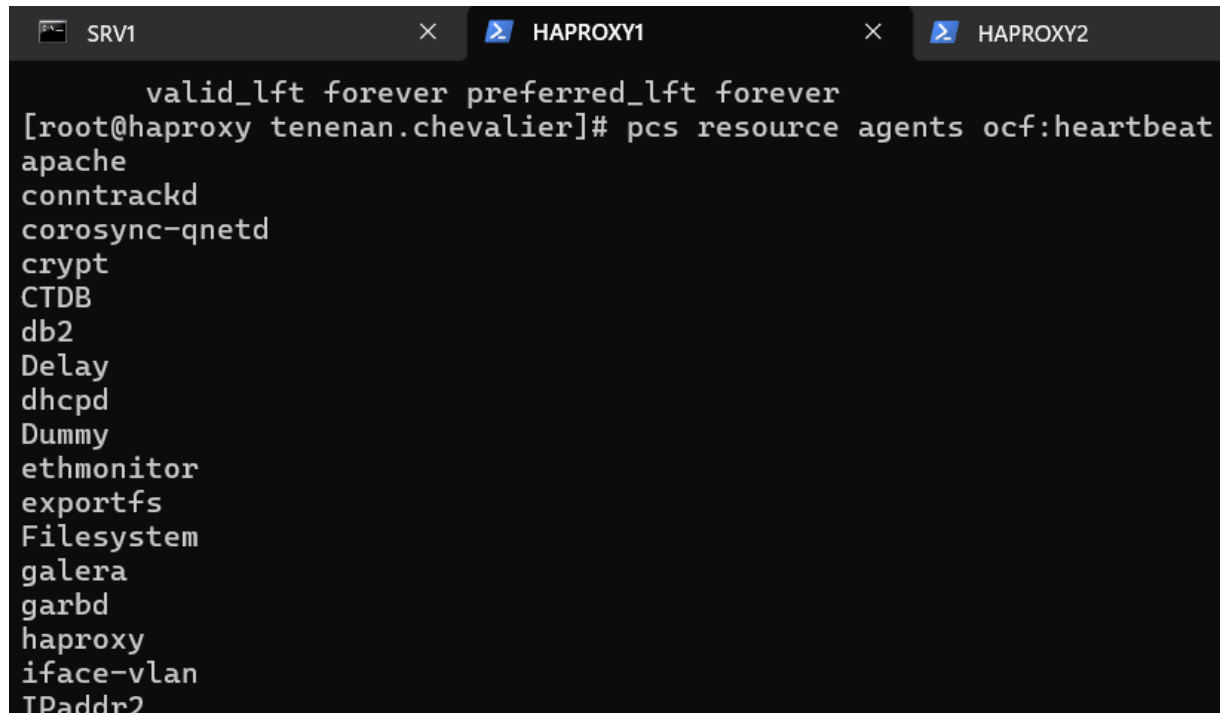
```
[root@haproxy tenenan.chevalier]# pcs status resources
* virtual_ip (ocf:heartbeat:IPaddr2): Started haproxy
```

```
[root@haproxy tenenan.chevalier]# ip a
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: ens160: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group default qlen 1000
    link/ether 00:0c:29:25:30:91 brd ff:ff:ff:ff:ff:ff
    altname enp3s0
    inet 192.168.179.145/24 brd 192.168.179.255 scope global dynamic noprefixroute ens160
        valid_lft 1453sec preferred_lft 1453sec
    inet 192.168.179.150/24 brd 192.168.179.255 scope global secondary ens160
        valid_lft forever preferred_lft forever
    inet6 fe80::20c:29ff:fe25:3091/64 scope link noprefixroute
        valid_lft forever preferred_lft forever
```

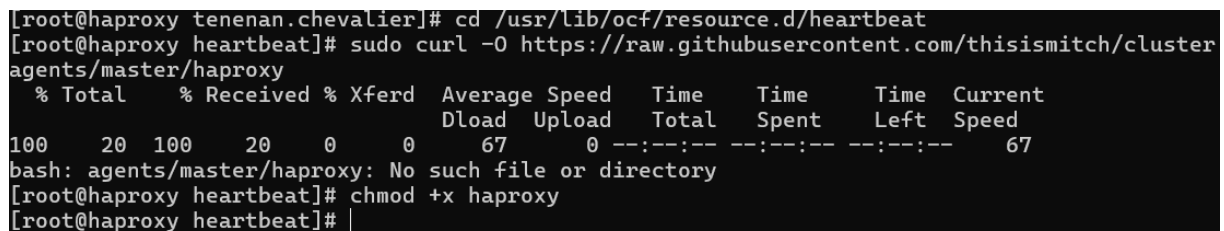
Nous allons ensuite pouvoir récupérer l'agent HAProxy sur nos deux serveurs s'il ne sont pas présents. Nous pourrions vérifier leurs présences avec la commande **pcs resource agents ocf:heartbeat**. S'ils ne sont pas présents, suite les commandes ci-dessous.

#### Liste des commandes

```
cd /usr/lib/ocf/resource.d/heartbeat
sudo curl -O https://raw.githubusercontent.com/thisismitch/cluster-
agents/master/haproxy
chmod +x haproxy
```

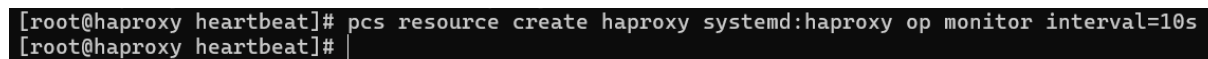


```
valid_lft forever preferred_lft forever
[root@haproxy tenenan.chevalier]# pcs resource agents ocf:heartbeat
apache
contrackd
corosync-qnetd
crypt
CTDB
db2
Delay
dhcpd
Dummy
ethmonitor
exportfs
Filesystem
galera
garbd
haproxy
iface-vlan
IPAddr2
```



```
[root@haproxy tenenan.chevalier]# cd /usr/lib/ocf/resource.d/heartbeat
[root@haproxy heartbeat]# sudo curl -O https://raw.githubusercontent.com/thisismitch/cluster-
agents/master/haproxy
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload  Total   Spent    Left   Speed
100   20  100   20    0     0    67      0  --:--:--  --:--:--  --:--:--   67
bash: agents/master/haproxy: No such file or directory
[root@haproxy heartbeat]# chmod +x haproxy
[root@haproxy heartbeat]# |
```

Une fois les agents importés, nous allons avec la commande **pcs resource create haproxy systemd:haproxy op monitor interval=10s**, créer notre cluster de ressources.

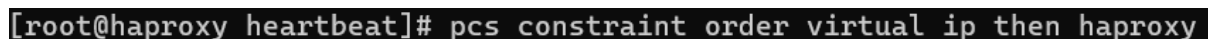


```
[root@haproxy heartbeat]# pcs resource create haproxy systemd:haproxy op monitor interval=10s
[root@haproxy heartbeat]# |
```

Je vais ensuite ajouter mon IP Virtuelle à mon groupe de ressource avec la commande **pcs resource group add HAproxyGroup virtual\_ip haproxy** et j'ajoute une contrainte pour que l'IP virtuelle et le haproxy soit actifs sur le même nœud en même temps avec la commande **pcs constraint order virtual\_ip then haproxy**.



```
[root@haproxy heartbeat]# pcs resource group add HAproxyGroup virtual_ip haproxy
```



```
[root@haproxy heartbeat]# pcs constraint order virtual_ip then haproxy
```

Je peux vérifier le status de mon cluster avec la commande **pcs status**.

```
[root@haproxy heartbeat]# sudo pcs status
Cluster name: mycluster
Cluster Summary:
 * Stack: corosync (Pacemaker is running)
 * Current DC: haproxy (version 2.1.10-1.1.el9_7-5693eae) - partition with quorum
 * Last updated: Tue Apr  7 04:00:44 2026 on haproxy
 * Last change:  Tue Apr  7 04:00:06 2026 by root via root on haproxy
 * 2 nodes configured
 * 3 resource instances configured

Node List:
 * Online: [ haproxy haproxy2 ]

Full List of Resources:
 * haproxy1 (systemd:haproxy):          Started haproxy
 * Resource Group: HAproxy1Group:
 * virtual_ip (ocf:heartbeat:IPaddr2):      Started haproxy
 * haproxy (systemd:haproxy):          Started haproxy

Daemon Status:
 corosync: active/enabled
 pacemaker: active/enabled
 pcsd: active/enabled
```

**haproxy > Online** + fait tourner les ressources (virtual\_ip + haproxy service).

**haproxy2 > Online** + en attente (prêt à prendre le relais si haproxy tombe).

Je mets l'IP virtuelle dans mon fichier host et on voit que le cluster est bien fonctionnel. J'éteint haproxy et haproxy2 prend bien le relais.

```
[root@haproxy tenenan.chevalier]# sudo pcs status
Cluster name: mycluster
Cluster Summary:
 * Stack: corosync (Pacemaker is running)
 * Current DC: haproxy2 (version 2.1.10-1.1.el9_7-5693eae) - partition with quorum
 * Last updated: Tue Apr  7 04:14:48 2026 on haproxy2
 * Last change:  Tue Apr  7 04:00:06 2026 by root via root on haproxy
 * 2 nodes configured
 * 3 resource instances configured

Node List:
 * Online: [ haproxy2 ]
 * OFFLINE: [ haproxy ]

Full List of Resources:
 * haproxy1 (systemd:haproxy):          Started haproxy2
 * Resource Group: HAproxy1Group:
 * virtual_ip (ocf:heartbeat:IPaddr2):      Started haproxy2
 * haproxy (systemd:haproxy):          Started haproxy2

Daemon Status:
 corosync: active/enabled
 pacemaker: active/enabled
 pcsd: active/enabled
```

## Formulaire d'inscription à l'abonnement

Basique	Standard	Prime
5 \$ par mois	10 \$ par mois	20 \$ par mois
Accès complet Documentation Service client Mises à jour gratuites Domaines illimités	Accès complet Documentation Service client Mises à jour gratuites Domaines illimités	Accès complet Documentation Service client Mises à jour gratuites Domaines illimités
<a href="#">S'inscrire</a>	<a href="#">S'inscrire</a>	<a href="#">S'inscrire</a>