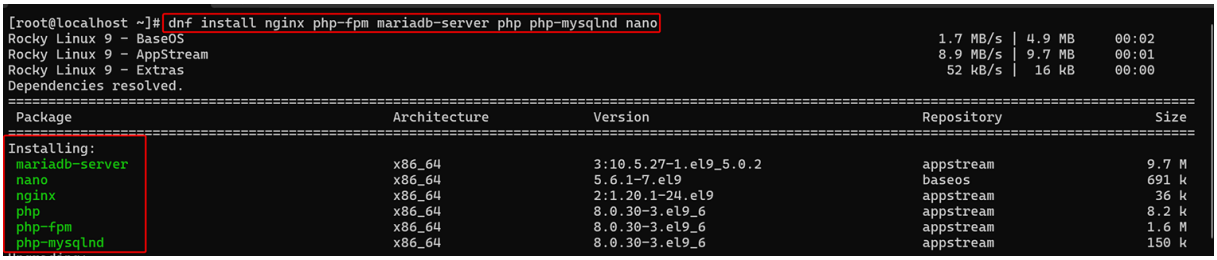
Ténénan Chevalier

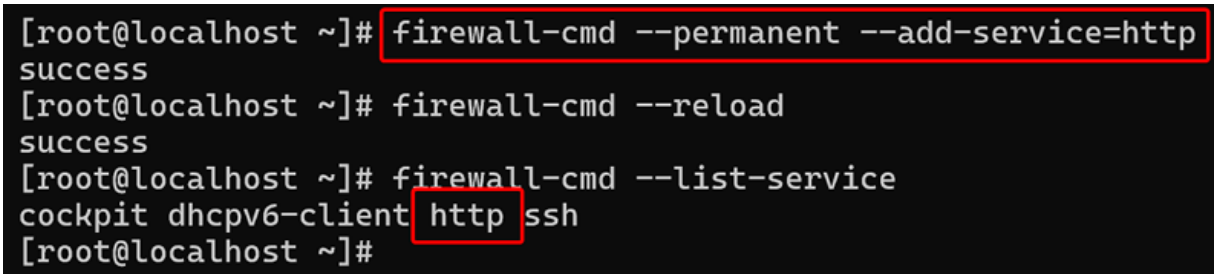
GUIDE MISE EN PLACE SERVICE WEB/HAPROXY

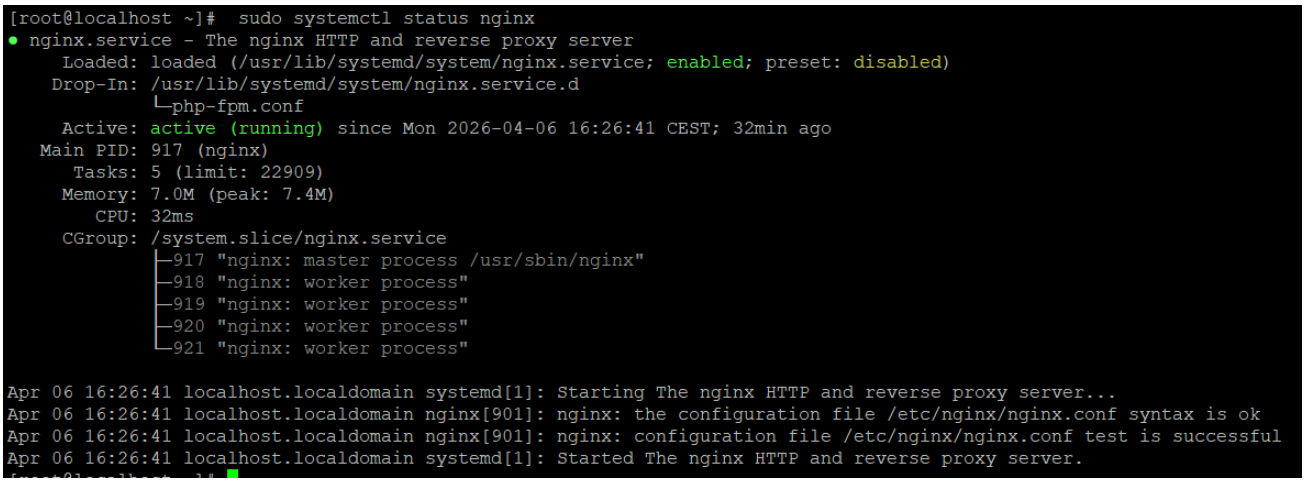
Sous rocky linux 9.6

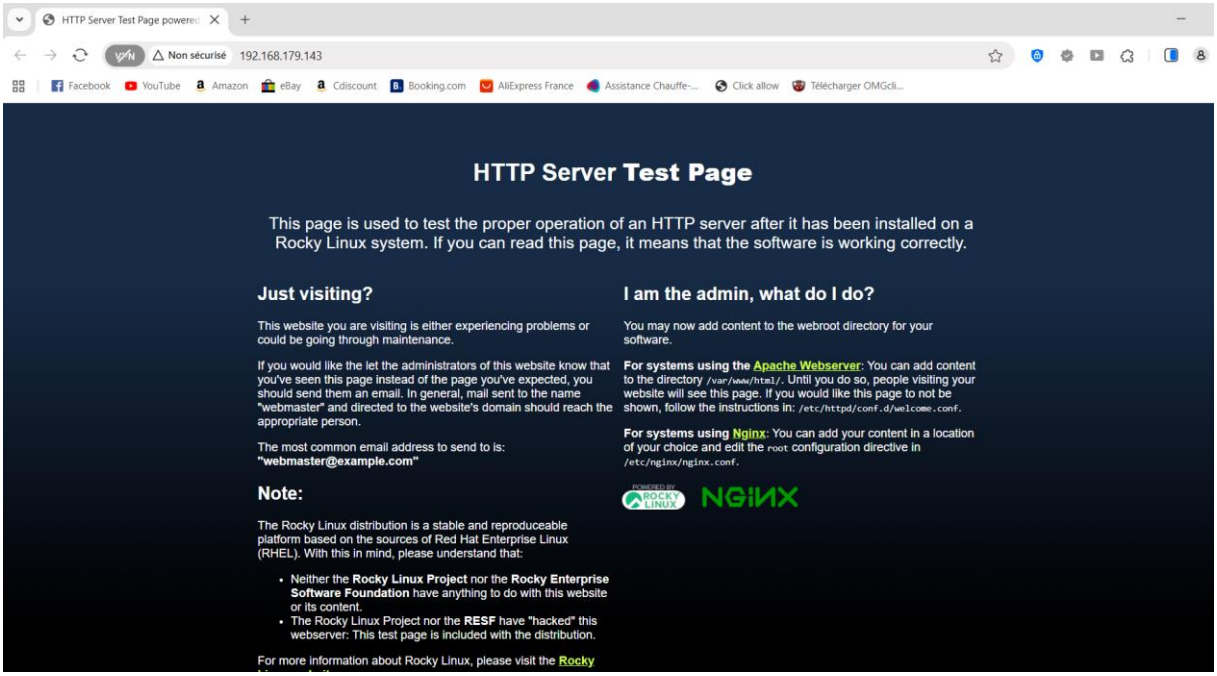
Installation de Nginx & MariaDB

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-mysqlnd nano.**









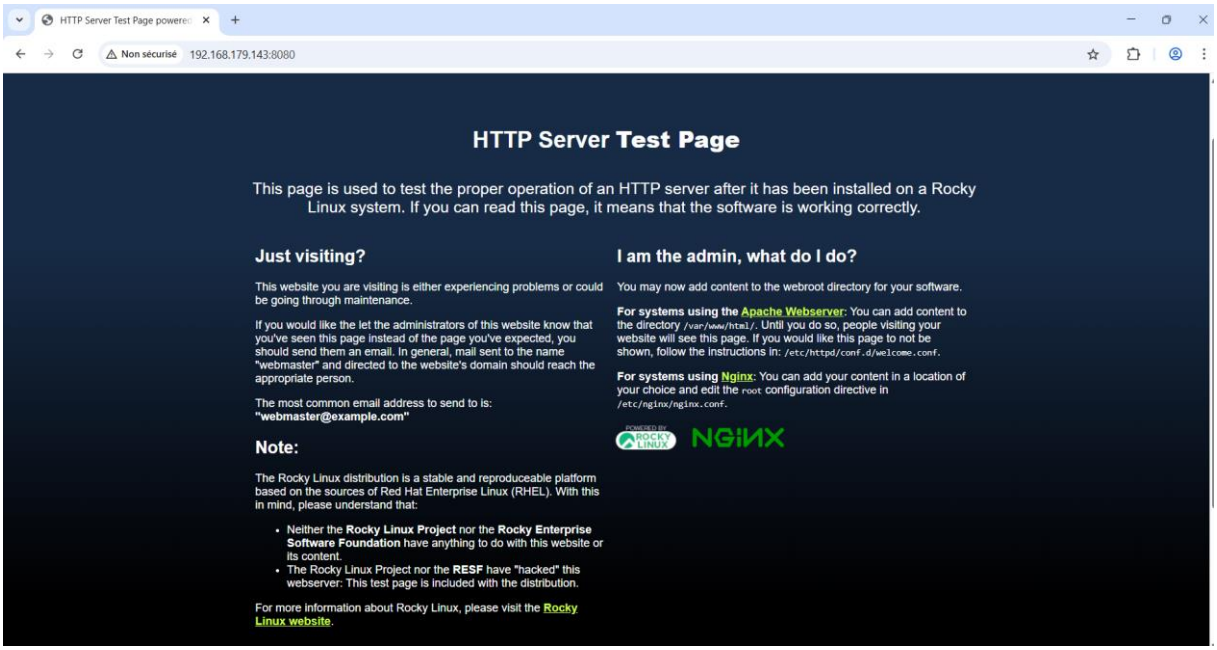
Changement du port d'écoute

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/ngx_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 pourrons 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 'xxxxxxxx';

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

FLUSH PRIVILEGES;

SHOW DATABASES;

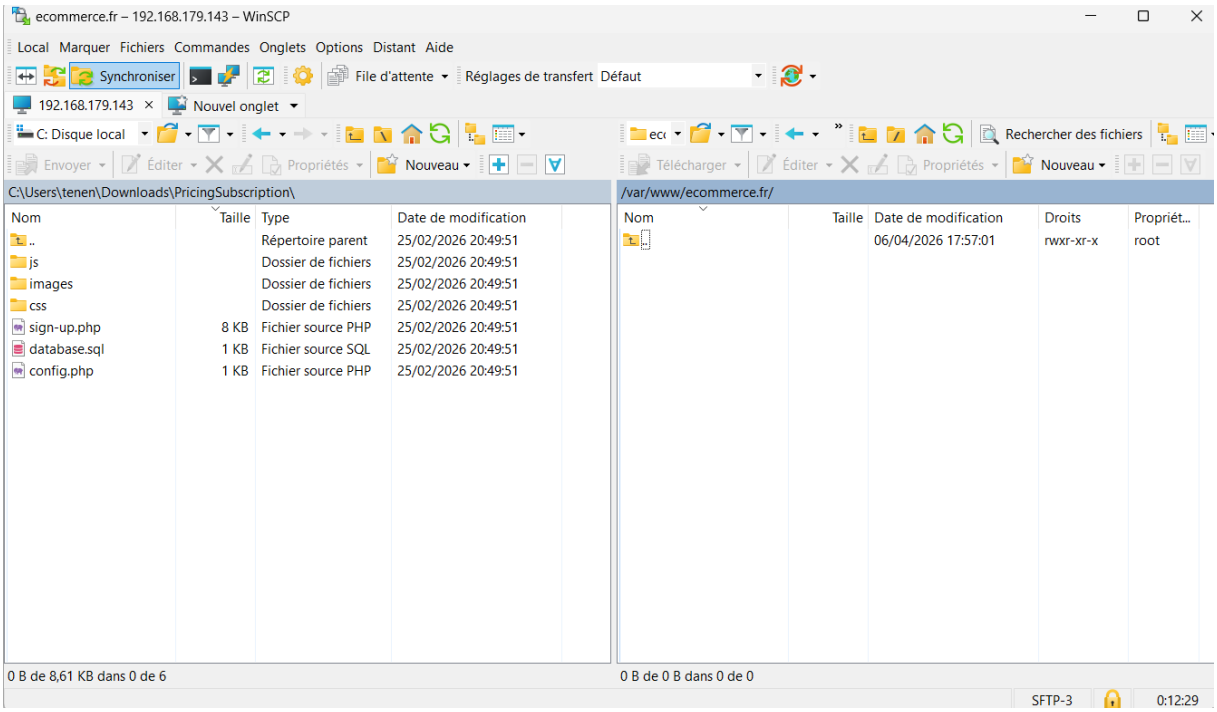
SHOW TABLES;

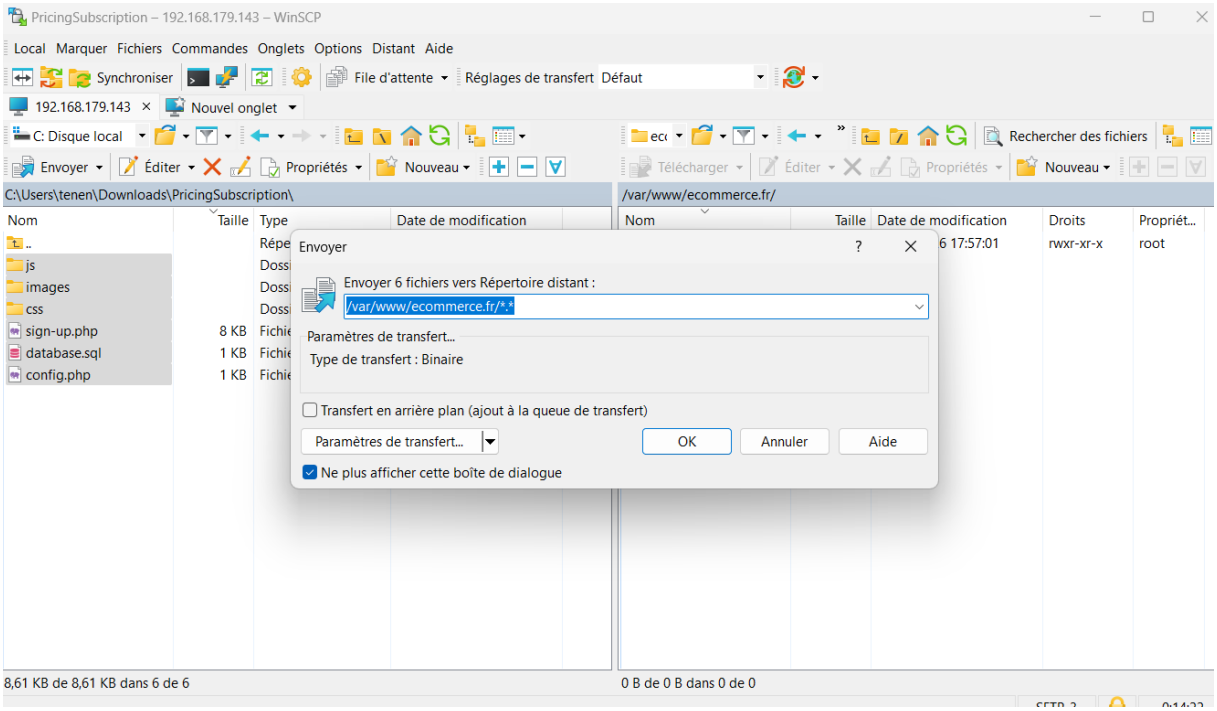
```
[root@localhost ~]#   sudo mysql -u root -e "SHOW DATABASES;"
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| site_fr            |
+--------------------+
```

```
[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 créer, 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 apparait lors de l'accès au site. Nous mentionnerons également les chemins des fichiers de logs et d'autres paramètres PHP. On créer 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";
?>
```

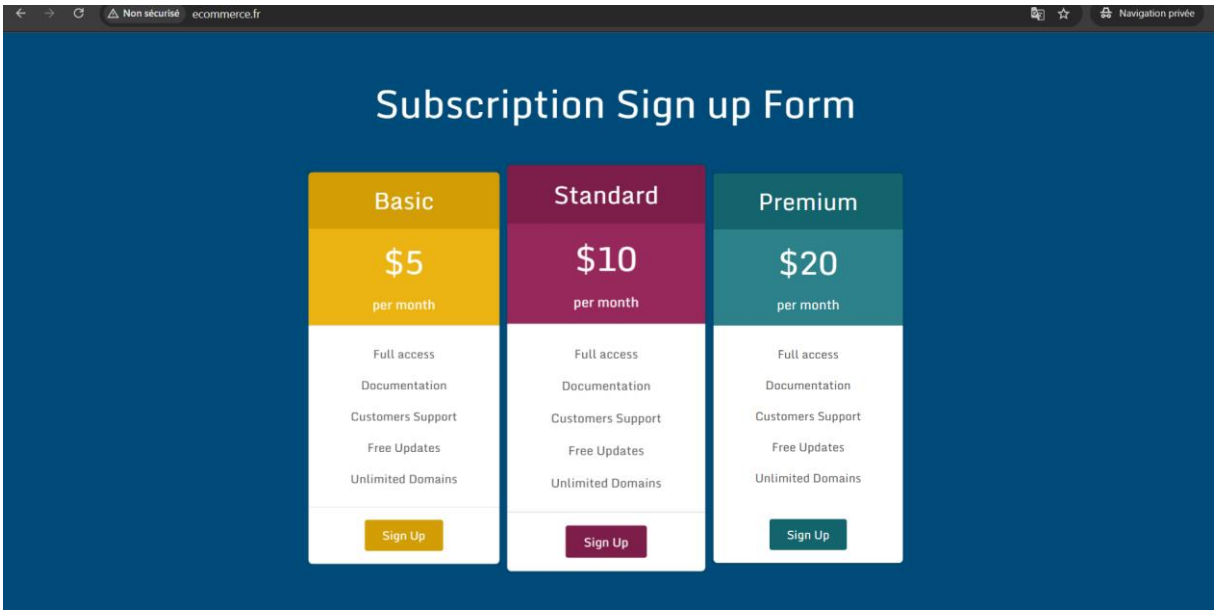
Nous retournons ensuite sur MariaDB avec la commande **mysql -u root** puis nous entrons les requêtes suivantes pour créer les tables nécessaires dans la base de données**.**

```
[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.

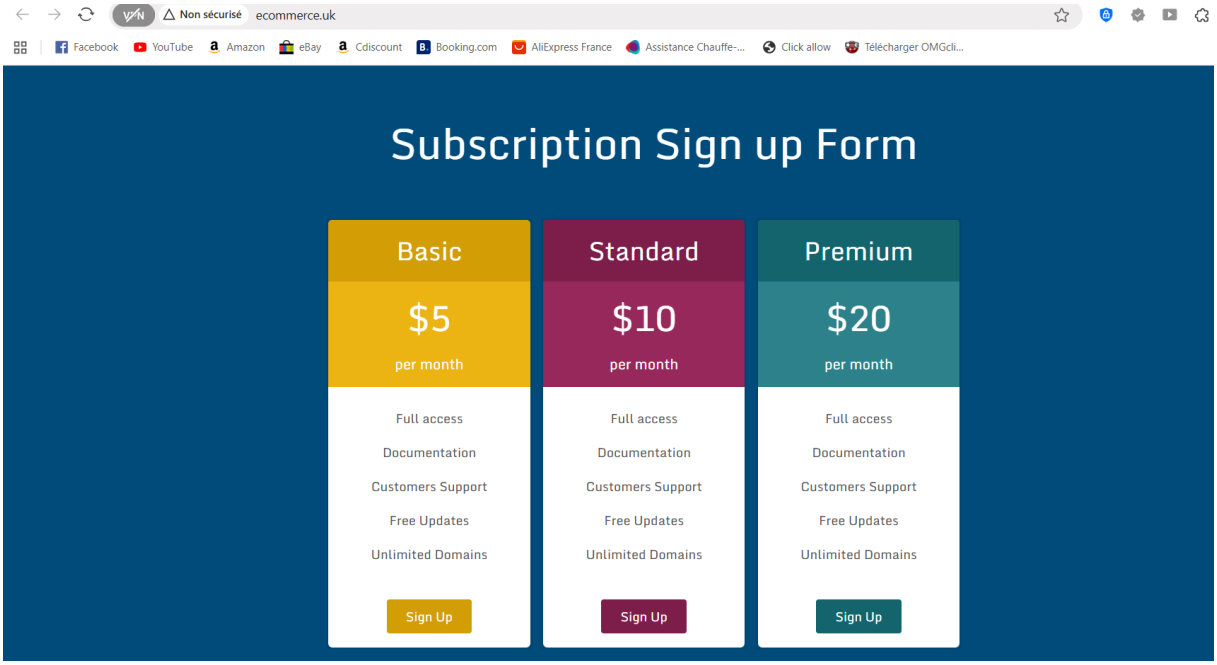
```
# Copyright (c) 1993-2009 Microsoft Corp.
  #
  # This is a sample HOSTS file used by Microsoft TCP/IP
   for Windows.
  #
  127.0.0.1       localhost
  ::1             localhost
  192.168.179.143   ecommerce.fr
```




```
[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 différente, nous changement 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;
 }
}
```

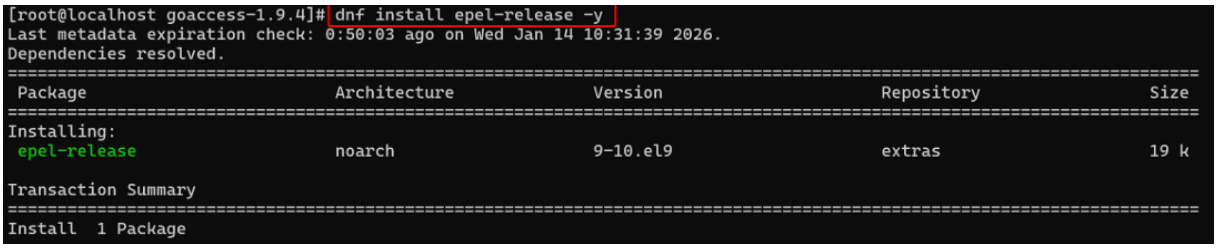
Une fois le service redémarrer avec la commande **systemctl restart nginx**, je constate que je n’ai plus accès au site.

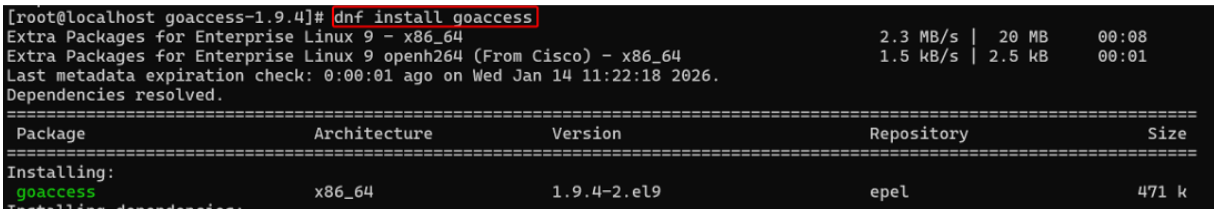
LES LOGS :

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

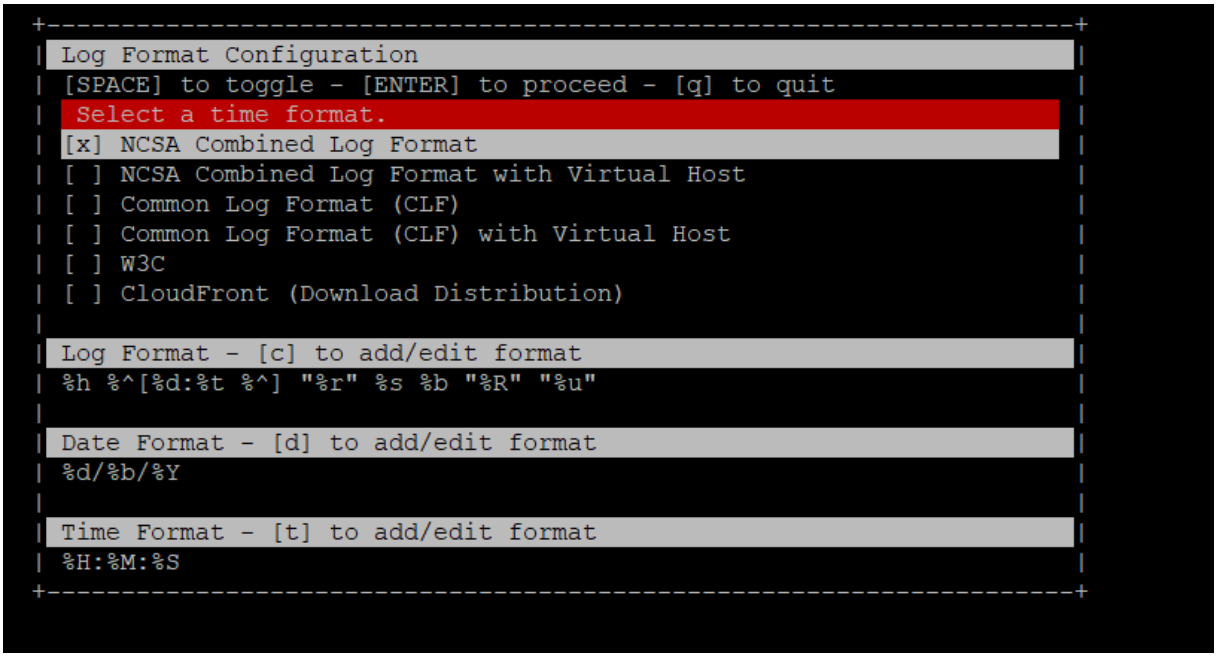
```
[root@localhost ~]# cd /var/log/nginx
[root@localhost nginx]# ls
access.log                access_ecommerce.uk.log  error_ecommerce.fr.log
access_ecommerce.fr.log   error.log                error_ecommerce.uk.log
```

Je peux également les lires plus facilement en installant goaccess avec la commande **dnf install epel release -y** puis la commande **dnf install goaccess.**



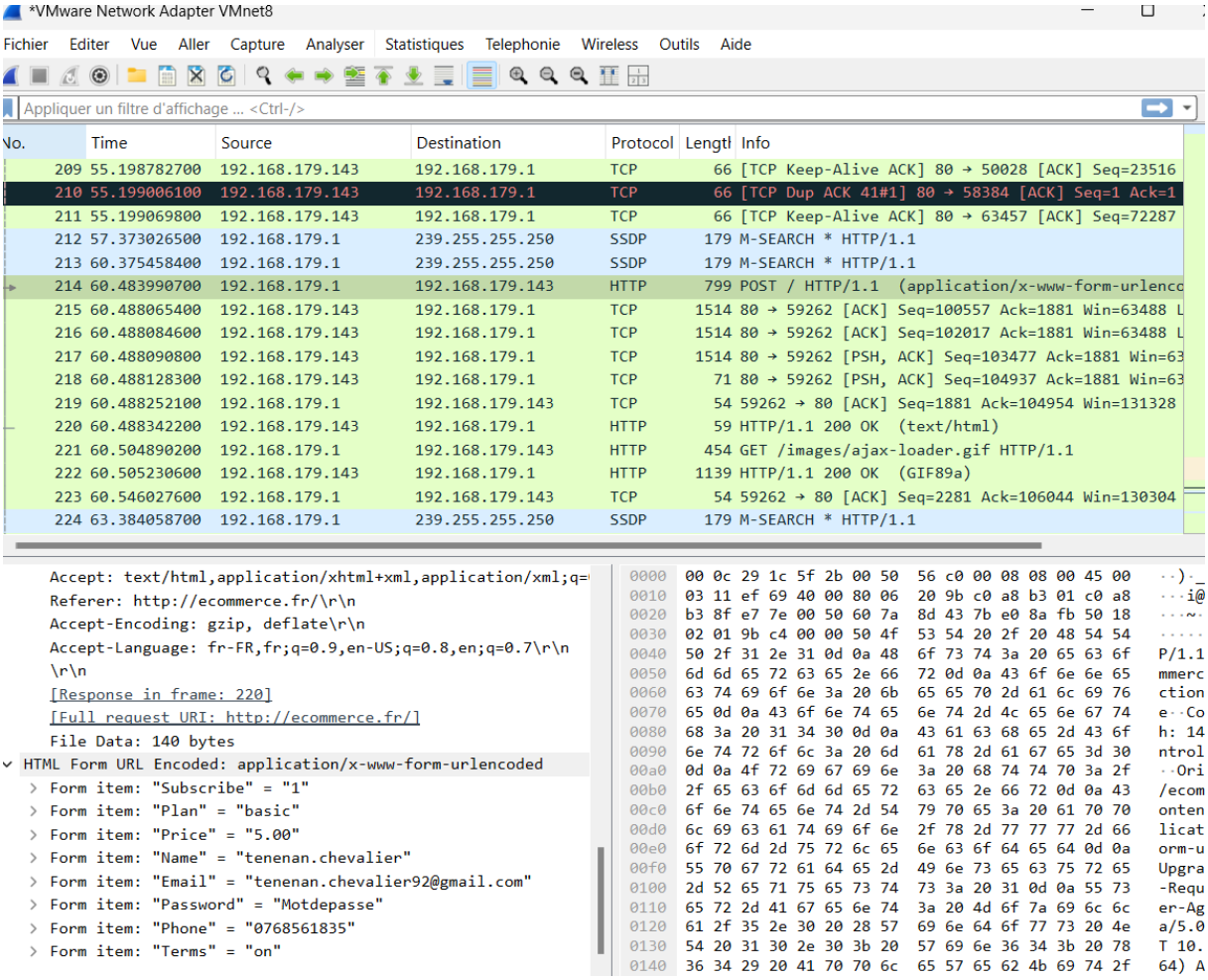


On utilisera la commande **goaccess -f /var/log/nginx/error.log** pour lire les logs.



Test de sécurité

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



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 :

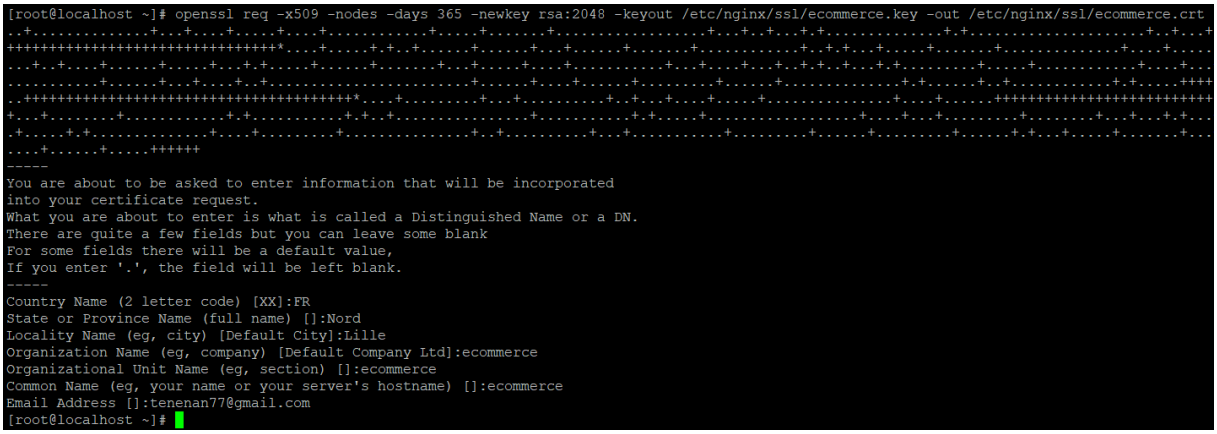
mkdir /etc/ssl/private & chmod 700 /etc/ssl/private.

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

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

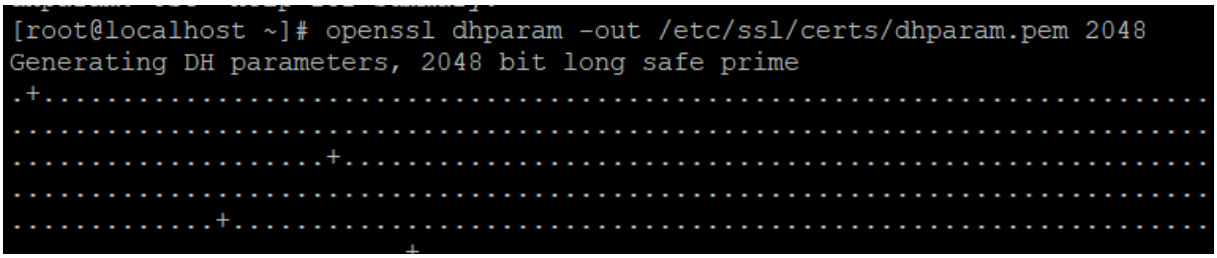
openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout /etc/nginx/ssl/ecommerce.key -out /etc/nginx/ssl/ecommerce.crt

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



Nous allons maintenant générer les paramètres DH avec la commande

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



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;
    }
}
```

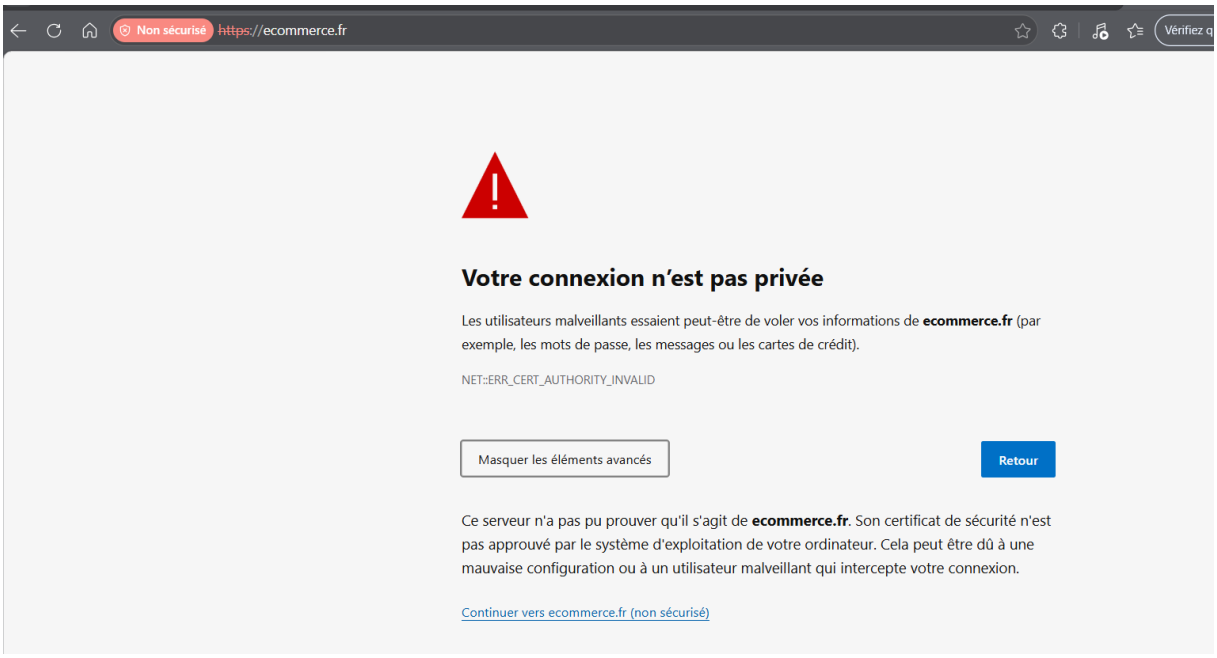
```
server {
    listen 443 ssl;
    listen [::]:443 ssl;
    ssl_certificate /etc/ssl/certs/nginx-selfsigned.crt;
    ssl_certificate_key /etc/ssl/private/nginx-selfsigned.key;
    ssl_dhparam /etc/ssl/certs/dhparam.pem;

    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;
    }
}
```

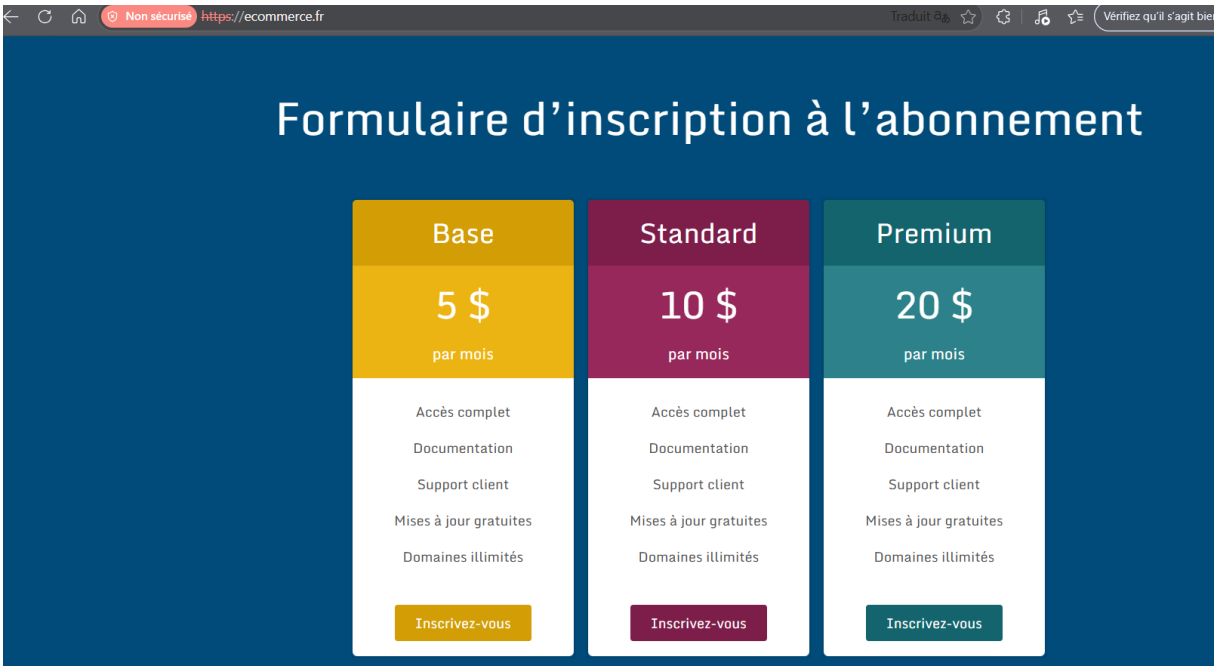
La commande **nginx -t** va permettre de vérifier que la conf est ok, suite à quoi il faudra redémarrer nginx avec la commande **systemctl restart nginx.**

```
[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.

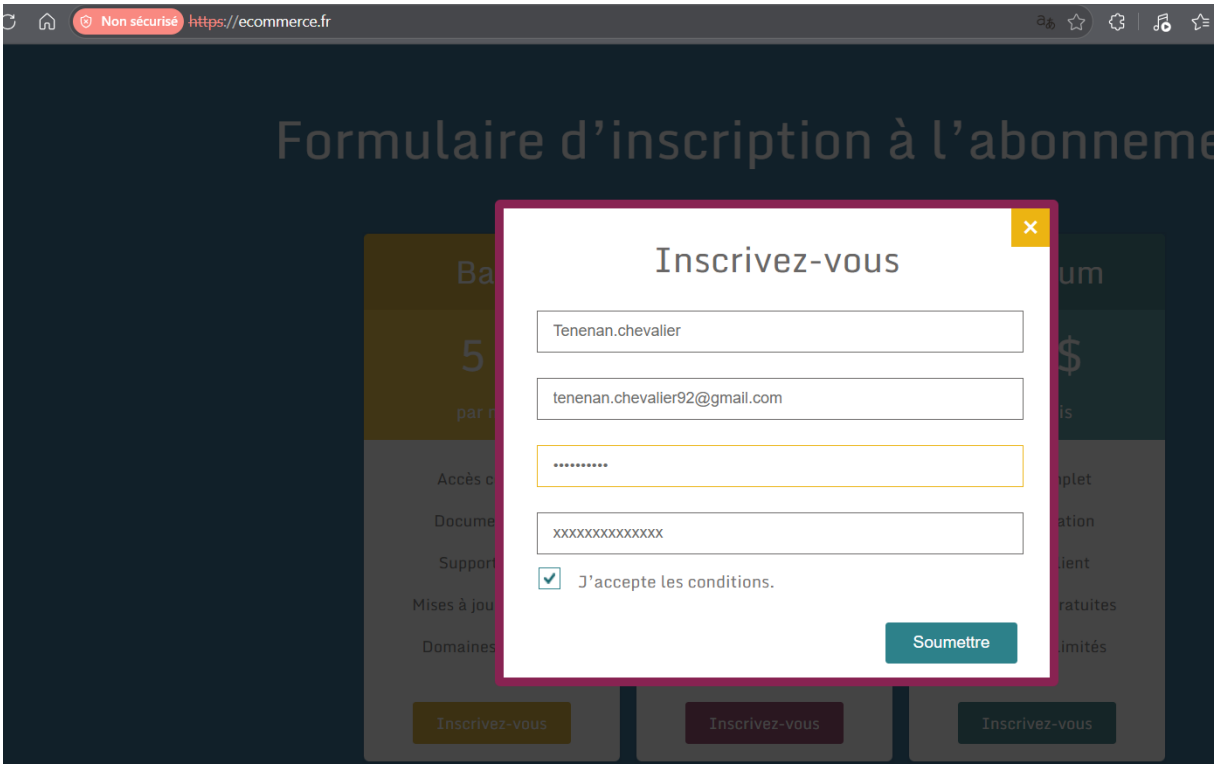
```
[root@localhost conf.d]# firewall-cmd --permanent --add-service=https
success
[root@localhost conf.d]# firewall-cmd --reload
success
[root@localhost conf.d]# firewall-cmd --list-service
cockpit dhcpv6-client http https ssh
[root@localhost conf.d]#
```

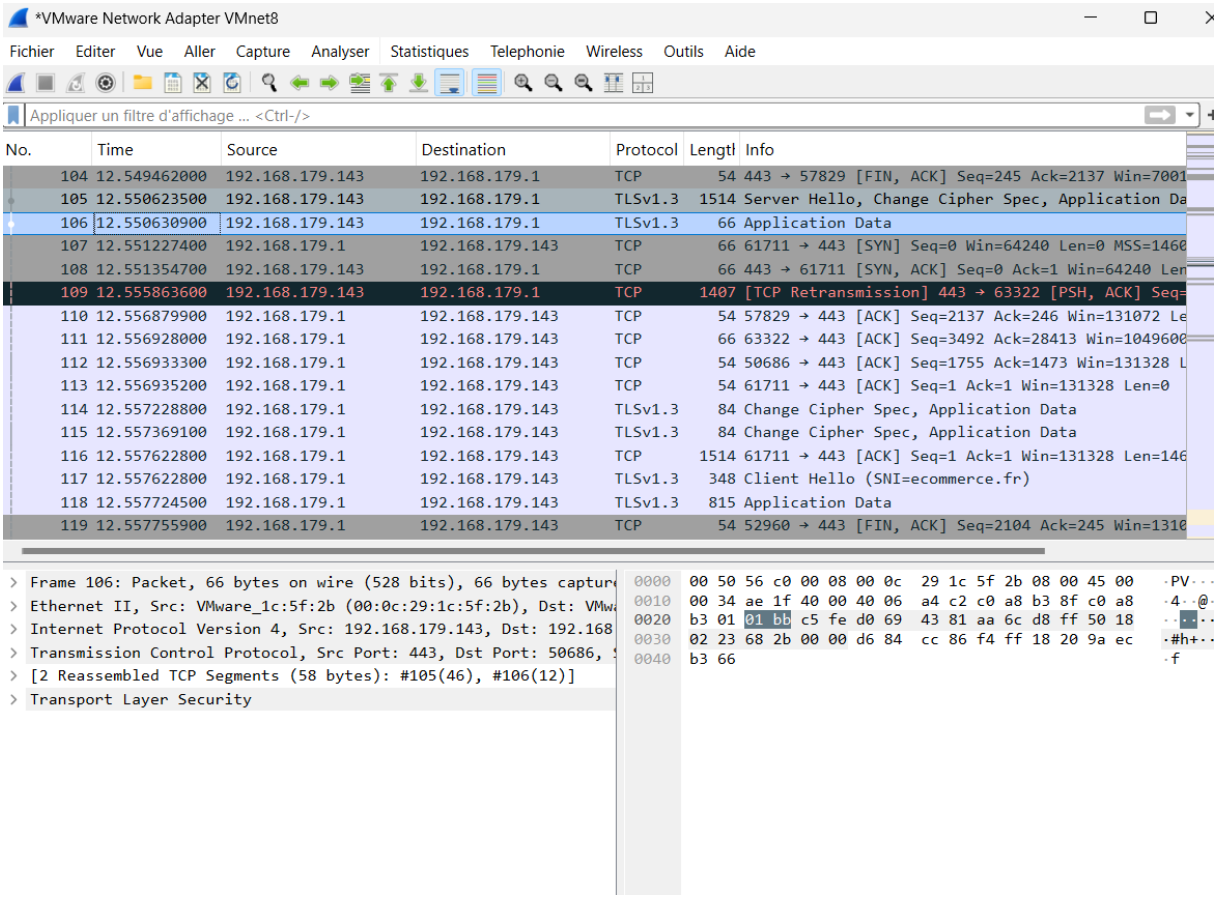

Non sécurisé
https://ecommerce.fr
Votre connexion n'est pas privée
Les utilisateurs malveillants essaient peut-être de voler vos informations de ecommerce.fr (par exemple, les mots de passe, les messages ou les cartes de crédit).
NET::ERR_CERT_AUTHORITY_INVALID
Masquer les éléments avancés
Retour
Ce 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
Formulaire d'inscription à l'abonnement
Base
5 $
par mois
Accès complet
Documentation
Support client
Mises à jour gratuites
Domaines illimités
Inscrivez-vous
Standard
10 $
par mois
Accès complet
Documentation
Support client
Mises à jour gratuites
Domaines illimités
Inscrivez-vous
Premium
20 $
par mois
Accès complet
Documentation
Support client
Mises à jour gratuites
Domaines illimités
Inscrivez-vous

TEST WIRESHARK

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





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_uti/;**

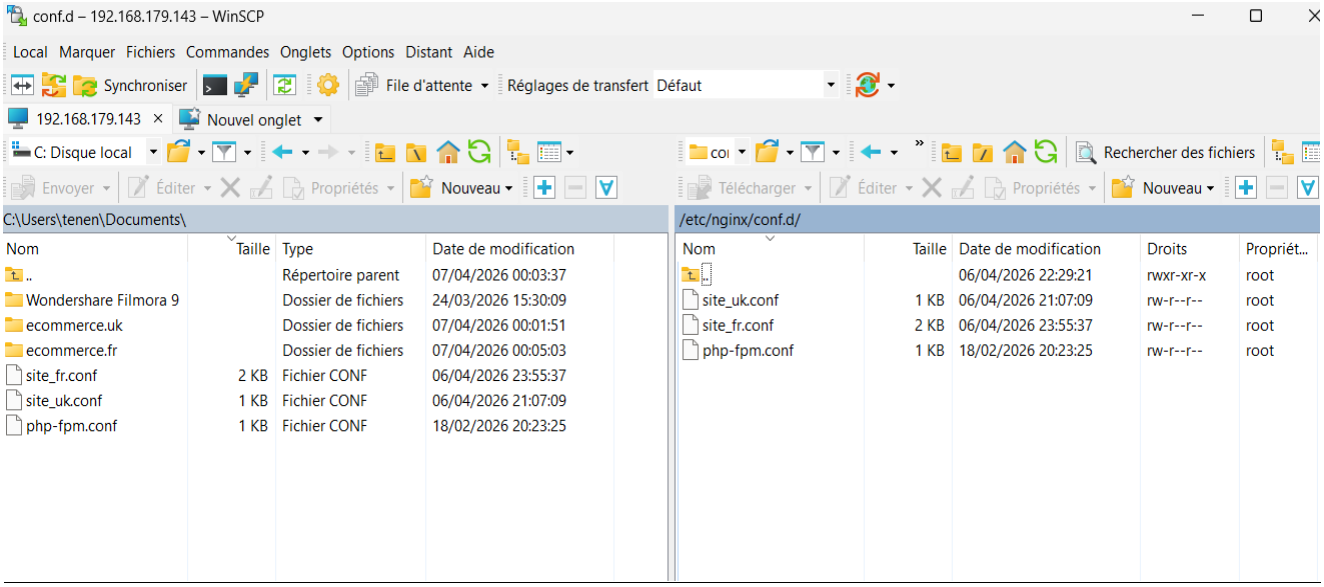
```
server {
    return 301 https://$host$request_uti/;
    listen 80;
    listen [::]:80;
    server_name ecommerce.fr;
    return 301 https://$host$request_uri;
}

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
```

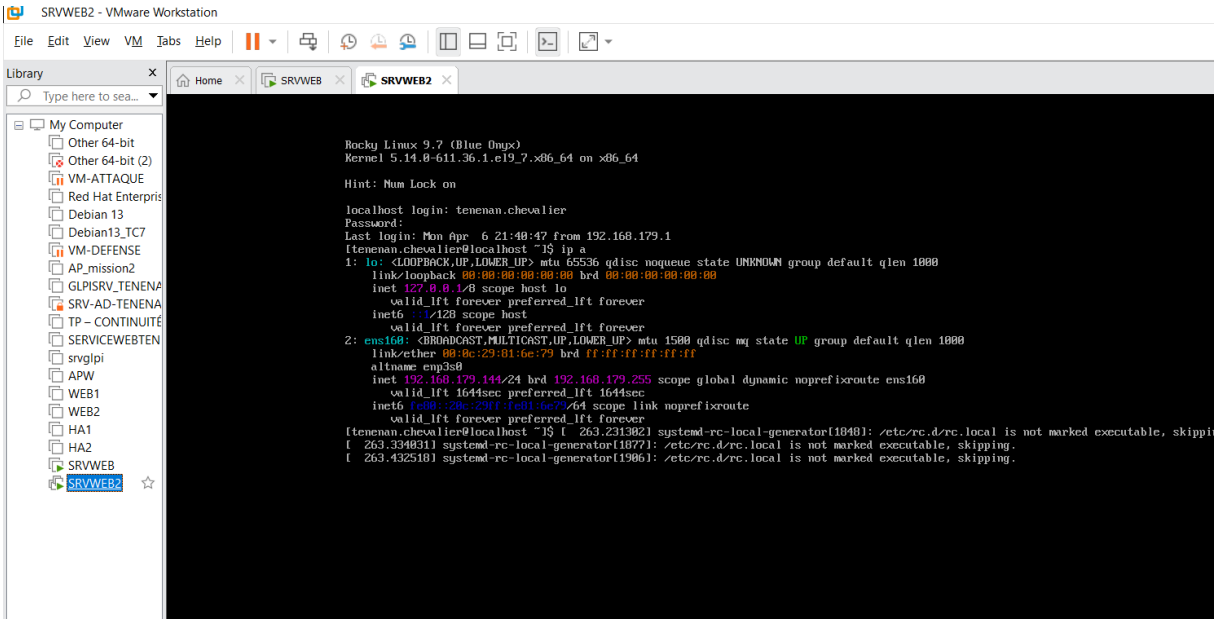
Mise en place d'un deuxième serveur web

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: di>
    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 s>
Apr 07 00:15:39 srvweb2 nginx[1825]: nginx: the configuration file /etc/nginx/n>
Apr 07 00:15:39 srvweb2 nginx[1825]: nginx: configuration file /etc/nginx/nginx>
Apr 07 00:15:39 srvweb2 systemd[1]: Started The nginx HTTP and reverse proxy se>
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.**

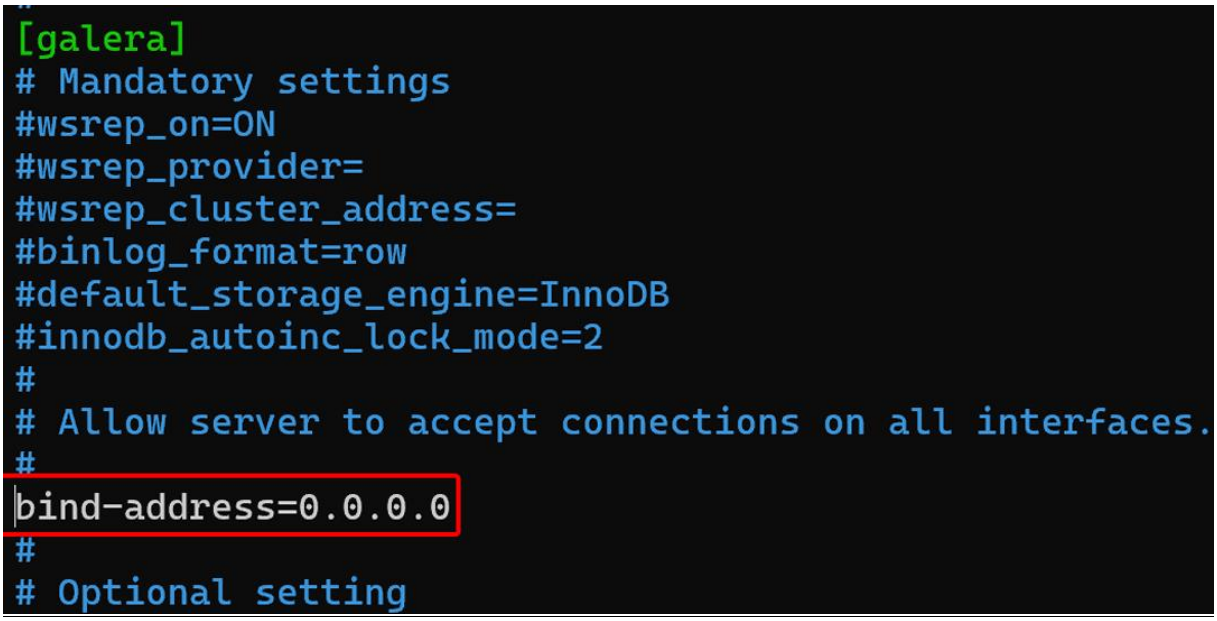
```
[tenenan.chevalier@localhost ~]$  cat /var/www/ecommerce.fr/config.php
<?php
$SETTINGS["mysql_user"]='tenenan';
$SETTINGS["mysql_pass"]='Lmrugby2015';
$SETTINGS["hostname"]='192.168.179.143';
$SETTINGS["mysql_database"]='site_fr';
$SETTINGS["data_table"]='registrations';
$SETTINGS["paypal_address"]='email@domain.com';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

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

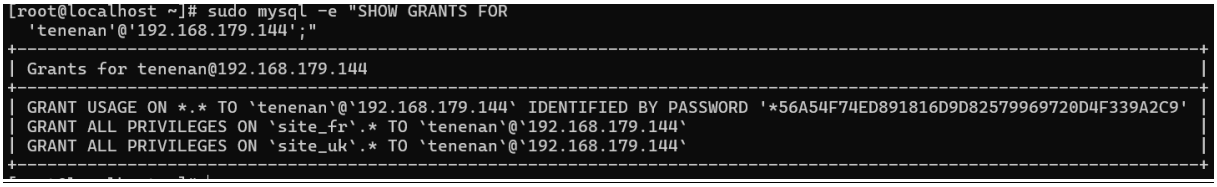
firewall-cmd --permanent --add-port=3306/tcp et recharger le firewall avec la commande **firewall-cmd –reload.**

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

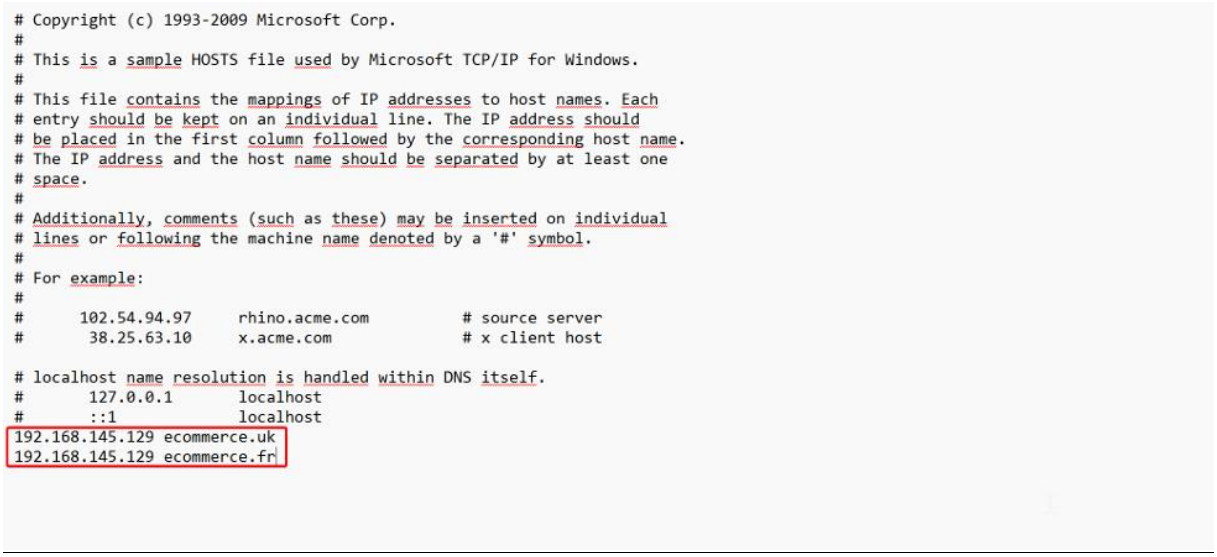
Modifier le fichier mariadb avec la commande **nano /etc/my.cnf.d/mariadb-server.cnf** pour autoriser la communication extérieure.

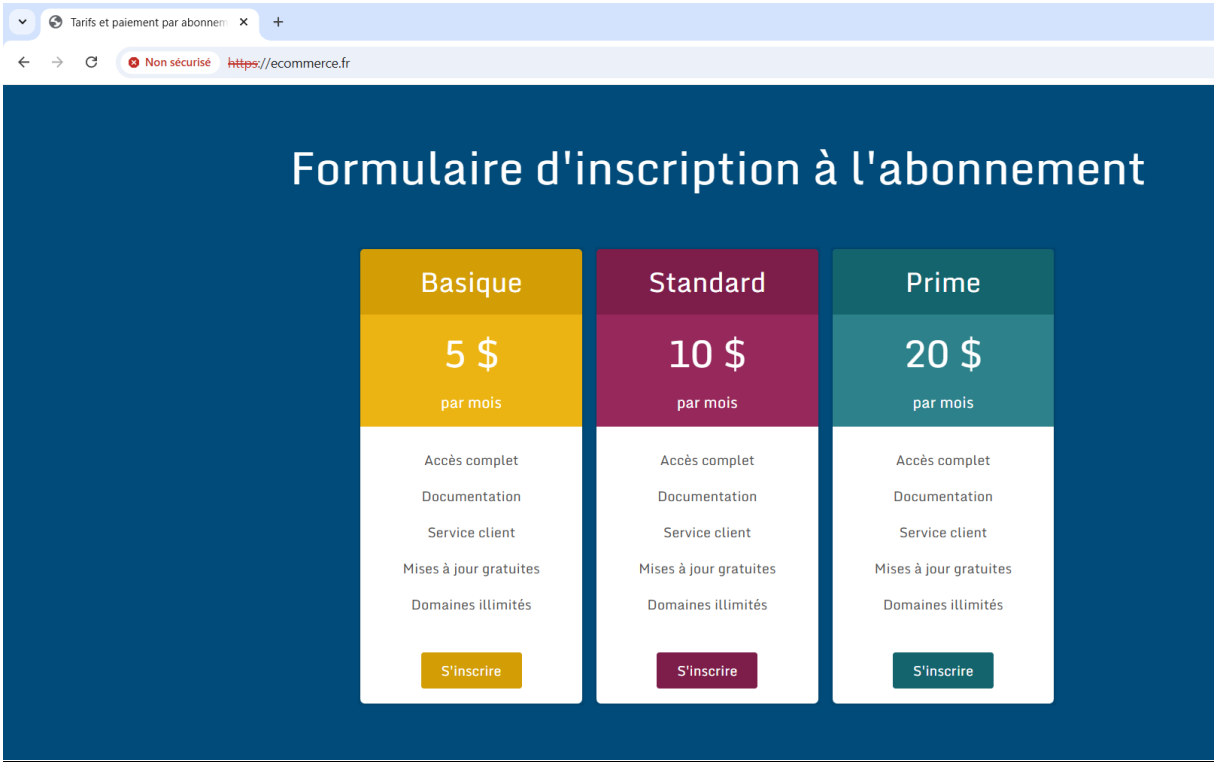


Il faudra aussi créer un autre utilisateur sur la base de données qui aura accès à notre serveur web 2, et lui donner les droits sur les différents sites.



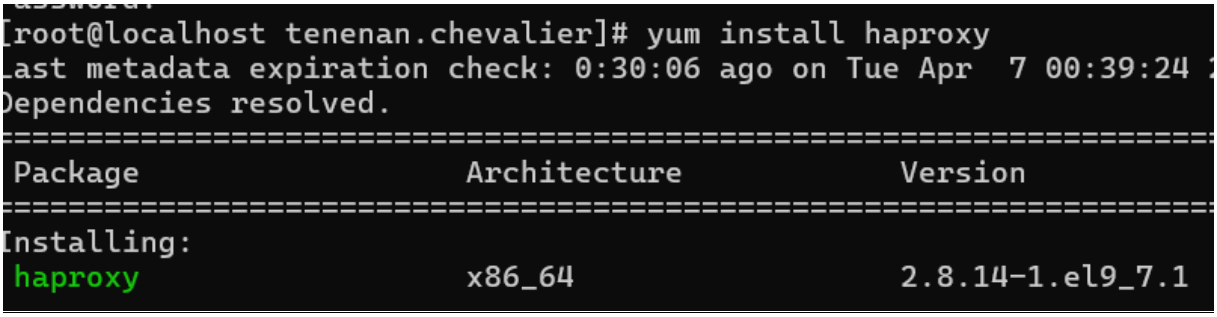
Il faudra remettre en place le HTTPS (voir table des matières). En modifiant l'IP présente dans mon fichier hosts, je remarque les sites sont bien fonctionnels.





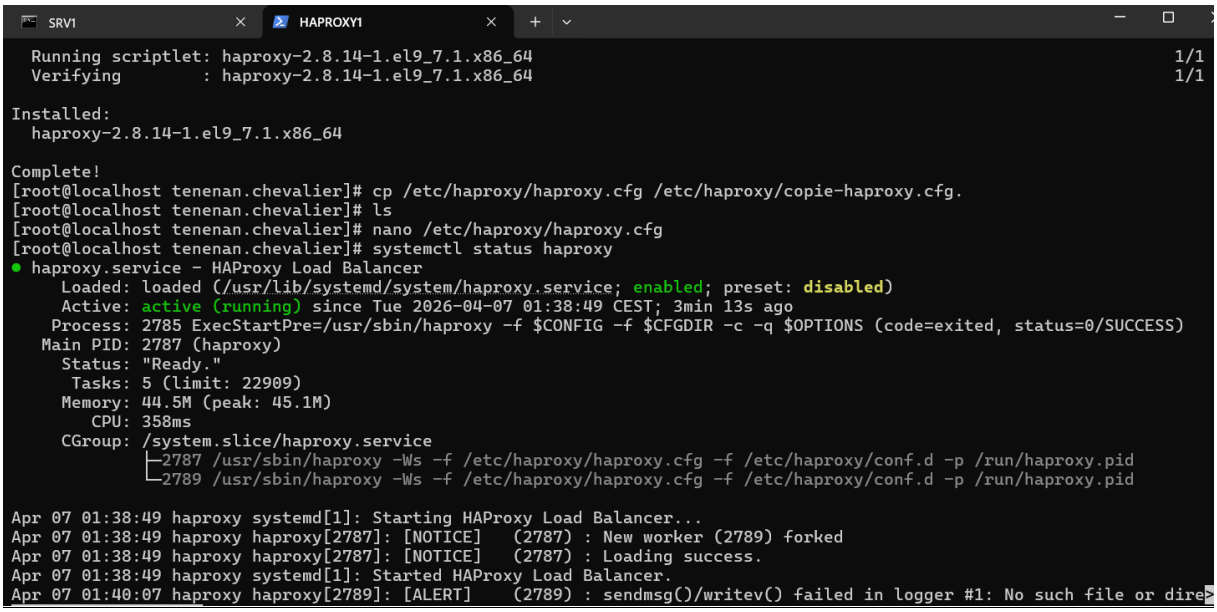
Mise en place du serveur HAProxy

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



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.




```
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 pourrons également ajouter la configuration pour l'accès à l'interface d'administration en ajoutant le texte ci-dessous dans la partir « 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 nummber 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/imtcp.html
#module(load="imtcp") # needs to be done just once
```

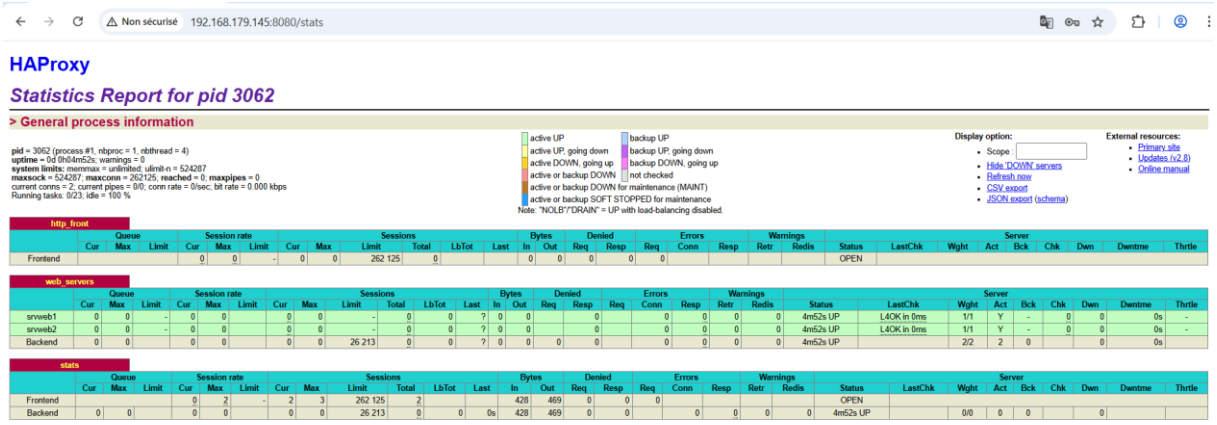
On va ensuite pouvoir démarrer les services haproxy et rsyslog avec la commande **systemctl restart haproxy && systemctl restart rsyslog**.

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

Pour accéder à l'interface graphique et vérifier que tous 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
```

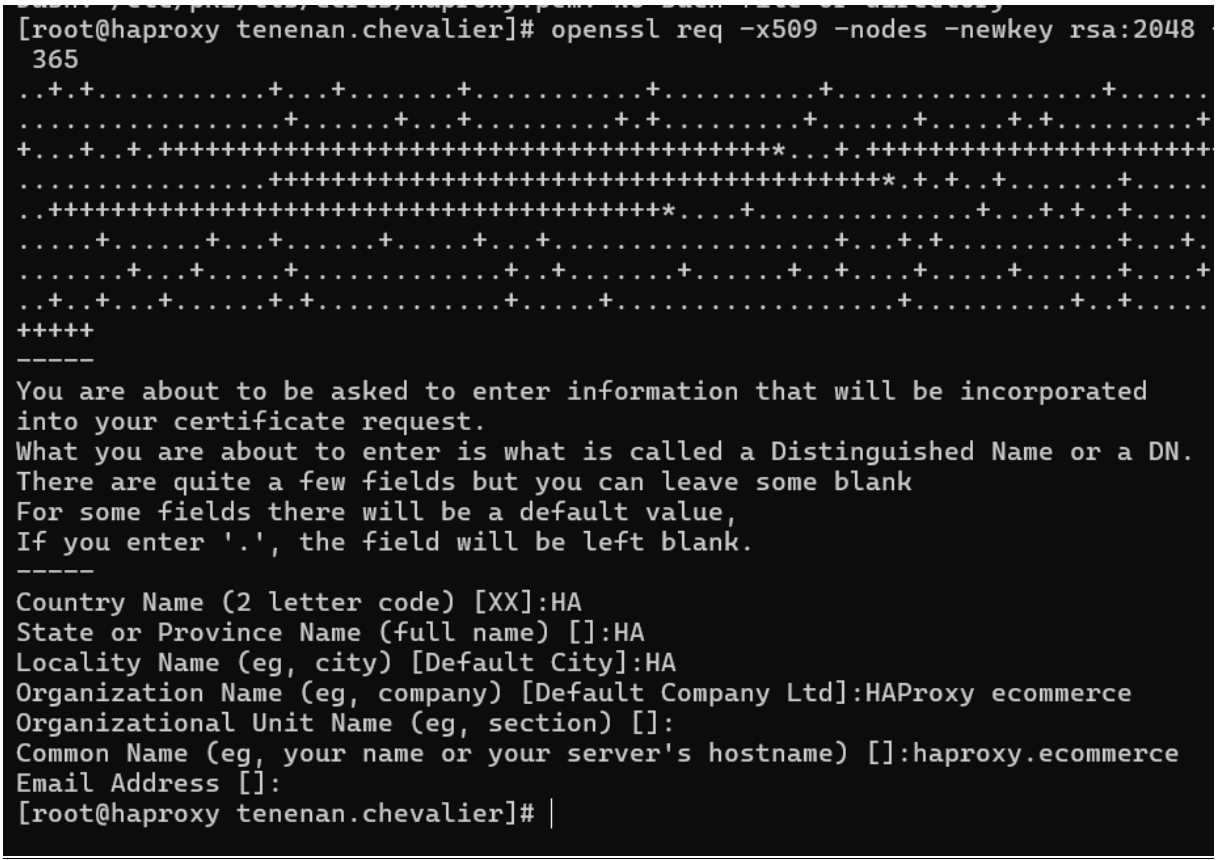
Si je vais sur le port 8080, je vois bien les stats de mon HAProxy.



Securisation du HAProxy

Je vais d'abord générer un certificat pour mon HAProxy avec la commande suivante **:**

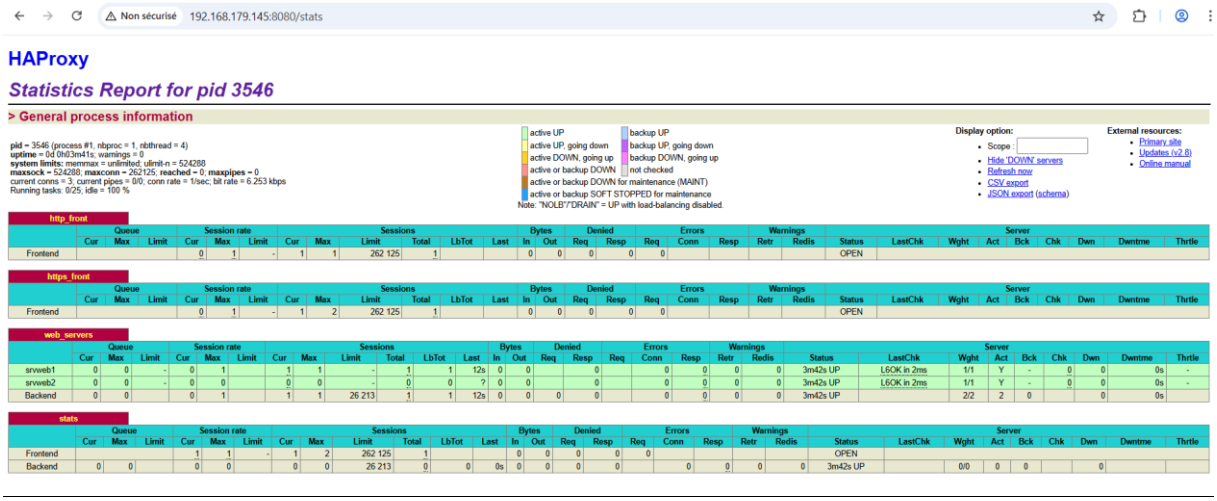
openssl req -x509 -nodes -newkey rsa:2048 -keyout /etc/pki/tls/certs/haproxy.pem -out /etc/pki/tls/certs/haproxy.pem -days 365



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.



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 highavailabilit
[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'
haproxy: 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_7-5693eaeee) - 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 propert
Cluster Properties: cib-bootstrap-options
  cluster-infrastructure=corosync
  cluster-name=mycluster
  dc-version=2.1.10-1.1.el9_7-5693eaeee
  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=30ssudo p
cs resource create virtual_ip
```

On peut vérifier que l'IP a bien été créer avec la commande **pcs status resources** et on peut la voir avec la commande ip a. On voir 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 pourrons 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
```

SRV1 | HAPROXY1 | HAPROXY2

```
       valid_lft forever preferred_lft forever
[root@haproxy tenenan.chevalier]# pcs resource agents ocf:heartbeat
apache
conntrackd
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
```

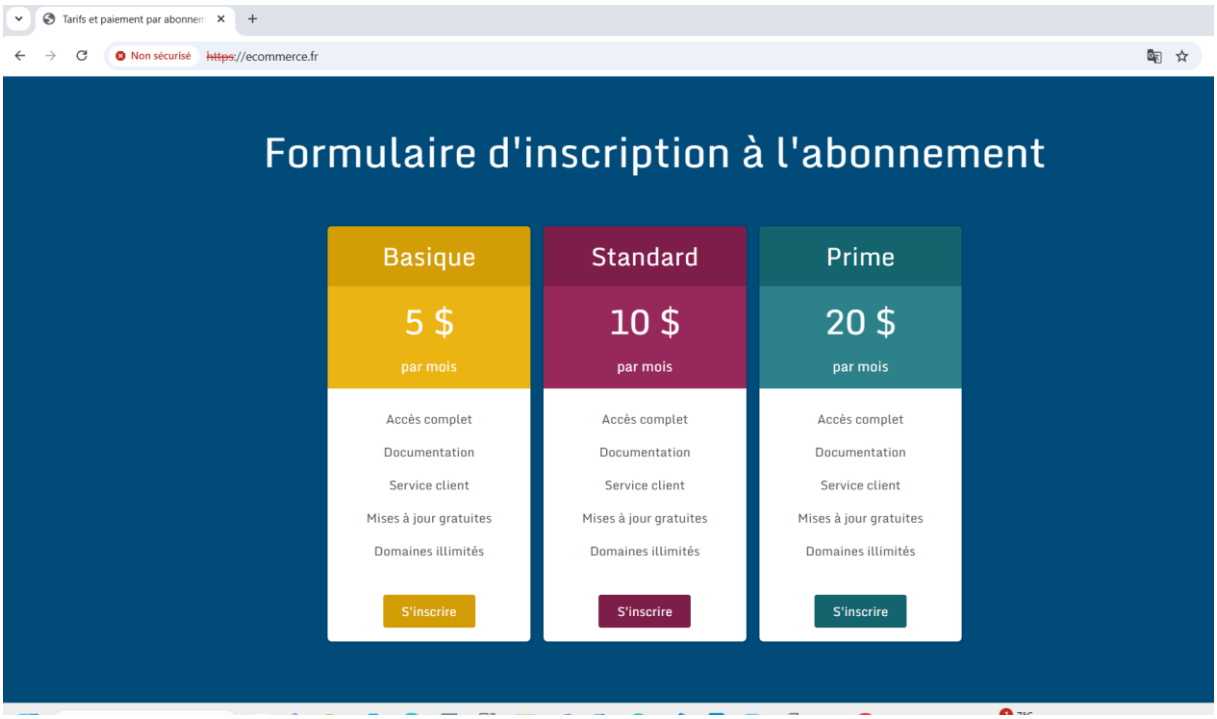
Je peux vérifier le status de mon cluster avec la commande **pcs status**.

```
[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_7-5693eaeee) - 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 relai.

```
[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_7-5693eaeee) - 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
```


Tarifs et paiement par abonnem
Non sécurisé
https://ecommerce.fr
Formulaire d'inscription à l'abonnement
Basique
5 $
par mois
Accès complet
Documentation
Service client
Mises à jour gratuites
Domaines illimités
S'inscrire
Standard
10 $
par mois
Accès complet
Documentation
Service client
Mises à jour gratuites
Domaines illimités
S'inscrire
Prime
20 $
par mois
Accès complet
Documentation
Service client
Mises à jour gratuites
Domaines illimités
S'inscrire