

TP Proxmox - part2



BUT2 – IUT Limoges 2025

R4.A.08 : Virtualisation

Documentation officielle : <https://pve.proxmox.com/pve-docs/pve-admin-guide.html>

Version PDF : <https://pve.proxmox.com/pve-docs/pve-admin-guide.pdf>

Proxmox Virtual Environnement est une solution de virtualisation libre (licence AGPLv3) basée sur l'hyperviseur Linux KVM, et offre aussi une solution de containers avec LXC. Elle est fournie avec un packaging par Proxmox Server Solutions GmbH et dispose d'un support payant.

Proxmox est une solution de virtualisation de type "bare metal" (Hyperviseur de type 1).

Le packaging de Proxmox VE est fourni sur une image iso. L'installateur (basé sur Debian) configure tout ceci :

- Système d'exploitation complet (Debian Stable 64 bits)
- Partitionnement de disque dur avec LVM2
- Support de LXC (containers) et du module KVM (virtualisation complète)
- Outils de sauvegarde et de restauration
- Interface web d'administration et de supervision.
- Fonctions de clustering qui permet par exemple la migration à chaud des machines virtuelles d'un serveur physique à un autre (à condition d'utiliser un stockage partagé, SAN, ou Ceph sinon la migration entraîne une courte interruption lors du redémarrage sur un autre nœud du cluster).

Proxmox propose donc 2 types de virtualisation :

- **virtualisation matérielle** (ou complète) : KVM : permet la virtualisation de tout système d'exploitation sur des processeurs d'architectures x86_64 disposant des technologies Intel VT ou AMD-V.
- **virtualisation par container** : LXC : permet la création d'instances de système d'exploitation isolées, Linux uniquement. Cette solution est plus performante (consomme moins de ressources) qu'une virtualisation matérielle du fait du peu d'overhead.

Page wikipedia de Proxmox

Table des matières

.....	.1
A. Résumé de l'épisode précédent.....	.3
1. Travaux réalisés.....	.3
2. Problèmes rencontrés.....	.3
3. Réponse(s).....	.3
Création d'un réseau NAT dans VirtualBox.....	.4
Redirection de ports.....	.6
B. Objectif de la partie 2 du TP.....	.8
C. Configuration de la VM (ou conteneur) Nginx.....	.9
1. Configuration du réseau de vm-nginx.....	.9
2. Ajout et configuration 2ème bridge.....	.10
3. Ajout d'une seconde interface réseau sur la VM-nginx.....	.11
4. Configuration de la nouvelle interface.....	.12
D. Création d'un premier conteneur web.....	.12
E. ANNEXES.....	.15
1. Configuration de la VM - Nginx.....	.15
2. Conteneur db1.....	.16
3. Configuration du serveur web1.....	.17
4. Tests.....	.19

Version de la partie 2 du TP mise à jour pour palier les différents problèmes rencontrés par les étudiants sur leurs machines personnelles (Manque de place, manque de ressources, processeur ARM, non connaissance des modes réseaux dans VirtualBox... etc)

- Une VM Proxmox pré-installée est également mise à disposition

Février 2023

A. Résumé de l'épisode précédent

Document réalisé avec VirtualBox 7.0.6

1. Travaux réalisés

Dans la première partie du TP Proxmox, vous avez :

- Installé Proxmox dans une machine virtuelle Virtualbox possédant 2 disques durs.
- Accédé à l'interface de Proxmox.
- Configuré Proxmox pour que les machines virtuelles et les conteneurs soient stockés sur le second disque dur.
- Créé des pools, groupes, utilisateurs.
- Défini des permissions pour ces groupes.
- Créé une VM Debian 11 dans Proxmox.
- Créé un conteneur Debian 11 dans Proxmox.
- Transformé la VM et le conteneur en *template*.

2. Problèmes rencontrés

Le réseau :

Comment accéder à l'interface web de Proxmox ?

Quel mode réseau (dans VirtualBox) choisir pour sa VM Proxmox ?

Il faut trouver une solution pour que :

- La VM Proxmox (et les VM / conteneurs créés dans Proxmox) aient accès à Internet
- L'hôte (Le système sur votre laptop) aient accès à la VM Proxmox (et si possible aux VM / conteneurs créés dans Proxmox)

3. Réponse(s)

On peut envisager plusieurs solutions pour le réseau de la VM Proxmox :

- NAT
- Accès par pont
- Réseau NAT

En **mode NAT**, une machine virtuelle VirtualBox aura accès à l'extérieur mais sera seule dans son propre réseau. Ce n'est pas trop envisageable pour qu'on puisse connecter les VM / Conteneurs de Proxmox en mode bridge (accès par pont)

De plus, pour avoir accès à la VM Proxmox, il faudra configurer des redirections de ports.

En **mode Accès par pont**, ça fonctionne pas mal, la VM Proxmox sera sur le même réseau que l'hôte (c'est à dire le réseau wifi de l'IUT) en utilisant la connexion réseau de l'hôte. C'est cependant peu pratique, les IP sont compliquées à retenir, elles risquent de changer au gré du serveur DHCP de l'IUT, etc.

Documentation sur le pontage : <https://wiki.debian.org/fr/BridgeNetworkConnections>

Le **mode réseau NAT** vient palier le problème principal du mode NAT qui est qu'une VM est seule dans son propre réseau. Il faut cependant configurer ce réseau NAT dans Virtualbox (c'est trivial) et il faudra également configurer les redirections de ports pour accéder aux différentes machines (C'est aussi assez simple à réaliser, autant faut-il savoir le faire et ce que cela implique).

Les choses vont se corser (gentiment) dans cette suite, car il va être question ici de créer une petite infrastructure de serveurs web / bases de données / load-balancing / reverse-proxy.

Il va donc falloir également mettre tout ce beau monde en réseau dans Proxmox.

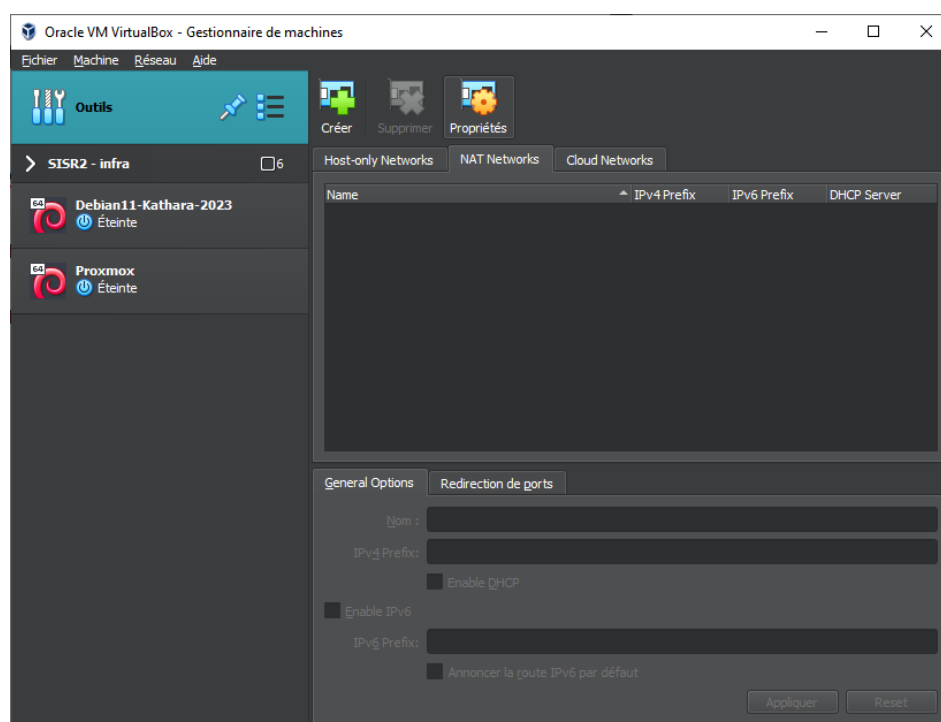
Comment faire le réseau NAT ?

Ici, je vous propose une solution pour configurer un **réseau NAT dans Virtualbox** pour votre VM Proxmox ainsi que les redirections de ports nécessaires (*à compléter plus tard par d'autres si besoin*)

Création d'un réseau NAT dans VirtualBox

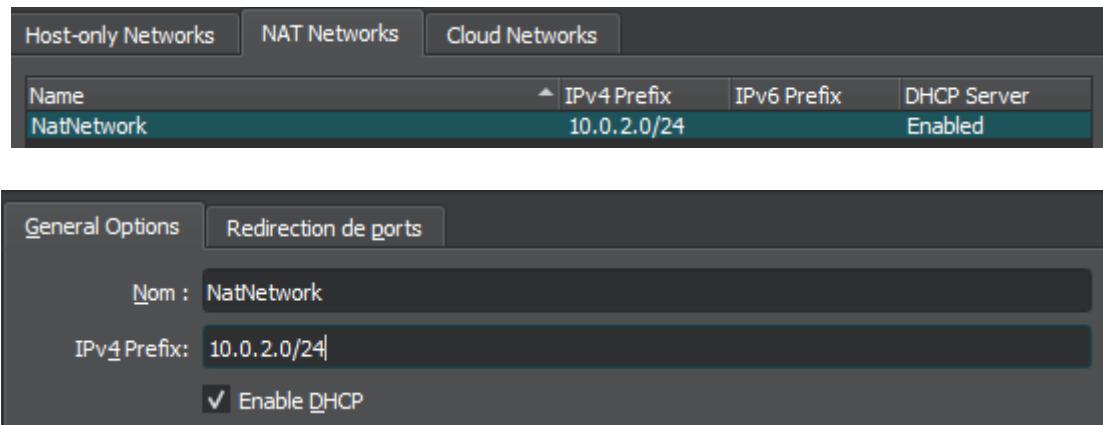
Dans la version 7.* de VirtualBox : Fichier → Outils → Network Manager (CTRL + H) → onglet NAT Networks

Dans la version 6.* de VirtualBox : Fichier → Paramètres → Réseau



Interface de création et configuration de réseaux NAT dans Virtualbox

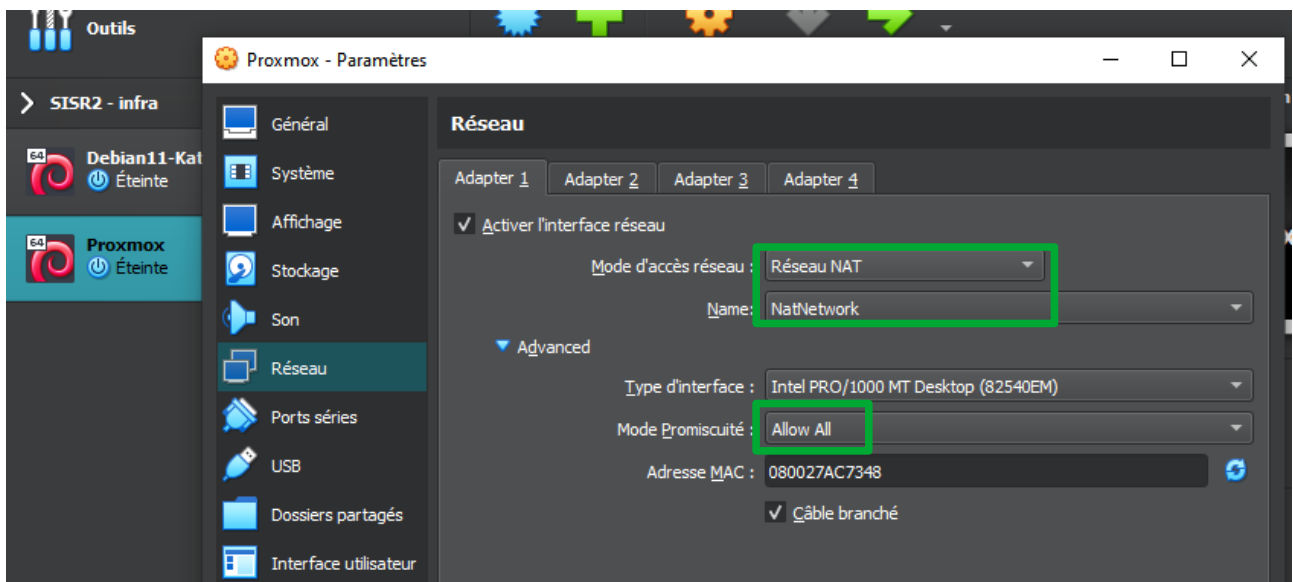
Cliquer sur **Créer +**



The screenshot shows the Proxmox web interface for configuring NAT Networks. At the top, there are three tabs: 'Host-only Networks', 'NAT Networks', and 'Cloud Networks'. Below the tabs is a table with the following columns: 'Name', 'IPv4 Prefix', 'IPv6 Prefix', and 'DHCP Server'. The table contains one entry: 'NatNetwork' with an IPv4 Prefix of '10.0.2.0/24' and 'Enabled' for the DHCP Server. Below the table, there are two tabs: 'General Options' and 'Redirection de ports'. The 'General Options' tab is active, showing the 'Nom' (Name) as 'NatNetwork', the 'IPv4 Prefix' as '10.0.2.0/24', and a checked checkbox for 'Enable DHCP'.

Par défaut, le réseau créé est 10.0.2.0/24 (la passerelle par défaut est 10.0.2.2)

On va pouvoir mettre la VM Proxmox en mode Réseau NAT avec l'IP 10.0.2.50 par exemple.



The screenshot shows the Proxmox web interface for configuring a VM's network settings. The left sidebar shows a list of VMs: 'SISR2 - infra', 'Debian11-Kat', and 'Proxmox'. The 'Proxmox' VM is selected, and the 'Réseau' (Network) tab is active. The 'Réseau' tab shows the 'Adapter 1' configuration. The 'Mode d'accès réseau' (Network access mode) is set to 'Réseau NAT' (highlighted with a green box). The 'Name' is set to 'NatNetwork'. Under the 'Advanced' section, the 'Type d'interface' (Interface type) is 'Intel PRO/1000 MT Desktop (82540EM)', the 'Mode Promiscuité' (Promiscuity mode) is 'Allow All' (highlighted with a green box), and the 'Adresse MAC' (MAC address) is '080027AC7348'. The 'Câble branché' (Cable plugged) checkbox is checked.

Ensuite, l'interface web de Proxmox est accessible par https://IP_Proxmox:8006

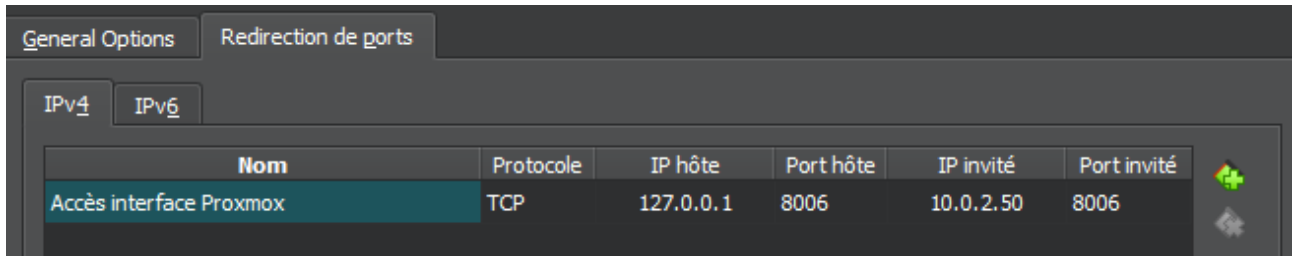
Or, en mode réseau NAT, l'hôte ne peut accéder directement au réseau NAT des VM.

Il faut créer une redirection de ports pour notre réseau NAT afin que l'hôte accède à la VM Proxmox

Redirection de ports

On revient dans la configuration du réseau NAT précédemment créé dans VirtualBox, onglet « Redirection de ports ».

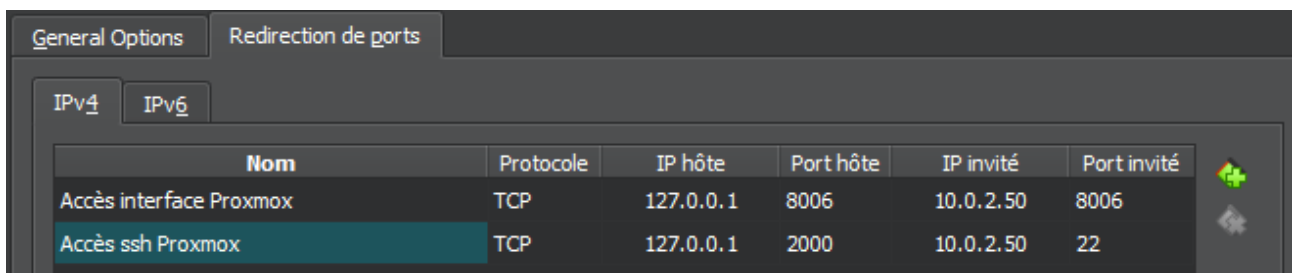
On ajoute une règle pour l'accès à l'interface de Proxmox



Cette règle signifie que lorsque l'on envoie une requête à l'adresse **127.0.0.1** sur le port **8006** de l'hôte, elle sera **redirigée** par Virtualbox vers la machine **10.0.2.50** (La VM Proxmox) sur le port **8006**.

Ce qui permettra d'avoir accès à l'interface web de Proxmox depuis un navigateur sur l'hôte en tapant l'adresse : <https://127.0.0.1:8006>

On pourrait éventuellement rajouter une règle pour se connecter (toujours depuis l'hôte) en **ssh** vers la VM Proxmox



En tapant sur l'hôte la commande **ssh root@127.0.0.1 -p2000**, on se connectera en ssh sur la VM Proxmox sur le port 22.

Il faut évidemment que la VM Proxmox ait pour adresse IP 10.0.2.50/24

Pour vérifier, on lance la VM Proxmox dans VirtualBox, on se connecte avec le compte root.

On édite le fichier **/etc/network/interfaces**

```
auto vmbr0
iface vmbr0 inet static
    address 10.0.2.50/24
    gateway 10.0.2.2
    bridge-ports enp0s3
    bridge-stp off
    bridge-fd 0
```

On modifie les directives address et gateway pour l'interface vmbr0
Address : 10.0.2.50/24
Gateway : 10.0.2.2

On sauvegarde et on redémarre le service réseau (ou la VM) pour prendre en compte les modifications

```
root@pve:~# systemctl restart networking
root@pve:~# _
```

On vérifie

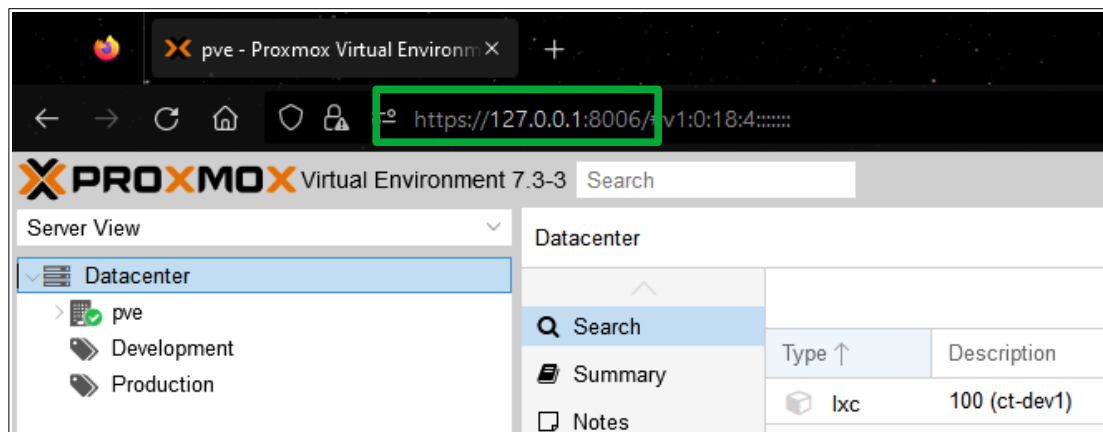
```
root@pve:~# ip -4 -o a show
1: lo        inet 127.0.0.1/8 scope host lo        valid_lft forever preferred_lft forever
5: vmbr0     inet 10.0.2.50/24 scope global vmbr0       valid_lft forever preferred_lft forever
```

Accès Internet

```
root@pve:~# ping www.google.fr
PING www.google.fr (172.217.18.195) 56(84) bytes of data.
64 bytes from par10s38-in-f3.1e100.net (172.217.18.195): icmp_seq=1 ttl=114 time=19.7 ms
64 bytes from par10s38-in-f3.1e100.net (172.217.18.195): icmp_seq=2 ttl=114 time=12.9 ms
64 bytes from par10s38-in-f3.1e100.net (172.217.18.195): icmp_seq=3 ttl=114 time=12.7 ms
```

Test de l'accès à l'interface web de Proxmox

On rappelle que VirtualBox redirige les requêtes vers 127.0.0.1:8006 vers 10.0.2.50:8006



Test de connexion ssh sur la VM Proxmox

```
fls@cygnus:~$ ssh root@127.0.0.1 -p2000
root@127.0.0.1's password:
Linux pve 5.15.74-1-pve #1 SMP PVE 5.15.74-1 (Mon, 07 Nov 2022 20:17:15 +0100)
x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Wed Feb  1 13:56:03 2023
root@pve:~#
```

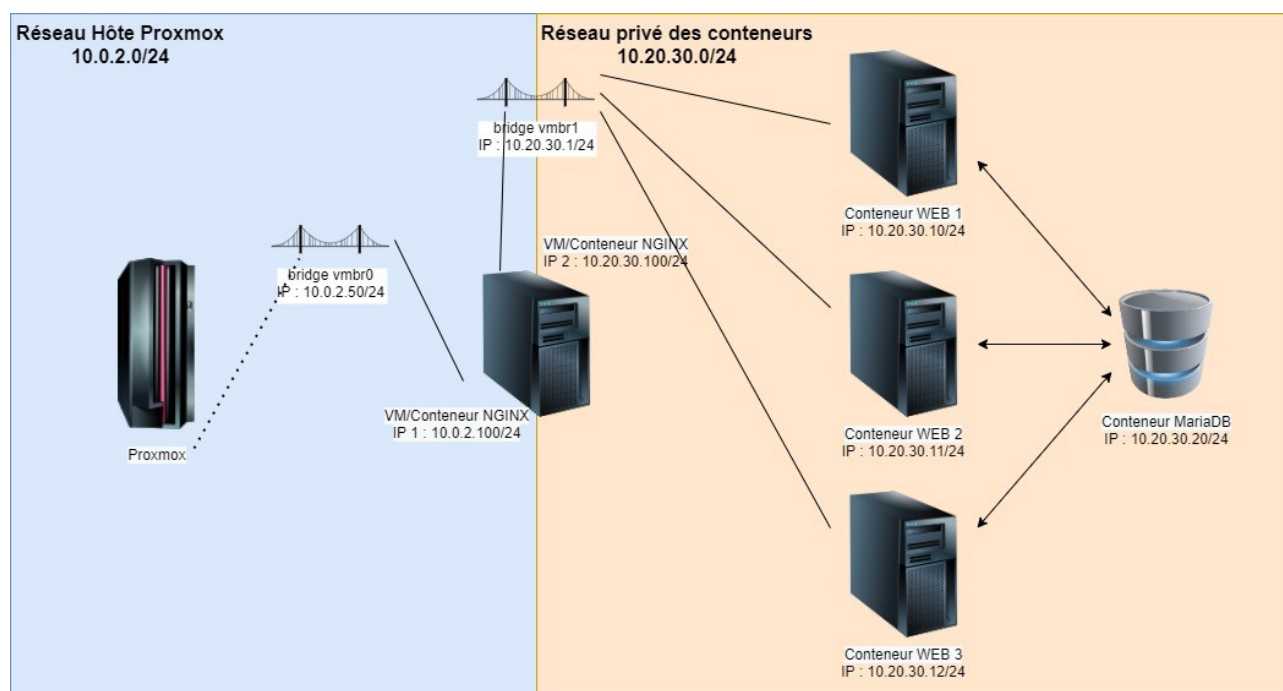
Note : Par défaut sur Proxmox, la connexion ssh avec le compte root est possible. Ce n'est pas le cas pour debian. Il convient de créer un compte utilisateur dans une VM ou un conteneur Debian afin de s'y connecter en ssh.

TAF

- ✓ Terminer (si ce n'est pas le cas) la configuration du serveur Proxmox de la partie 1 du TP, en configurant le réseau NAT dans VirtualBox.
- ✓ Une VM Virtualbox est disponible si besoin sur mon serveur, elle contient :
 - Proxmox installé (IP : 10.0.2.50/24) avec les 2 disques durs
 - Une VM Debian 11 (convertie en template) installée dans Proxmox (car c'est long !)

B. Objectif de la partie 2 du TP

Le but de cette partie 2 est de mettre en place une petite infrastructure réseau dans Proxmox afin de déployer (et load-balancer) une petite application web (i.e. un site php/mysql)



Elle est constituée d'un serveur Proxmox qui contient :

- Une machine (VM ou conteneur) Debian + Nginx
- Trois conteneurs web (Apache)
- Un conteneur de bases de données (MariaDB)

Les trois serveurs web contiendront la même application web/php et le conteneur de bases de données permettra de stocker les données nécessaires à l'application.

La machine Nginx servira à la fois de reverse-proxy HTTP et de répartiteur de charge (load-balancing). Ce sera la seule machine exposée à l'extérieur.

Deux réseaux :

- 10.0.2.0/24 (Réseau « *public* » d'accès à Internet)
- 10.20.30.0/24 (Réseau privé des conteneurs)

La machine Nginx possèdera 2 interfaces réseau :

- Une interface sur le réseau 10.0.2.0/24 afin de pouvoir être contactée de l'extérieur.
- Une interface sur le réseau 10.20.30.0/24 afin de rediriger les requêtes HTTP aux différents serveurs web.

C. Configuration de la VM (ou conteneur) Nginx

Si vous utilisez la VM disponible sur mon serveur : **Cloner** le template vm-template en vm-nginx.

Elle dispose d'un compte utilisateur (std / password) si vous avez besoin de vous connecter en ssh (*une redirection de ports dans Virtualbox sera nécessaire*)

1. Configuration du réseau de vm-nginx

Dans le fichier /etc/network/interfaces, on définit son adresse IP et sa passerelle par défaut.

```
# The primary network interface
auto ens18
iface ens18 inet static
    address 10.0.2.100/24
    gateway 10.0.2.2
```

Note : le nom de l'interface ens18 peut varier selon votre configuration

Redémarrer service réseau

```
systemctl restart networking
```

Tester l'accès internet de la machine vm-nginx et corriger si besoin.

Ajouter des règles de portforwarding pour ssh et web (80)

IPv4		IPv6				
		Nom	Protocole	IP hôte	Port hôte	Port invité
		Accès interface Proxmox	TCP	127.0.0.1	8006	10.0.2.50 8006
		Accès ssh Nginx	TCP	127.0.0.1	2001	10.0.2.100 22
		Accès ssh Proxmox	TCP	127.0.0.1	2000	10.0.2.50 22
		Accès web Nginx	TCP	127.0.0.1	8000	10.0.2.100 80

2. Ajout et configuration 2ème bridge

Pour connecter les conteneurs sur un réseau privé, il faut créer dans Proxmox un second **bridge** (vmbr1 – vmbr0 étant le bridge par défaut)

pve →
network →
create →
Linux bridge

Create: Linux Bridge

Name: Autostart: ☒

IPv4/CIDR: VLAN aware: ☐

Gateway (IPv4): Bridge ports:

IPv6/CIDR: Comment:

Gateway (IPv6):

MTU:

Help Advanced ☒ Create

On attribue à ce bridge la première adresse du réseau soit : 10.20.30.1/24

On se retrouve avec :

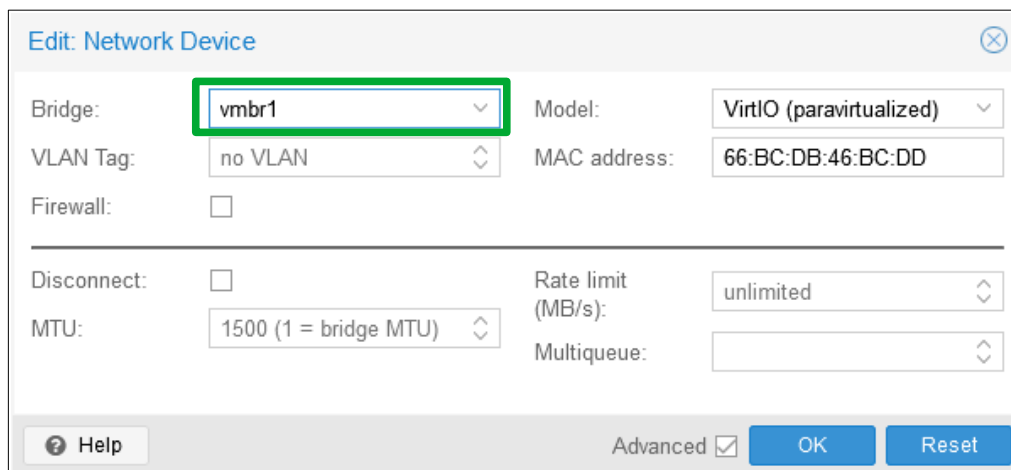
Search	Create	Revert	Edit	Remove	Apply Configuration				
Summary	Name ↑	Type	Active	Autostart	VLAN a...	Ports/Slaves	Bond Mode	CIDR	Gateway
Notes	enp0s3	Network Device	Yes	No	No				
Shell	vmbr0	Linux Bridge	Yes	Yes	No	enp0s3		10.0.2.50/24	10.0.2.2
System	vmbr1	Linux Bridge	No	Yes	No			10.20.30.1/24	
Network									
Certificates									
DNS									

**Ne pas oublier d'appuyer sur le bouton
« Apply Configuration »
sinon le bridge ne sera pas créé.**

3. Ajout d'une seconde interface réseau sur la VM-nginx

On peut maintenant ajouter une seconde interface réseau sur la VM Nginx afin de pouvoir la connecter au réseau privé des conteneurs.

vm-nginx →
 Hardware →
 add →
 network device



Créer la nouvelle interface en la connectant au bridge créé précédemment (vmbr1)

Note : il faut redémarrer la VM pour que l'interface soit réellement créée.

Hard Disk (scsi0)	hdd2:100/vm-100-disk-0.qcow2,iotread=1,size=8G
Network Device (net0)	virtio=DE:06:F3:BF:E3:DF,bridge=vmbr0
Network Device (net1)	virtio=66:BC:DB:46:BC:DD,bridge=vmbr1

Après redémarrage, on vérifie la présence de l'interface sur la machine vm-nginx.

```
root@vm-nginx:~# ifconfig -a
ens18: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 10.0.2.100 netmask 255.255.255.0 broadcast 10.0.2.255
    inet6 fe80::dc06:f3ff:febf:e3df prefixlen 64 scopeid 0x20<link>
    ether de:06:f3:bf:e3:df txqueuelen 1000 (Ethernet)

ens19: flags=4098<BROADCAST,MULTICAST> mtu 1500
    ether 66:bc:db:46:bc:dd txqueuelen 1000 (Ethernet)
```

Ici, elle se nomme **ens19**.

4. Configuration de la nouvelle interface

Il faut maintenant configurer la nouvelle interface conformément au schéma proposé, c'est à dire avec l'IP **10.20.30.100/24**

```
# The secondary network interface
auto ens19
iface ens19 inet static
    address 10.20.30.100/24
```

On redémarre le service réseau puis on vérifie

```
root@vm-nginx:~# ip -4 -o a show
1: lo      inet 127.0.0.1/8 scope host lo\
2: ens18   inet 10.0.2.100/24 brd 10.0.2.255 scope global ens18\
3: ens19   inet 10.20.30.100/24 brd 10.20.30.255 scope global ens19\
```

On vérifie la connexion internet, corriger en cas de problème.

D. Création d'un premier conteneur web

Créer (*bouton en haut à droite de l'interface Proxmox*) un conteneur Debian 11 dans Proxmox avec les caractéristiques suivantes :

Hostname : ct-web1

Ressource Pool : Development

Choisir le template debian 11 téléchargé dans la partie 1 du TP.

Storage : hdd2

Disque : 2Go

Core CPU : 1

RAM : 512

Network : **vmbri1**

IPv4 : 10.20.30.10/24

gateway : 10.20.30.1

On démarre le conteneur créé et on teste

```
root@ct-web1:~# ip -4 -o a show
1: lo      inet 127.0.0.1/8 scope host lo\          valid_lft forever preferred_lft forever
2: eth0    inet 10.20.30.10/24 brd 10.20.30.255 scope global eth0\      valid_lft forever preferred_lft forever
root@ct-web1:~# ip route
default via 10.20.30.1 dev eth0 onlink
10.20.30.0/24 dev eth0 proto kernel scope link src 10.20.30.10
```

Si on teste l'accès internet du conteneur, **cela ne fonctionne pas**. Le conteneur est sur un réseau privé 10.20.30.0/24 (voir RFC 1918) qui ne peut pas avoir accès à Internet sans NAT.

Il va donc falloir « Natter » le réseau 10.20.30.0/24 au niveau du serveur Proxmox.

Si on édite le fichier **/etc/network/interfaces**, on trouvera la configuration pour les deux bridges **vmbr0** et **vmbr1**.

Vmbr0 est le bridge par défaut de Proxmox et vmbr1 et le second bridge ajouté précédemment pour le réseau privé des conteneurs.

On va ajouter des règles iptables pour natter le réseau 10.20.30.0/24 lors de la création du pont vmbr1 et également une directive pour supprimer cette règle lorsque le pont sera déconnecté.

On active également le routage.

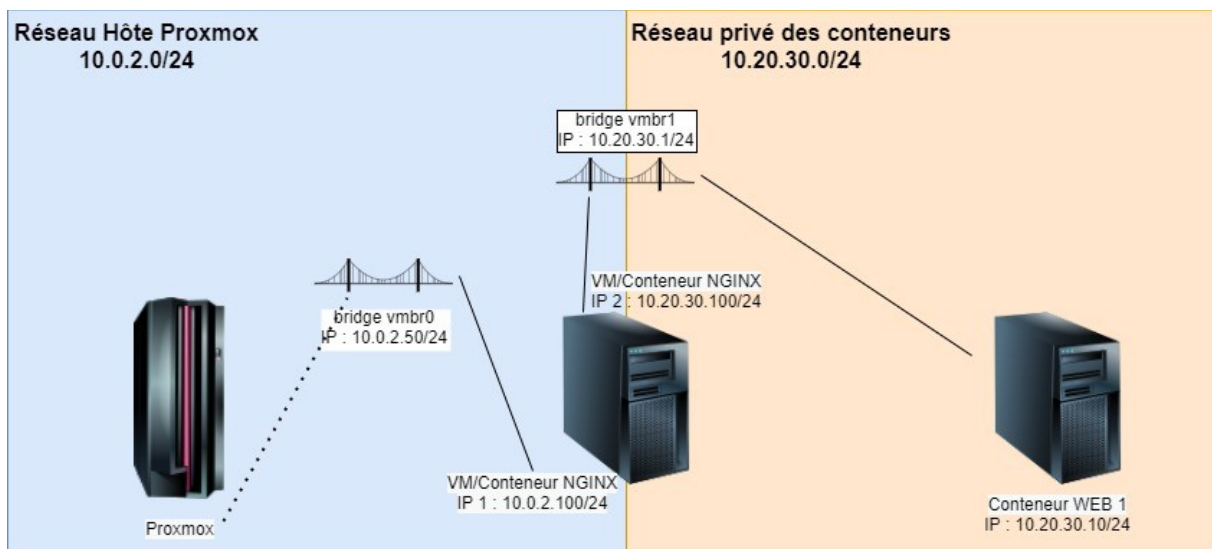
```
auto vmbr1
iface vmbr1 inet static
    address 10.20.30.1/24
    bridge-ports none
    bridge-stp off
    bridge-fd 0
    post-up echo 1 > /proc/sys/net/ipv4/ip_forward
    post-up iptables -t nat -A POSTROUTING -s '10.20.30.0/24' -o vmbr0 -j MASQUERADE
    post-down iptables -t nat -D POSTROUTING -s '10.20.30.0/24' -o vmbr0 -j MASQUERADE
```

On redémarre le service réseau sur Proxmox et on teste l'accès net dans le conteneur web.

```
root@ct-web1:~# ping -c4 www.google.fr
PING www.google.fr (142.250.201.163) 56(84) bytes of data.
64 bytes from par21s23-in-f3.1e100.net (142.250.201.163): icmp_seq=1 ttl=113 time=13.1 ms
64 bytes from par21s23-in-f3.1e100.net (142.250.201.163): icmp_seq=2 ttl=113 time=13.6 ms
64 bytes from par21s23-in-f3.1e100.net (142.250.201.163): icmp_seq=3 ttl=113 time=13.8 ms
64 bytes from par21s23-in-f3.1e100.net (142.250.201.163): icmp_seq=4 ttl=113 time=12.8 ms

--- www.google.fr ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3007ms
rtt min/avg/max/mdev = 12.848/13.335/13.784/0.377 ms
```

Pour le moment, l'architecture de vos machines ressemble à ceci :



TAF

- ✓ Installer apache2, php, php-mysql sur le conteneur web1
- ✓ Déployer une application php dans le conteneur web1 (un exemple est donné en annexe)
- ✓ Créer (par clonage) et configurer (IP) les conteneurs web 2 et 3
- ✓ Créer le conteneur et configurer (IP) db1, installer mariadb-server, créer la base de données pour l'application web
- ✓ Installer et configurer nginx en tant que reverse-proxy dans la VM nginx (voir annexe)
- ✓ Tester

E. ANNEXES

1. Configuration de la VM - Nginx

On installe nginx

```
root@vm-nginx:~# apt update
root@vm-nginx:~# apt install nginx
```

On supprime la configuration par défaut

```
root@vm-nginx:~# rm /etc/nginx/sites-enabled/default
```

Configuration minimale du reverse-proxy / load-balancing

```
root@vm-nginx:~# nano /etc/nginx/conf.d/load-balancer.conf
```

```
upstream backend {
    server 10.20.30.10;
    server 10.20.30.11;
    server 10.20.30.12;
}
server {
    listen 80;
    server_name 10.0.2.100;
    location / {
        proxy_pass http://backend;
    }
}
```

Explications rapides : on crée une « liste » (upstream backend) de serveurs web.
On fait écouter nginx sur le port 80

Toutes les requêtes seront redirigées (grâce à la directive proxy_pass) vers les serveurs définis dans l'upstream (à tour de rôle)

il est possible de définir un « poids » particulier sur chaque serveur (ex : weight = 2) afin de répartir la charge différemment selon les besoins.

On redémarre nginx après configuration

```
root@vm-nginx:~# systemctl restart nginx
```

2. Conteneur db1

Taille du conteneur (2Go)

Installation et configuration de MariaDB

```
root@ct-db1:~# apt update
root@ct-db1:~# apt install mariadb-server
```

```
root@ct-db1:~# mysql_secure_installation
```

Prendre tous les choix par défaut.

Création d'une base de données et d'un utilisateur (pour l'exemple).

```
root@ct-db1:~# mysql -u root
```

```
mysql> create database store;
mysql> create user 'dba'@'%' identified by 'password';
mysql> grant all on store.* to 'dba'@'%' ;
mysql> flush privileges;
```

Création d'une table et d'enregistrements dans la base de données

```
use store

CREATE TABLE `products` (
  `id` int(11) NOT NULL AUTO_INCREMENT PRIMARY KEY,
  `name` varchar(100) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

INSERT INTO `products` (name) VALUES ("RTX 3060");
INSERT INTO `products` (name) VALUES ("RTX 3070");
INSERT INTO `products` (name) VALUES ("RTX 3080");

exit
```

Par défaut, MariaDB n'accepte les requêtes que depuis l'adresse 127.0.0.1 (boucle locale).

Cependant, l'application php est hébergée sur une autre machine (web 1, 2, 3) donc il faut permettre à ces applications d'interroger la base de données.

On installe net-tools pour vérifier avec la commande netstat.

```
root@ct-db1:~# apt install net-tools
```

```
root@ct-db1:~# netstat -nat
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 127.0.0.1:3306          0.0.0.0:*               LISTEN
```


On modifie la configuration de MariaDB pour changer ça.

```
root@ct-db1:~# nano /etc/mysql/mariadb.conf.d/50-server.cnf
```

Remplacer

```
bind-address          = 127.0.0.1
```

Par

```
bind-address          = 0.0.0.0
```

On redémarre le service mariadb

```
root@ct-db1:~# systemctl restart mariadb
```

On vérifie les changements

```
root@ct-db1:~# netstat -nat
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 0.0.0.0:3306             0.0.0.0:*               LISTEN
```

3. Configuration du serveur web1

On installe ce dont on a besoin, c'est à dire apache2, php et php-mysql

```
root@ct-web1:~# apt update
root@ct-web1:~# apt install apache2 php php-mysql
```

Application php/mysql de test (Oui, je sais, on ne juge pas !)

Fichier index.php

```
<?php
$db="store";
$dbhost="10.20.30.20";
$dbport=3306;
$dbuser="dba";
$dbpasswd="password";

try {
    $pdo = new PDO('mysql:host='.$dbhost.';port='.$dbport.';dbname='.$db.',
$dbuser, $dbpasswd);
    $pdo->exec("SET CHARACTER SET utf8");
    $pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

    $stmt = $pdo->prepare("SELECT * FROM products");
```

```

$stmt->execute();

$res = $stmt->fetchAll();
echo "<h1>Instance 1</h1>";
foreach ( $res as $row ) {
    echo $row['id'] . " - " . $row['name'] . "<br>";
}
} catch(PDOException $e) {
    echo $e->getMessage();
}

echo "<br>";

echo "
<form action='insert.php' method='post'>
Nom: <input type='text' name='name' />
<input type='submit' value='Inserer' />
</form>

<form action='delete.php' method='post'>
ID: <input type='text' name='id' size='3' />
<input type='submit' value='Supprimer' />
</form>
";

$pdo = null;
?>

```

Fichier insert.php

```

<?php
$db="store";
$dbhost="10.20.30.20";
$dbport=3306;
$dbuser="dba";
$dbpasswd="password";

try {
    $pdo = new PDO('mysql:host='.$dbhost.';port='.$dbport.';dbname='.$db.',
$dbuser, $dbpasswd);
    $pdo->exec("SET CHARACTER SET utf8");
    $pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

    $sql = "INSERT INTO `products` (name) VALUES ('$_POST[name]')";
    $pdo->exec($sql);
} catch(PDOException $e) {
    echo $e->getMessage();
}

$pdo = null;
header('Location: ./index.php');
exit;
?>

```

```

<?php
$db="store";
$dbhost="10.20.30.20";
$dbport=3306;
$dbuser="dba";
$dbpasswd="password";

echo $_POST[name];

try {
    $pdo = new PDO('mysql:host='.$dbhost.';port='.$dbport.';dbname='.$db.'',
$dbuser, $dbpasswd);
    $pdo->exec("SET CHARACTER SET utf8");
    $pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

    $sql = "DELETE FROM `products` WHERE `id` = $_POST[id];";
    $pdo->exec($sql);
} catch(PDOException $e) {
    echo $e->getMessage();
}
$pdo = null;
header('Location: ./index.php');
exit;
?>

```

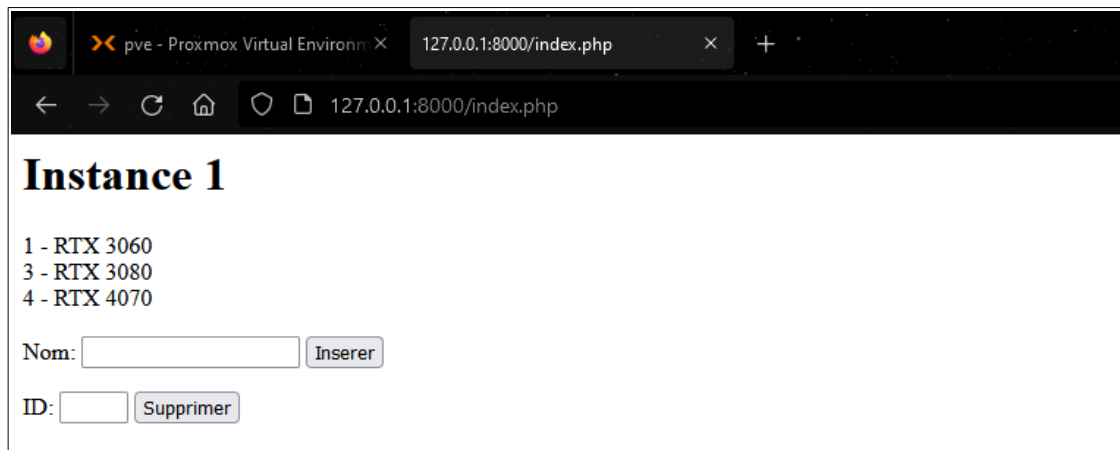
4. Tests

Si vous avez bien créé les règles de portforwarding dans Virtualbox

IPv4					
IPv6					
Nom	Protocole	IP hôte	Port hôte	IP invité	Port invité
Accès interface Proxmox	TCP	127.0.0.1	8006	10.0.2.50	8006
Accès ssh Nginx	TCP	127.0.0.1	2001	10.0.2.100	22
Accès ssh Proxmox	TCP	127.0.0.1	2000	10.0.2.50	22
Accès web Nginx	TCP	127.0.0.1	8000	10.0.2.100	80

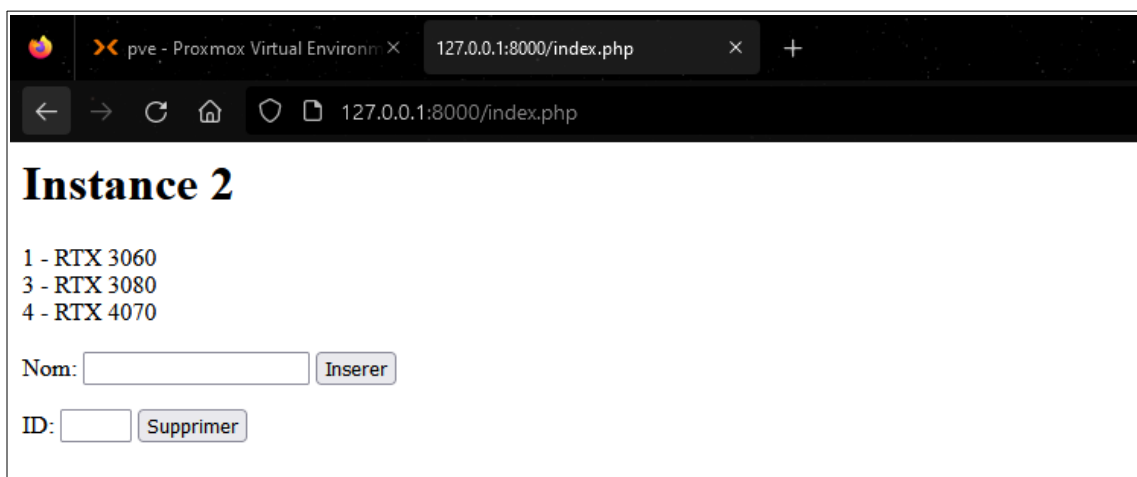
vous pouvez accéder en HTTP à la machine Nginx par l'URL : <http://127.0.0.1:8000>

On tombe alors sur l'application hébergée sur web1



A screenshot of a web browser window. The address bar shows '127.0.0.1:8000/index.php'. The page title is 'Instance 1'. Below the title, there is a list of RTX cards: '1 - RTX 3060', '3 - RTX 3080', and '4 - RTX 4070'. Below the list, there are two input fields: 'Nom:' and 'ID:'. The 'Nom:' field has an 'Inserer' button next to it. The 'ID:' field has a 'Supprimer' button next to it.

En faisant « f5 » sur le navigateur pour rafraîchir la page on voit, qu'alternativement, on tombe sur les 3 serveurs web (**nb** : modifier le code html pour visualiser le changement de serveur, ex : Instance 1, 2, 3).



A screenshot of a web browser window. The address bar shows '127.0.0.1:8000/index.php'. The page title is 'Instance 2'. Below the title, there is a list of RTX cards: '1 - RTX 3060', '3 - RTX 3080', and '4 - RTX 4070'. Below the list, there are two input fields: 'Nom:' and 'ID:'. The 'Nom:' field has an 'Inserer' button next to it. The 'ID:' field has a 'Supprimer' button next to it.



A screenshot of a web browser window. The address bar shows '127.0.0.1:8000/index.php'. The page title is 'Instance 3'. Below the title, there is a list of RTX cards: '1 - RTX 3060', '3 - RTX 3080', and '4 - RTX 4070'. Below the list, there are two input fields: 'Nom:' and 'ID:'. The 'Nom:' field has an 'Inserer' button next to it. The 'ID:' field has a 'Supprimer' button next to it.