Installation

Configurations

Fichier rc.local

• Voici respectivement les configurations IP automatique des deux machines (fichier rc.local routeur et serveur)

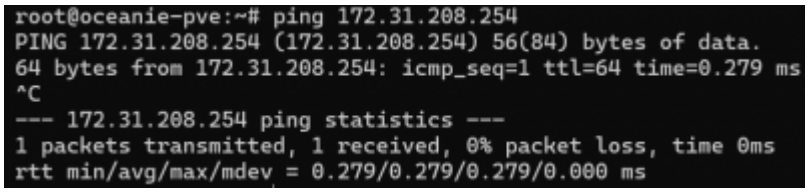
```
#!/bin/sh -e
#Indique que ce script doit être exécuté par /bin/sh. L'option -e indique
que le script doit s'arrêter en cas d'erreur (si une commande échoue).

ifconfig enp4s0 172.31.208.254/16 UP
#Configure l'interface réseau enp4s0 avec l'adresse IP 172.31.208.254 et un
masque de sous-réseau /16. Le paramètre UP active l'interface réseau.

route add default gw 172.31. 0.1
#Définit la passerelle par défaut (default gateway) pour le routage à
l'adresse IP 172.31.0.1. Cela permet au routeur d'envoyer du trafic vers
d'autres réseaux via cette passerelle.

ifconfig enp1s4 10.31.211.254/22 up
#Configure une autre interface réseau, enp1s4, avec l'adresse IP
10.31.211.254 et un masque de sous-réseau /22. Le paramètre up active cette
interface.

1 > /proc/sys/net/ipv4/ip_forward
#Active le forwarding IPv4 sur le système, ce qui permet au routeur de
transférer des paquets entre différentes interfaces réseau (essentiel pour
le rôle de routeur).

echo "nameserver 8.8.8.8" > /etc/resolv.conf
#Définit le serveur DNS à utiliser pour résoudre les noms de domaine. Ici,
il s'agit du serveur DNS public de Google (8.8.8.8).

iptables -t nat -A POSTROUTING -s 10.31.208.0/22 -j MASQUERADE
#Ajoute une règle NAT (Network Address Translation) dans la table nat
d'iptables. La règle MASQUERADE permet de masquer l'adresse IP source des
paquets sortants provenant du réseau 10.31.208.0/22, ce qui est nécessaire
pour que les ordinateurs derrière le routeur puissent accéder à Internet via
une seule adresse IP publique.
```

```
#!/bin/sh -e
ifconfig eno1 10.31.208.1/22 up
route add default gw 10.31.211.254
echo "nameserver 8.8.8.8" > /etc/resolv.conf
```

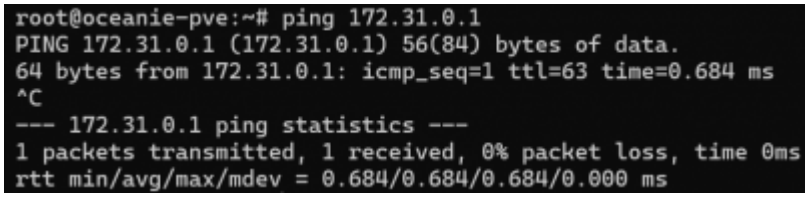
Communication

Toutes les machines doivent pouvoir communiquer entre elles (ping): voici les images illustrant les différents ping de nos machines

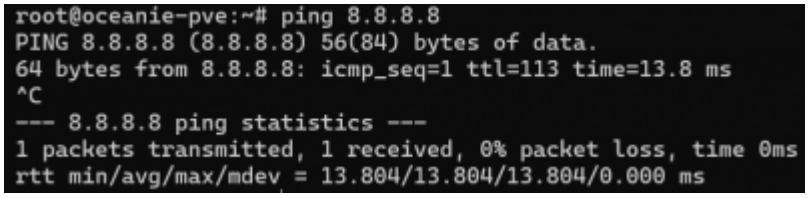
ping du serveur(10.31.208.1 vers le routeur 172.31.208.254



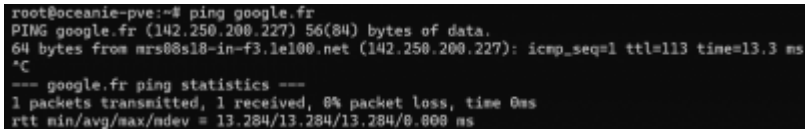
ping du serveur(10.31.208.1 vers le routeur 172.31.0.1 (fls)



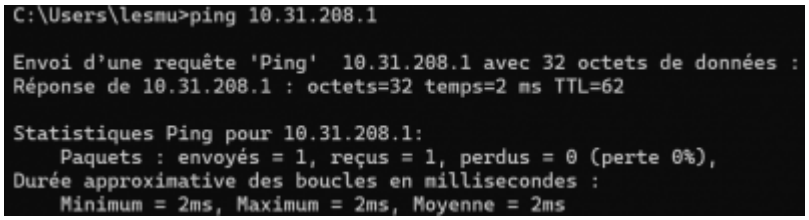
ping du serveur(10.31.208.1 vers le dns public de google 8.8.8.8



ping du serveur(10.31.208.1 vers le dns public de google avec la résolution de nom google.fr



ping du client vers le serveur (10.31.208.1)



```
ping du client vers le routeur (172.31.208.254)
```

```
C:\Users\lesmu>ping 172.31.208.254

Envoi d'une requête 'Ping'  172.31.208.254 avec 32 octets de données :
Réponse de 172.31.208.254 : octets=32 temps=13 ms TTL=63

Statistiques Ping pour 172.31.208.254:
    Paquets : envoyés = 1, reçus = 1, perdus = 0 (perte 0%),
Durée approximative des boucles en millisecondes :
    Minimum = 13ms, Maximum = 13ms, Moyenne = 13ms
```

```
ping du client vers le serveur d'un autre groupe (10.31.200.1)
```

```
root@oceanie-pve:~# ping 10.31.200.1
PING 10.31.200.1 (10.31.200.1) 56(84) bytes of data.
^C
--- 10.31.200.1 ping statistics ---
1 packets transmitted, 0 received, 100% packet loss, time 0ms
```

Caractéristique des machines

Feuille récapitulant les caractéristiques techniques de la machine

Pour avoir les feuilles caractéristiques des deux machines nous avons télécharger l'outil neofetch pour cela rien de plus simple :

```
apt install neofetch
```

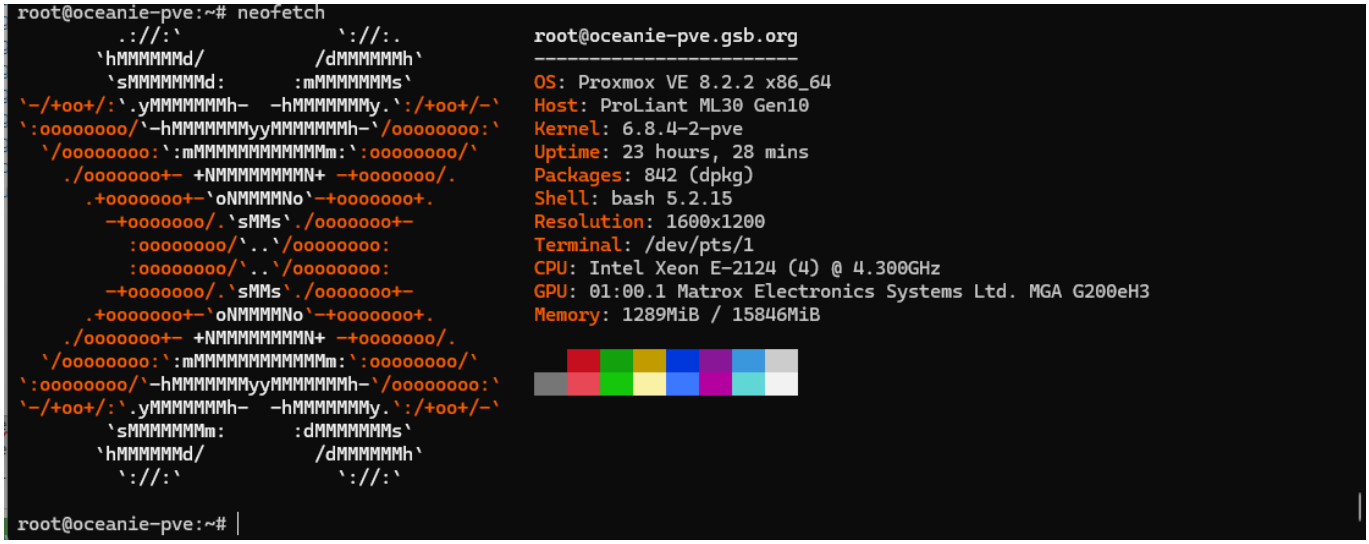
puis pour voir les caractéristiques il suffit de taper la commande

```
neofetch
```

#feuille caractéristique routeur

```
root@oceanie-rtr:~# neofetch
       _,met$$$$$gg.          root@oceanie-rtr.gsb.org
    ,g$$$$$$$$$$$$$$$P.      ------------------------
  ,g$$P"     """Y$$.".        OS: Debian GNU/Linux 12 (bookworm) x86_64
 ,$$P'              `$$$.     Host: ProLiant ML150 G6 1.0
',$$P       ,ggs.     `$$b:   Kernel: 6.1.0-25-amd64
`d$$'     ,$P"'   .    $$$    Uptime: 23 hours, 46 mins
 $$P      d$'     ,    $$P    Packages: 1337 (dpkg)
 $$:      $$.   -    ,d$$'    Shell: bash 5.2.15
 $$;      Y$b._   _,d$P'      Resolution: 1024x768
 Y$$.    `.`"Y$$$$P"'         Terminal: /dev/pts/0
 `$$b      "-.__              CPU: Intel Xeon E5502 (2) @ 1.811GHz
  `Y$$                        GPU: 03:00.0 Matrox Electronics Systems Ltd. MGA G200e [Pilot] ServerEngines
   `Y$$.                      Memory: 525MiB / 7929MiB
     `$$b.
       `Y$$b.
          `"Y$b._
              `"""
```

#feuille caractéristique du serveur



descriptif des compte créer sur les machines

Comptes sur le Routeur : Compte root (administrateur) :

Nom d'utilisateur : root
Description : Le compte root est l'utilisateur super-administrateur du système. Il a tous les droits sur la machine, peut configurer les interfaces réseau, gérer les fichiers système critiques, modifier les règles de pare-feu, etc. Répertoire home : /root

```
  Shell : /bin/bash ou /bin/sh
```

Compte standard :

Nom d'utilisateur : STD
Description : Il s'agit d'un utilisateur non privilégié, utilisé pour les tâches quotidiennes comme les tests de réseau, la gestion des fichiers, mais sans droits administratifs. Ce compte n'a pas les privilèges nécessaires pour modifier la configuration critique du routeur sans élévation de droits (via su - par exemple). Répertoire home : \\

```
 /home/std
```

```
Shell : /bin/bash
```

CONCERNANT LE SERVEUR ?
QUI EST SOUS PROXMOX NOUS N'AVONS PAS EU A CREER UN COMPTE USER (STD) ET LA CONFIGURATION DU COMPTE ADMIN SOUS PROXMOX EST SIMILAIRE À CELLE SUR DEBIAN 12

Test d’accès ssh à partir d’un poste distant

1. Principe de l'authentification par clé SSH L'accès à distance via SSH (Secure Shell) permet de se connecter à une machine distante de manière sécurisée. Il repose généralement sur deux méthodes d'authentification : par mot de passe ou par clés SSH. L'authentification par clé est plus sécurisée car elle repose sur une paire de clés asymétriques, composée d'une clé privée et d'une clé publique.

2. Génération des clés SSH
Pour activer l'accès SSH sans mot de passe, il est nécessaire de générer une paire de clés sur l'hôte (machine locale). Voici la commande pour générer une clé SSH sur un terminal :

```
ssh-keygen -t rsa -b 4096 -C
```

Cette commande crée une paire de clés (publique et privée) dans le répertoire ~/.ssh/ :

Clé privée : ~/.ssh/id_rsa (à garder secrète)
Clé publique : ~/.ssh/id_rsa.pub (à partager avec les machines auxquelles vous souhaitez vous connecter)
3. Ajout manuelle de la clé dans le fichier authorized_keys
la commande ssh-copy-id n'etant pas disponible sur nos machines hôte , nous avons du le faire manuellement :

Nous avons copier la clé publique (~/.ssh/id_rsa.pub) sur la machine distante et nous l'avons placer dans le fichier suivant :

```
.ssh/authorized_keys
```

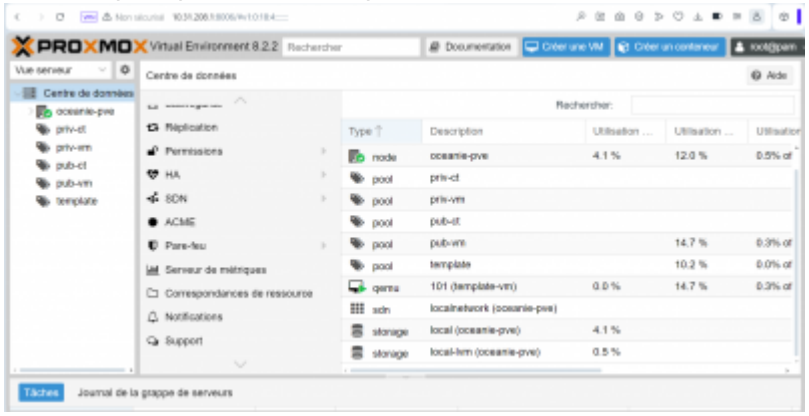
Exemple des clés présentes dans le fichier authorized_keys
Le fichier ~/.ssh/authorized_keys sur la machine distante contient les clés publiques autorisées à se connecter à cette machine.
Voici un exemple des clés ajoutées dans ce fichier :

```
ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAACAQDIvWPmMhhL3W7+0GZLAGEy21tijbW3xj9G0iZ7t+cPmUYK
096v1+OKDZbwBptI6hzSuc4kgNxEHXFgF8Y>ssh-rsa

AAAAB3NzaC1yc2EAAAADAQABAAACAQDNexJg3jXXjKGnNsKzLG6riLG8RMQsV36zU1MK+RfZclNc
fEAzJO5naOSYW8mHFXpAdGpweoPvpKcsGgC>ssh-rsa

AAAAB3NzaC1yc2EAAAADAQABAAACAQCp/iNckoEmepD9TvpBtW+NCAqnHrwJuUOtuu+fsQ6LPNV7
RmzEYuUihxSvs5w2O19apd82QU1e6t2u8r6>ssh-rsa

AAAAB3NzaC1yc2EAAAADAQABAAACAQDBj+eQR0I+OOeErmHrp16c7VxAL+p9S8l53WblQxYPNHGY
4uIFsaRE+BISp3urdWO32kbhn+tlqIjSP/6>
```

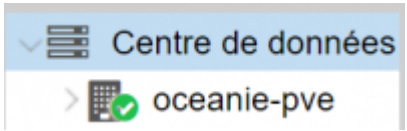
```
C:\Users\USER>ssh root@10.31.208.1
Enter passphrase for key 'C:\Users\USER/.ssh/id_rsa':
Linux oceanie-pve.gsb.org 6.8.4-2-pve #1 SMP PREEMPT_DYNAMIC PMX 6.8.4-2 (2024-04-10T17:36Z) x86_64

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: Mon Sep  9 14:21:38 2024
root@oceanie-pve:~#
```

voici une illustration de la connexion ssh sur une machine distante , il suffit de mettre sa pass phrase créer à la création de la clé pour que la connexion puissent être fonctionnelle

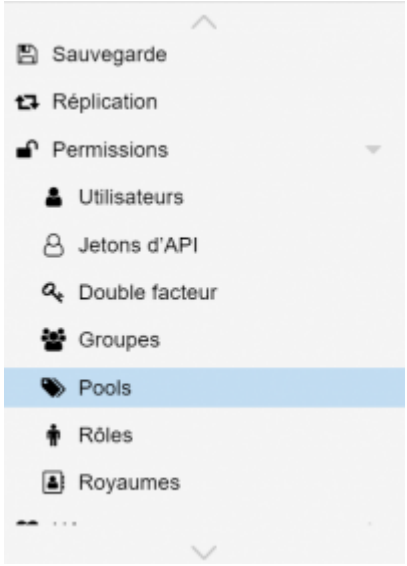
Nous pouvons accéder à ProxMox via un navigateur web tant que notre poste est connecté au même réseau que notre serveur. Nous pouvons donc taper dans la barre d'URL “10.31.208.1:8006” (8006 étant le port par défaut de proxmox), entrer l'identifiant root pour accéder à notre interface :



Sur celui-ci nous allons créer des **pools** qui sont des étiquettes qui nous permettrons de distinguer nos futurs machines virtuelles et conteneurs.
Pour ce faire on se rend sur Proxmox on clique sur “Centre de données” :

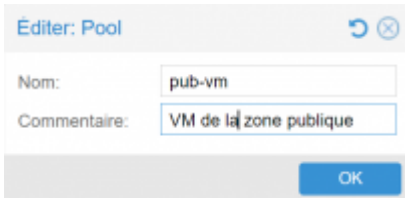


Puis sur “Pools” :

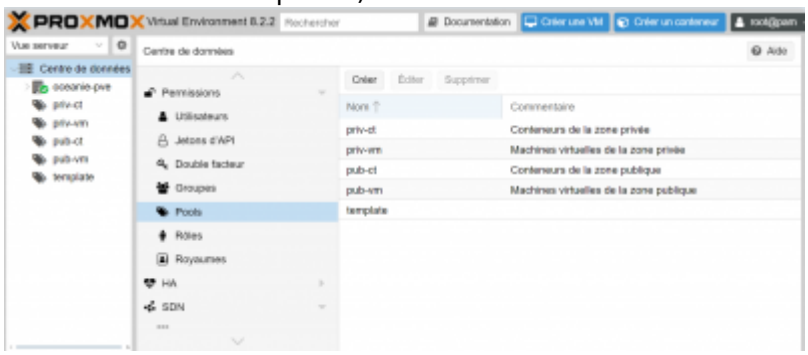


Puis sur créer :

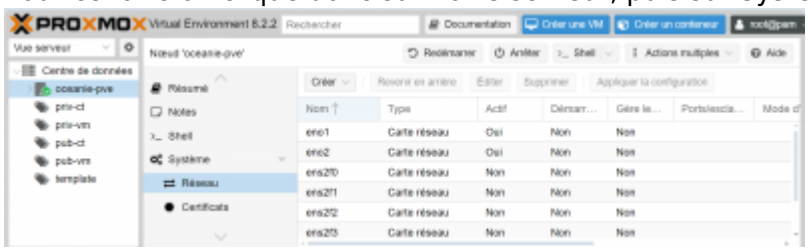
On peut donc ajouter un nom et une description :



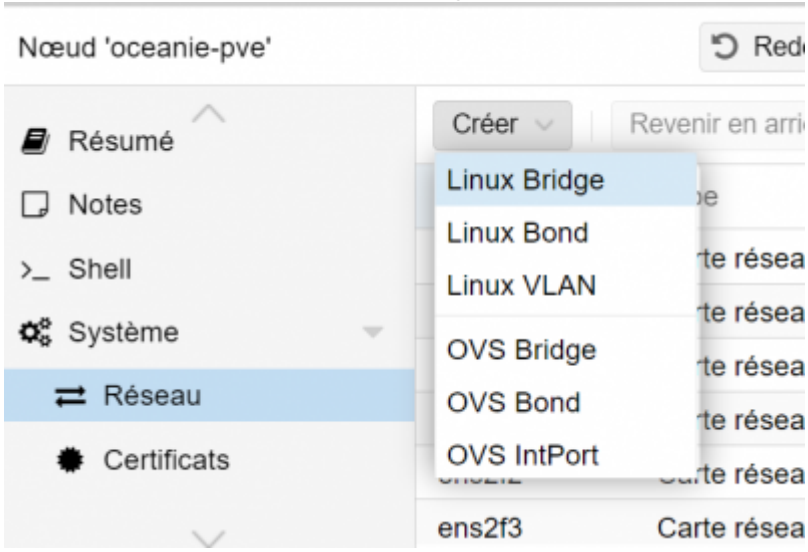
Nous allons donc créer 5 pools : pub-ct (conteneur de la zone publique) ; pub-vm (vm de la zone publique) ; priv-ct (conteneur de la zone privée) ; priv-vm (vm de la zone privée) ; template (qui contiendras nos templates).



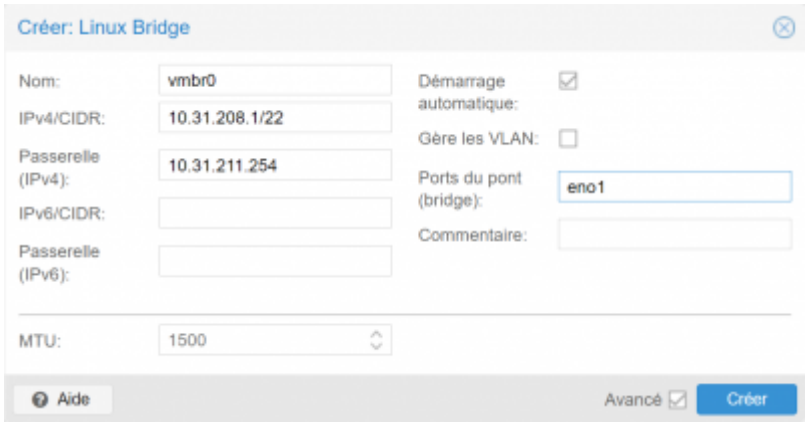
Notre réseau 10.31.208.0/20 a une partie privée et une partie publique, nous devons donc créer deux ponts vmbr0 pour l'interface eno1 et vmbr2 pour l'interface eno2.
Pour ce faire on clique donc sur notre serveur, puis sur Système et Réseau :\\



Ensuite on fais Créer → Linux Bridge :



Enfin nous pouvons remplir les informations : En adresse IP : on va mettre celle du serveur ; en passerelle : la dernière adresse du réseau publique ; en port du pont on utilisera eno1 :

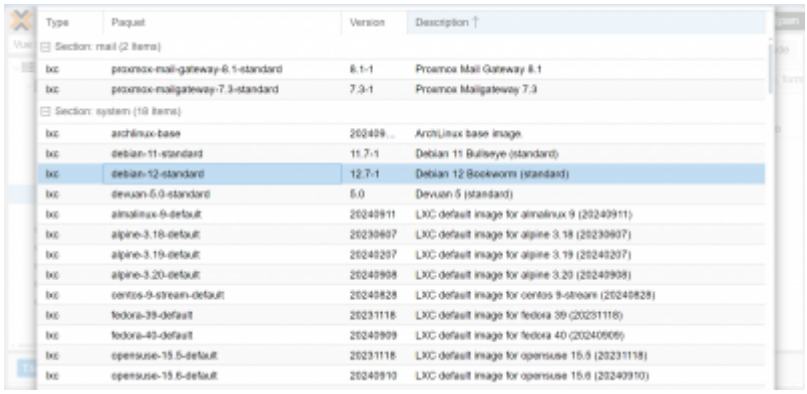


Création des templates

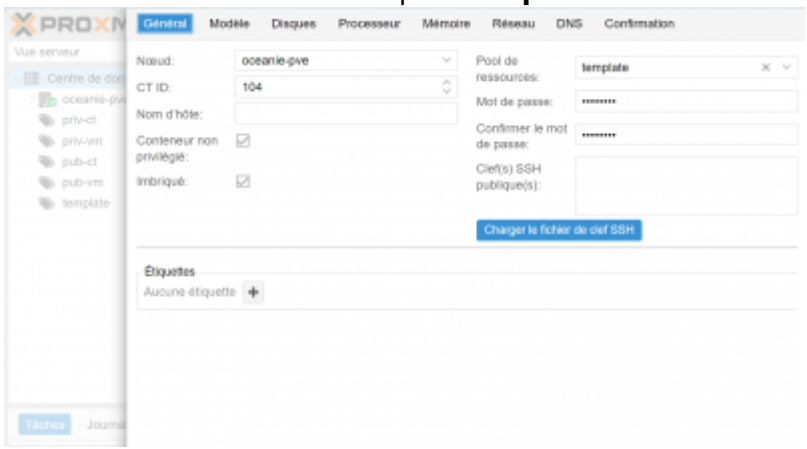
Conteneur

Nous pouvons à présent créer nos templates, nous faire un template conteneur et un autre machine virtuelle.
Pour créer un conteneur on se rend dans Centre de données → océanie-pve → local → modèles de conteneurs :

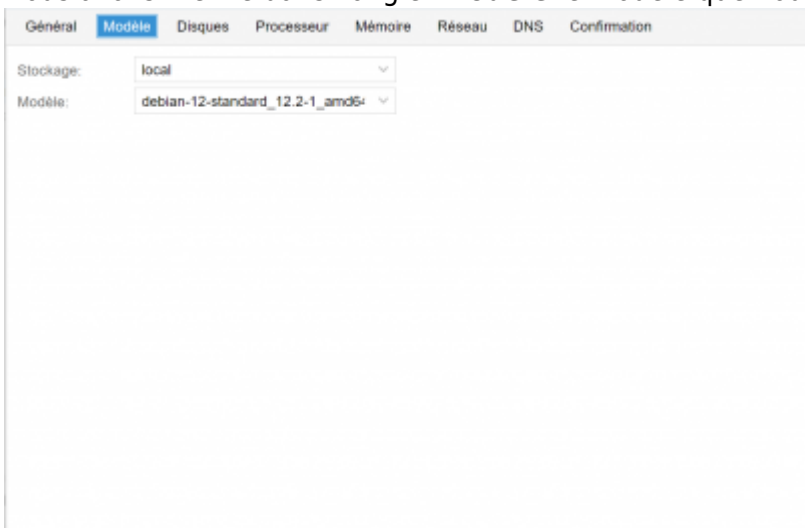
Puis on clique sur Modèles et on choisis notre modèle, on utilisera le modèle debian 12 standard :



Maintenant que notre modèle est installé nous pouvons créer le conteneur. Pour ce faire on va sur **Créer un conteneur**
Nous allons le mettre dans le pool **template** et mettre le mot de passe *password*



Nous allons mettre dans l'onglet **Modèle** le modèle que nous avons télécharger précédemment.



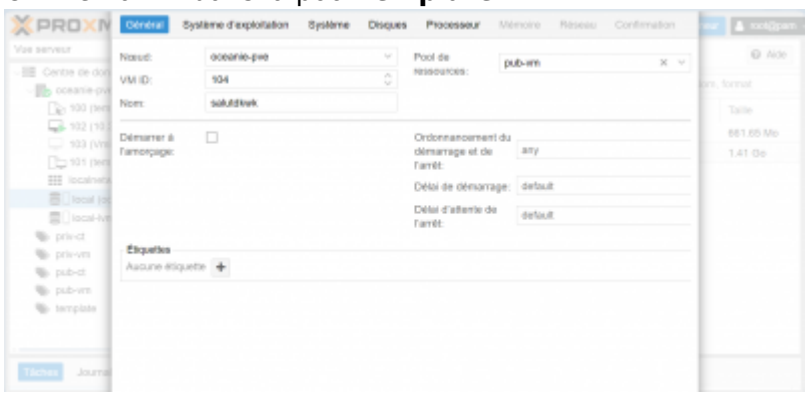
On fera attention à bien configurer le réseau et le DNS avant de valider.

Maintenant que notre conteneur est créé nous allons rentrer dedans, installer les clé SSH, mettre l'IP 10.31.208.2 et installer :

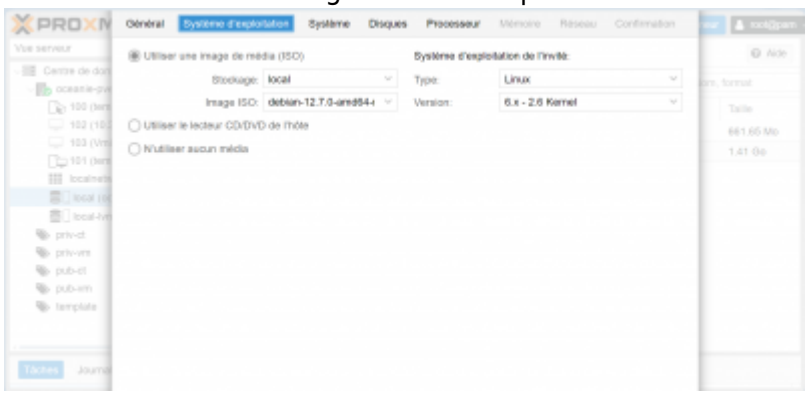
```
apt update
apt upgrade
apt install iptables tcpdump net-tools vim nano inetutils-ping sudo less
cron wget logrotate netcat-traditional ntpdate dnsutils tracroute nmap
rsyslog
```

Machine Virtuelle

Pour l'installation de la machine virtuelle nous allons devoir téléverser une image Debian sur Proxmox. Pour cela on télécharge un fichier ISO de Debian 12, on se rend ensuite sur centre de donnée → oceanie-pve → local → Image ISO → Téléverser.
Une fois le fichier installer dans Proxmox on clique sur **Créer une VM**
On met la VM dans la pool **template**



Et on le lance sous l'image Debian 12 qu'on vient d'installer sur Proxmox



On s'assure que la configuration réseau est bonne et crée la VM. Une fois la VM créé on entre dedans et on configure l'installation de Debian 12.
Maintenant que notre machine virtuelle est créé nous allons rentrer dedans, installer les clé SSH, mettre l'IP 10.31.208.2 et installer :

```
apt update
apt upgrade
apt install iptables tcpdump net-tools vim nano inetutils-ping sudo less
cron wget logrotate netcat-traditional ntpdate dnsutils traceroute nmap
rsyslog
```

Pour terminer, nous allons changer la VM et le conteneur en template. Pour se faire on éteint nos machines, puis clique droit → **convertir en modèle** :