RAPPORT DE PROJET SAE 5.01 : INFRASTRUCTURE DE VIRTUALISATION ET D'ACCÈS DISTANT (VDI)

Auteurs : Pierre FAMCHON
Formation : R&T - 3ème Année
Année : 2025-2026

SOMMAIRE DÉTAILLÉ

I. INTRODUCTION ET CONTEXTE

1.1. Objectifs du projet

Ce projet vise à concevoir et déployer une infrastructure de type VDI (Virtual Desktop Infrastructure) complète. L'objectif est de permettre aux étudiants et enseignants d'accéder à des environnements de Travaux Pratiques (Linux, Windows, Kali) à la demande, depuis n'importe quel navigateur web, sans installation de client lourd.

1.2. Choix techniques

- **Hyperviseur : Proxmox VE :**

  Choisi pour sa licence Open Source, sa gestion native des conteneurs LXC et KVM, et surtout pour son API REST complète qui facilitera l'automatisation.

- **Pare-feu : pfSense :**

  Solution robuste basée sur FreeBSD, permettant une gestion fine du NAT, du DHCP et des règles de filtrage (ACL).

- **Accès Distant : Apache Guacamole :**

  Clientless remote desktop gateway. Il supporte les protocoles standards (RDP, SSH, VNC) et les convertit en HTML5.

- **Développement : Python (Flask) :**

  Langage retenu pour le développement du portail web d'automatisation pour sa rapidité de mise en œuvre et ses bibliothèques de gestion de requêtes HTTP.

1.3. Architecture globale

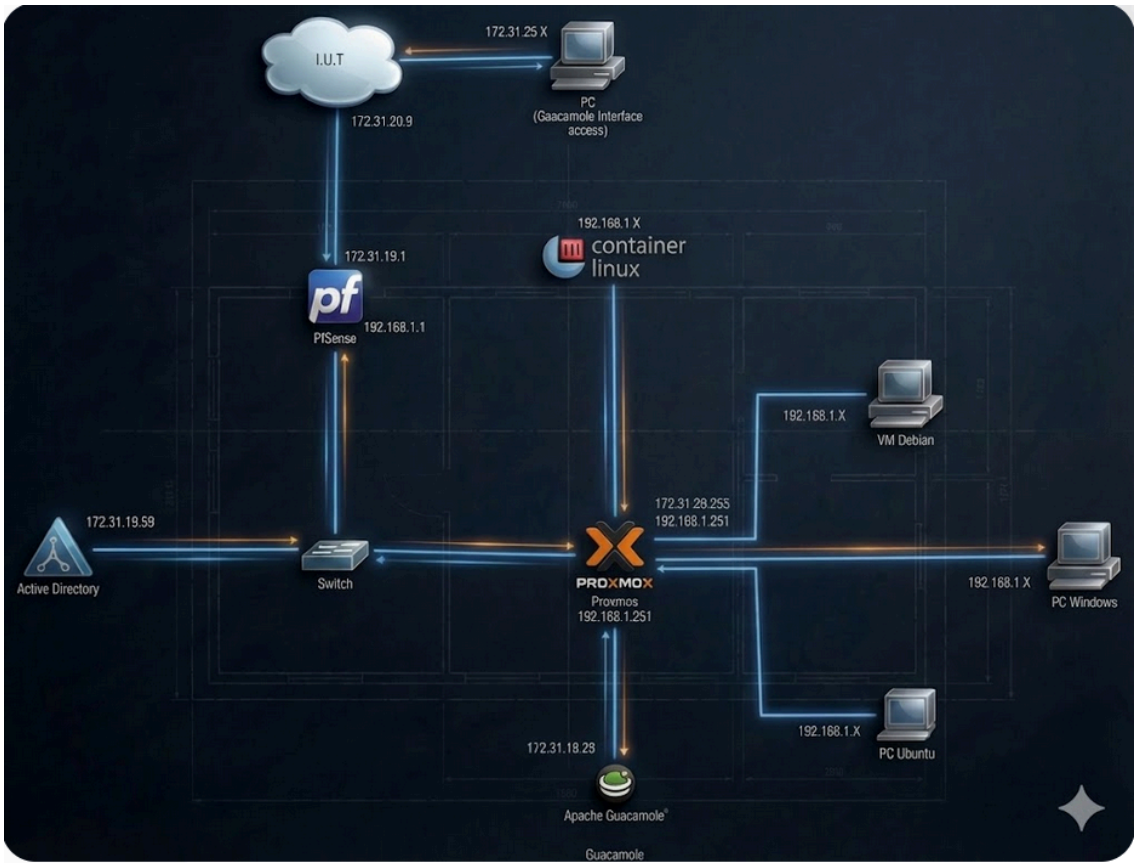
L'infrastructure repose sur un serveur physique hébergeant un hyperviseur. Pour garantir la sécurité et l'isolation, nous avons opté pour une architecture réseau segmentée :

- **Zone Publique (WAN) :**

  Connectée au réseau de l'IUT (172.31.xx.xx).

- **Zone Privée (LAN) :**

  Réseau interne (192.168.1.0/24) hébergeant les VMs et services critiques,



inaccessible directement depuis l'extérieur.

- **Passerelle :**

  Un routeur virtuel assure la liaison et le filtrage entre ces zones.

1.4. Planification et répartition des tâches

Tâche	Statut	Responsable
Phase 1 : Installation/config de l'hyperviseur Proxmox	Terminé	Nicolas Édouard piekyohann2005...
Phase 2 : Installation et configuration de Pfsense	En cours	Nicolas Édouard
Phase 3 : Installation et configuration du Windows Serveur	Terminé	Pierre Famchon
Phase 4 : Installation et configuration de Guacamole	Terminé	Pierre Famchon
Phase 5 : Portail Web et Automatisation	Terminé	Pierre Famchon piekyohann2005...
Phase 6 : Création et configuration des VM	Terminé	Nicolas Édouard
Phase 7 : Environnement et Conclusion	Terminé	piekyohann2005...
Rédaction des documents	En cours	Pierre Famchon

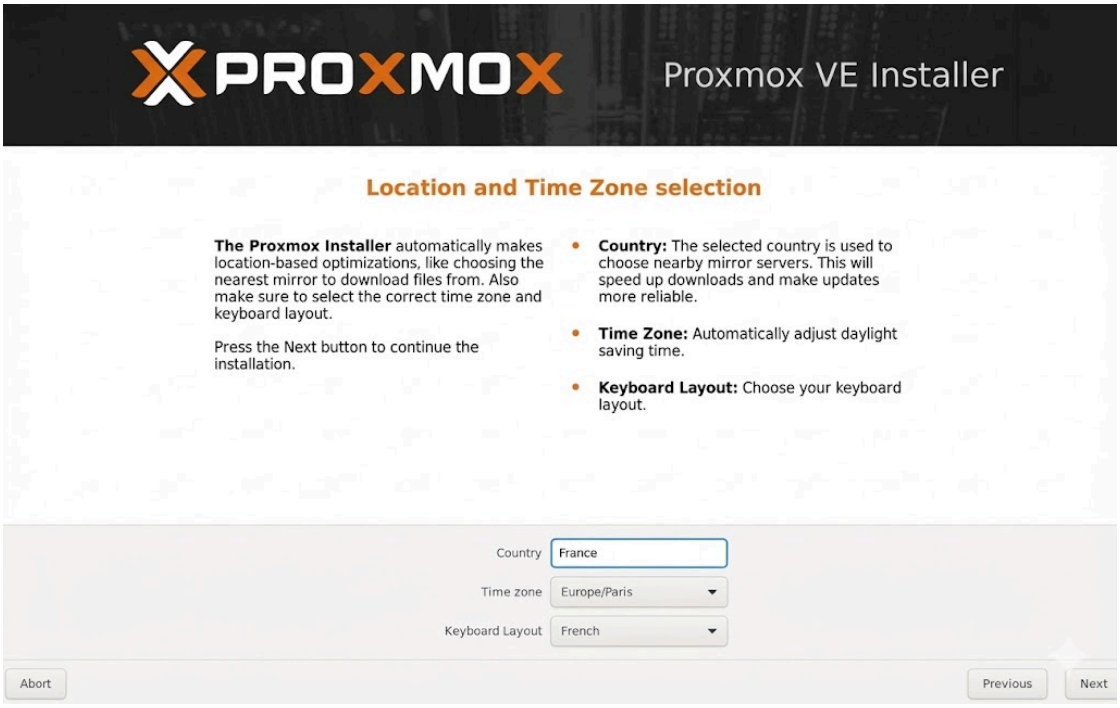
II. PHASE 1 : DÉPLOIEMENT DE L'HYPERVISEUR (PROXMOX VE)

2.1. Installation du système

L'installation a été réalisée sur le serveur physique attribué. Nous avons utilisé l'ISO officielle de Proxmox VE 8.0.

- **Paramètres régionaux :**

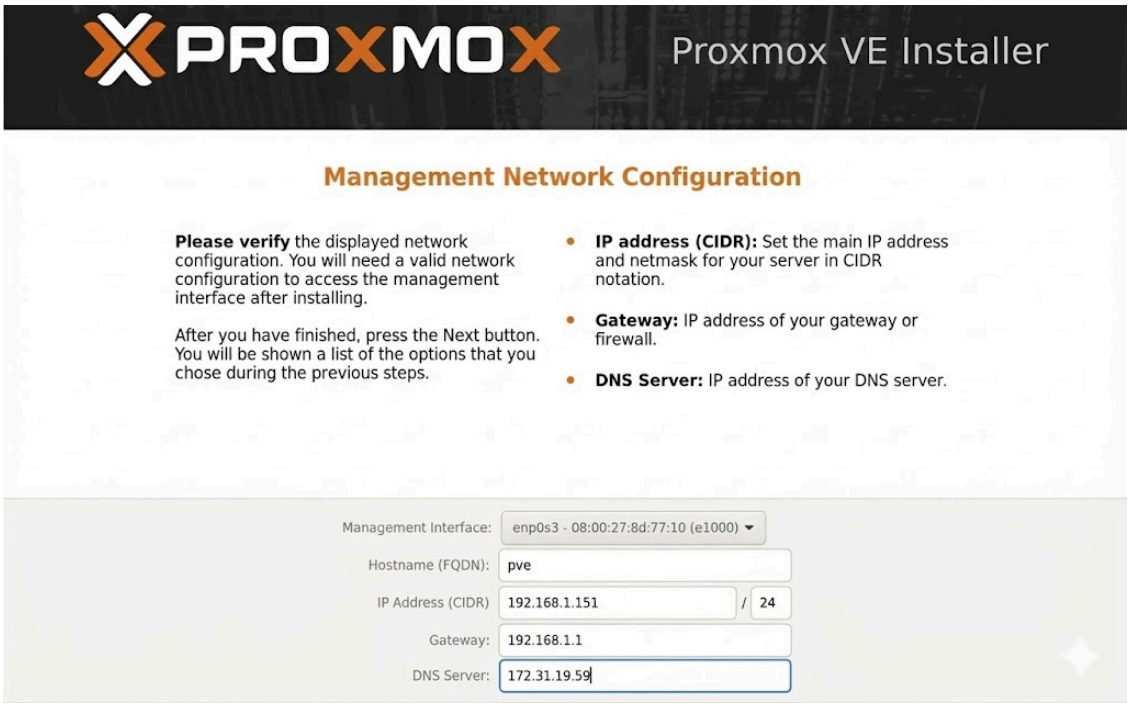
  France / Europe-Paris.

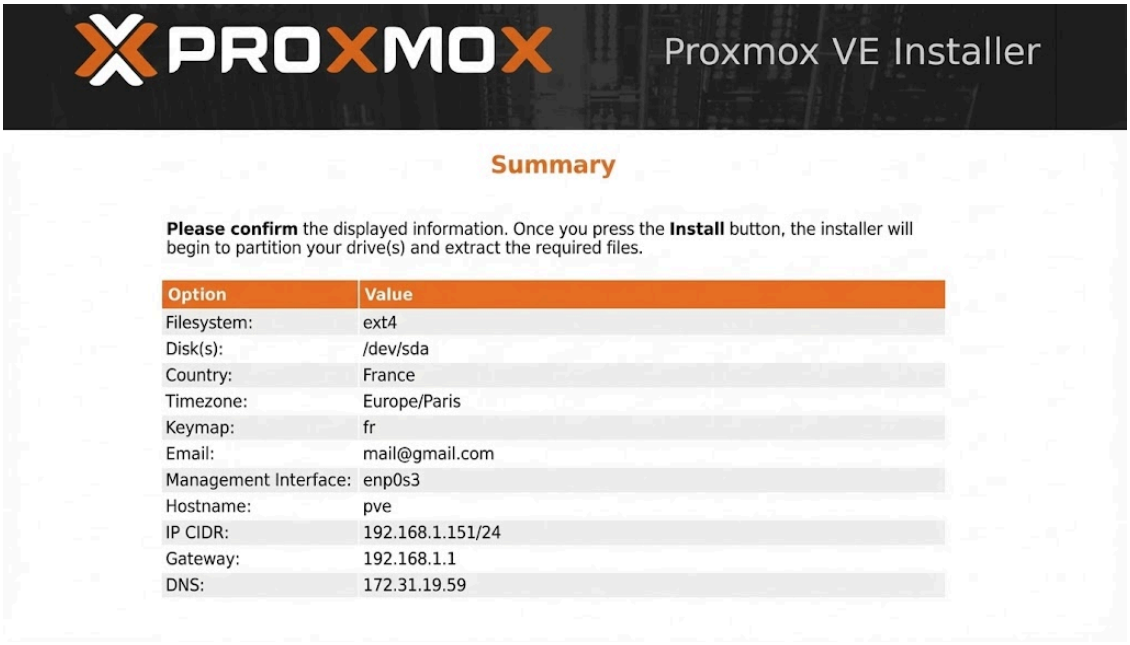


- **Partitionnement :**

  Utilisation du disque entier avec gestion LVM pour une flexibilité sur l'extension des partitions.

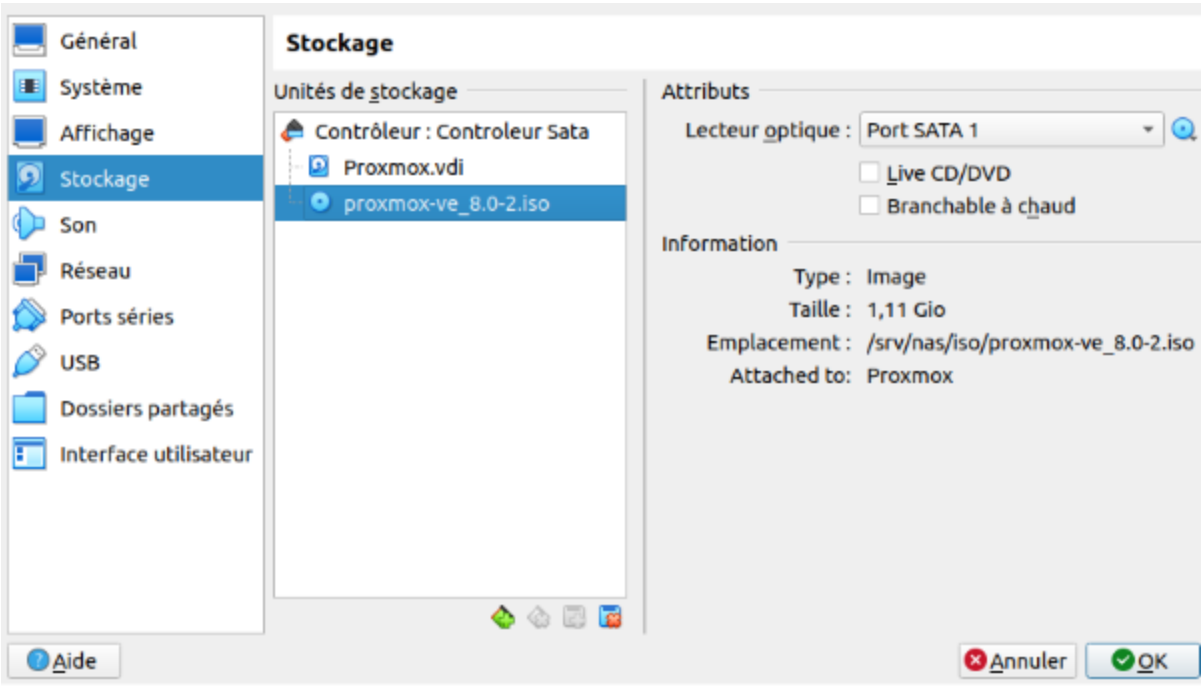
- **Configuration IP de gestion :**
  - IP : 192.168.1.151
  - Passerelle : 192.168.1.1
  - DNS : 172.31.19.59 (Anticipation du futur AD)





On observe le résumé de la configuration globale.

Une fois l'installation terminée, ne pas oublier de retirer l'ISO, pour pouvoir lancer notre VM Proxmox et éviter de booter en boucle sur l'ISO d'installation :



2.2. Configuration réseau

Pour isoler les VMs étudiantes, nous ne pouvions pas utiliser le pont par défaut vmbr0 qui est relié à la carte physique et au réseau public.

Action réalisée : Création d'un pont Linux (vmbr0) isolé.

- **Interface Web :**

  Système > Réseau > Créer > Linux Bridge.

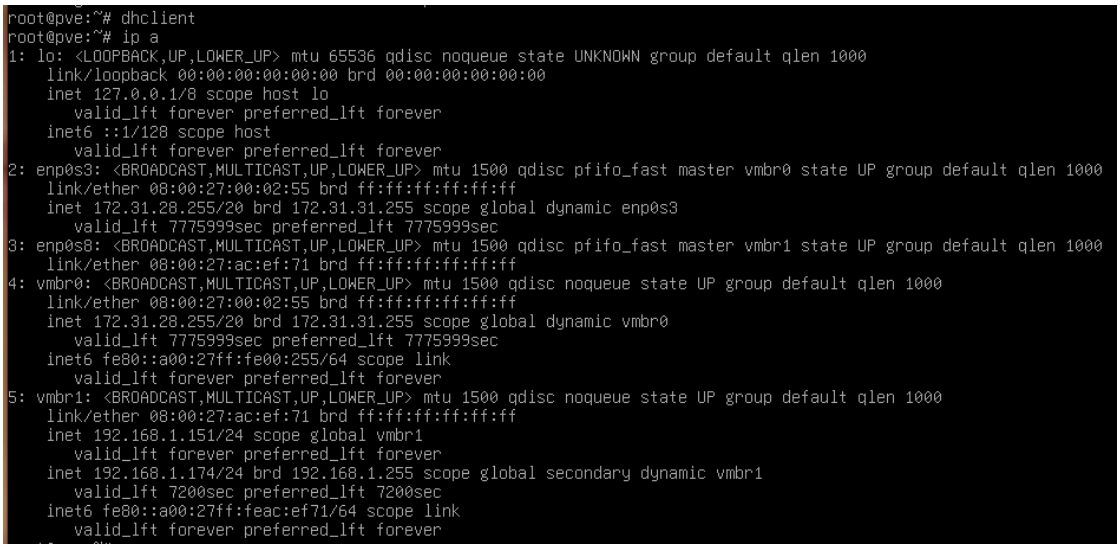
- **Configuration :**

  Aucune IP assignée, aucun port physique lié. Il agit comme un switch virtuel interne.

```
root@pve:~# nano /etc/network/interfaces
```

```
 GNU nano 7.2                                  /etc/network/interfaces
auto lo
iface lo inet loopback

iface enp0s8 inet manual

iface enp0s3 inet manual

auto vmbr0
iface vmbr0 inet static
        bridge-ports enp0s3
        bridge-stp off
        bridge-fd 0

auto vmbr1
iface vmbr1 inet static
        address 192.168.1.151/24
        gateway 192.168.1.1
        bridge-ports enp0s8
        bridge-ports none
        bridge-stp off
        bridge-fd 0
```

Comme on peut voir, une fois que nous effectuons la commande dhclient on peut voir que les adresses qui ont été attribué dans le document /etc/network/interfaces sont mises en places :



2.3. Configuration du Proxy

Le réseau de l'université nécessitant un proxy pour l'accès Internet (mises à jour système), nous avons configuré apt et les variables d'environnement.

- Fichier /etc/apt/apt.conf.d/70proxy : `Acquire::http::proxy "http://cache-etu.univ-artois.fr:3128";`

- Fichier .bashrc pour wget/curl : `export http_proxy=http://cache-etu.univ-artois.fr:3128`

```
root@pve:~# export http_proxy=cache-etu.univ-artois.fr:3128
root@pve:~# export https_proxy=cache-etu.univ-artois.fr:3128
```

- Fichier cat /etc/resolv.conf pour DNS :
  search dom-famchon.rt.lan
  nameserver 172.31.19.59

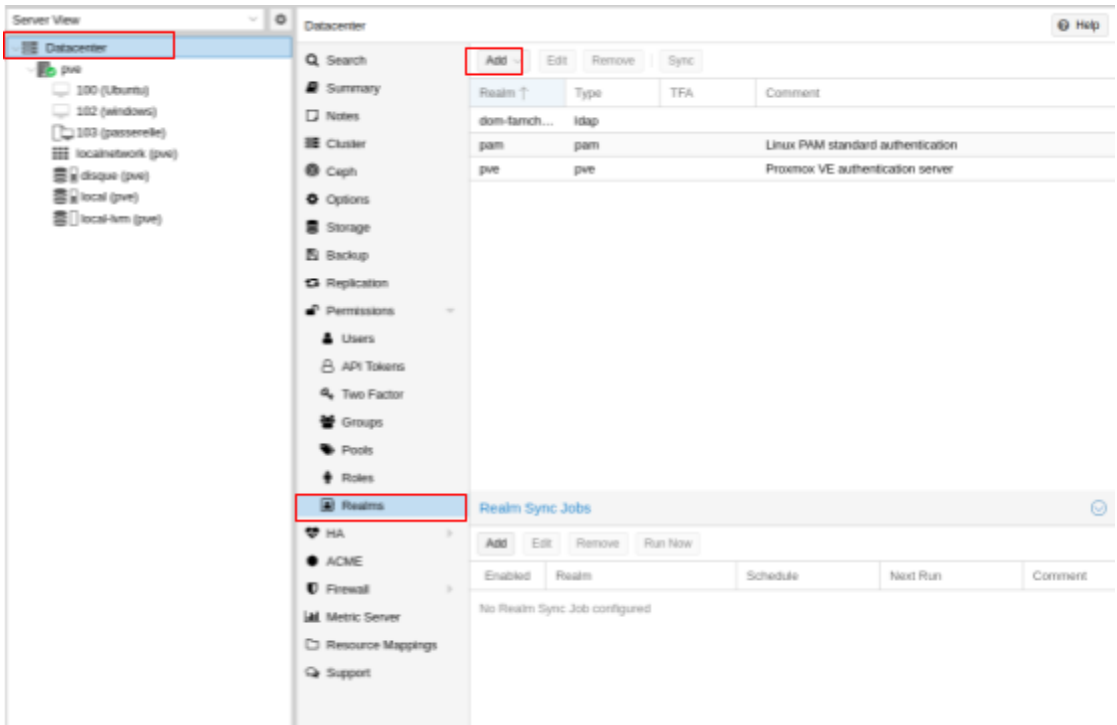
```
root@pve:~# cat /etc/resolv.conf
domain dom-famchon.rt.lan
search dom-famchon.rt.lan
nameserver 172.31.19.59
```

Nous avons modifié les DNS qui sont présents dans la machine pour utiliser le DNS que nous allons configurer sur l'AD.

2.4 Liaison AD et Proxmox

Pour la liaison entre notre AD et le serveur Proxmox, il faut se rendre sur l'interface web de Proxmox et suivre ces différentes étapes :

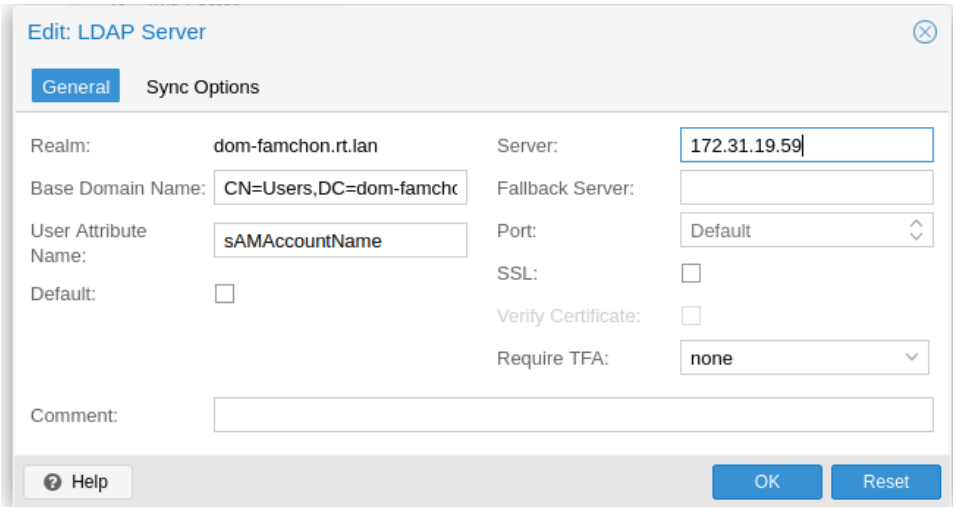
Tout d'abord, il faut se rendre dans le datacenter pour ajouter le serveur LDAP qui correspond à notre domaine.



Dans la création du serveur LDAP voici la configuration qu'il faut suivre :

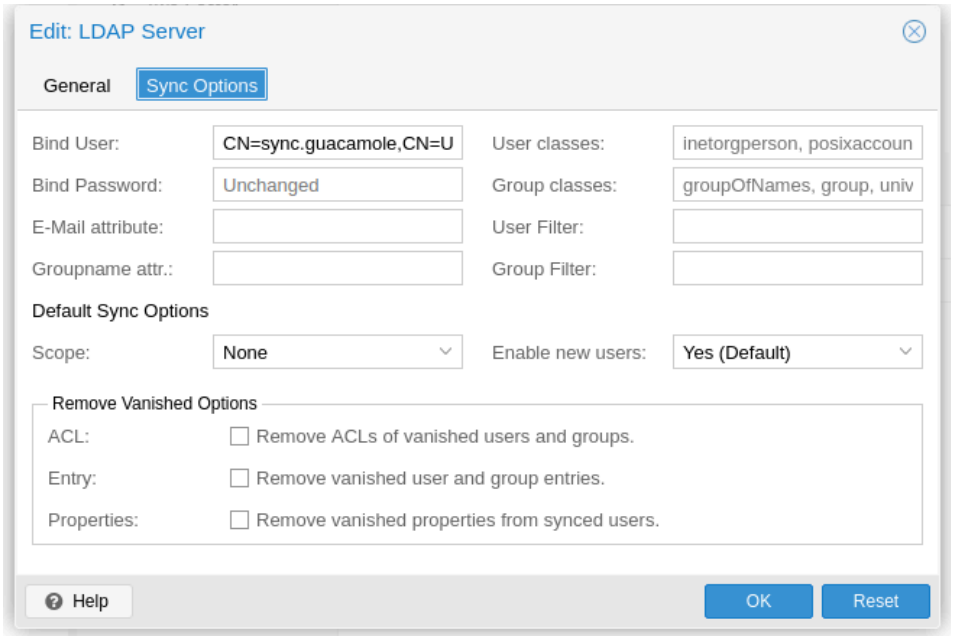
Dans la case Base Domain Name il faut rentrer ces informations pour que Proxmox cherchent les Utilisateurs et Groupes dans l'arborescence de notre annuaire LDAP/AD :

CN=Users,DC=dom-famchon,DC=rt,DC=lan

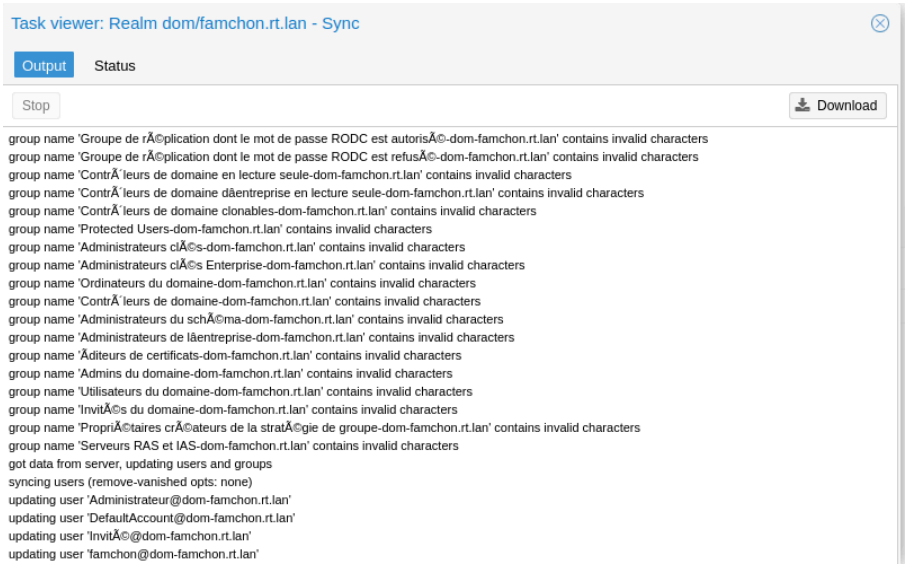


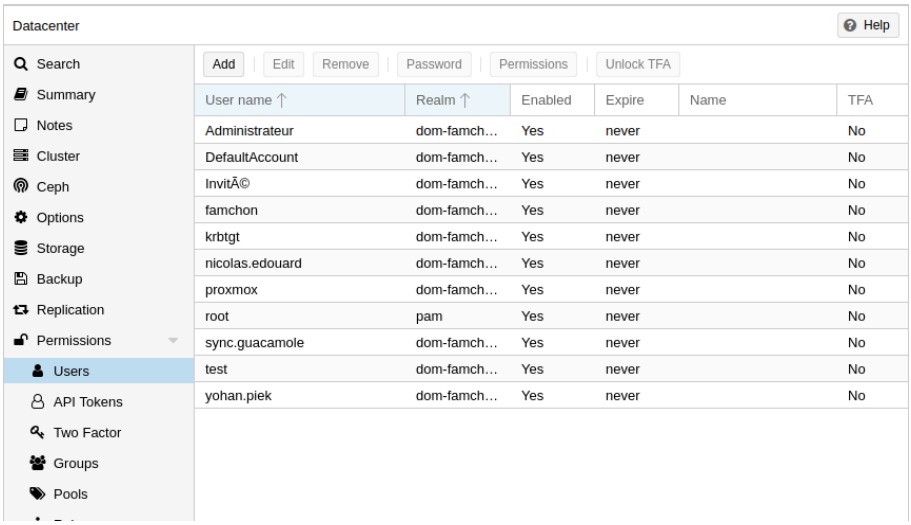
Dans la case Bind user on a besoin de rentrer ces informations pour que Proxmox se connecte avec un compte de l'AD pour se connecter à l'annuaire :

CN=sync.guacamole,CN=Users,Dc=dom-famchon,DC=rt,DC=lan



Comme on peut voir que Proxmox a récupéré tous les utilisateurs ainsi que les groupes qui sont présents dans notre AD.





III. PHASE 2 : SÉCURISATION ET ROUTAGE (PFSENSE)

C'est la pièce maîtresse de la sécurité du projet. pfSense agit comme la porte d'entrée et de sortie unique pour toutes les VMs.

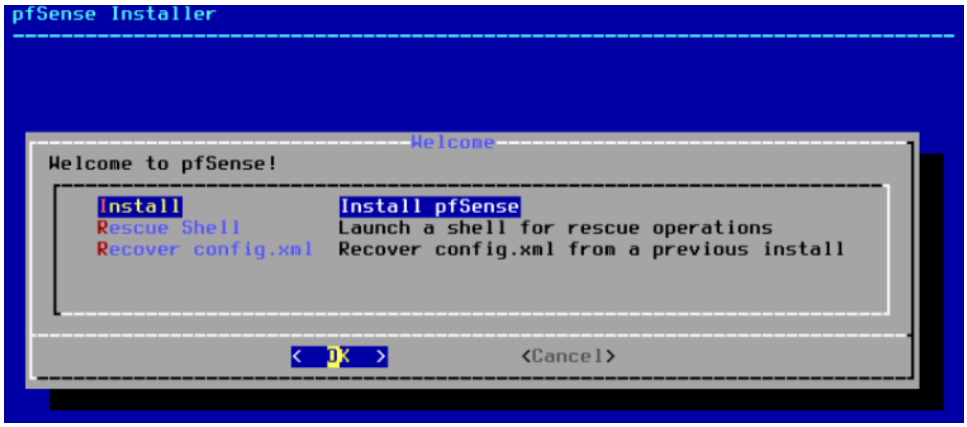
3.1. Installation de la VM

- **Ressources :**

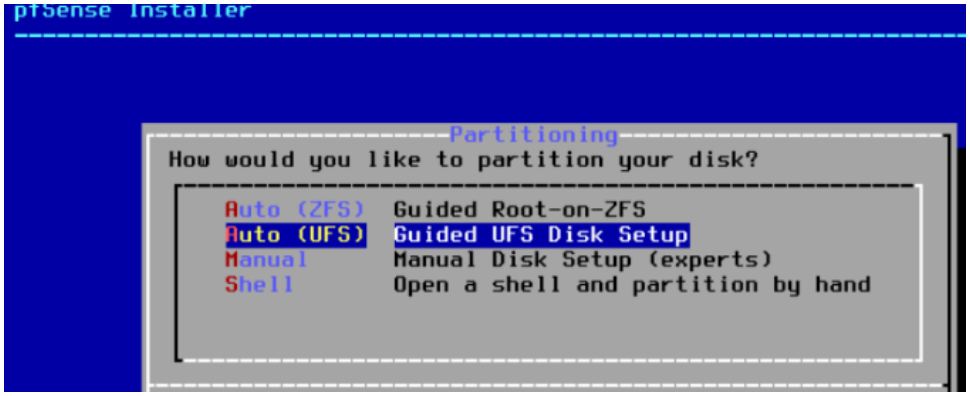
  1 vCPU, 1 Go RAM, 10 Go Disque.

- **Interfaces Réseaux :**
  1. net0 (WAN) → Liée au bridge vmbr0 (Accès Internet).
  2. net1 (LAN) → Liée au bridge vmbr1 (Réseau Privé).

On choisit **Install** puis **OK** pour lancer l'installation :



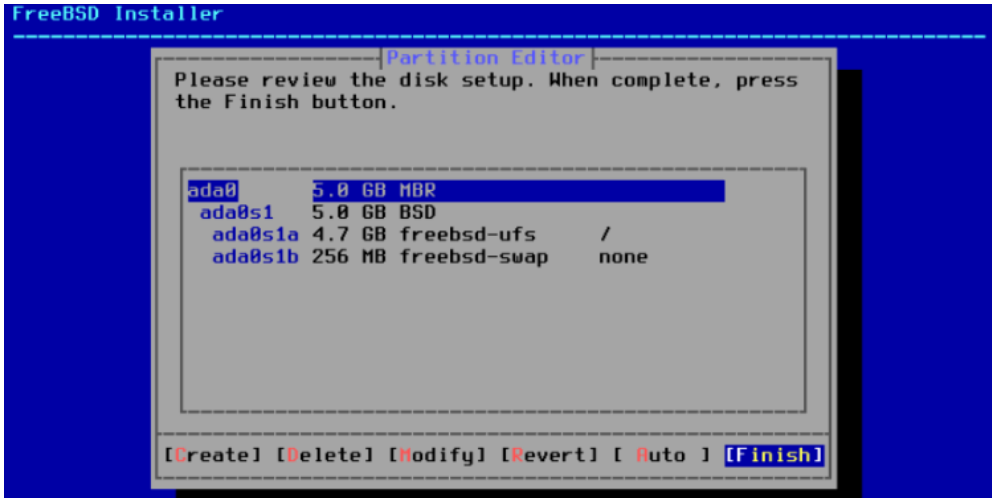
On partitionne notre disque en mode **Auto (UFS)** :



On choisit **Entire Disk** :

On choisit comme schéma de partition de volume **MBR DOS** :

Puis on choisit **ada0** pour terminer le **disk setup** :



Ensuite pour finir on fait un **commit** et il ne faut pas redémarrer la machine dans cet état il faut enlever l'ISO Pfsense.

3.2. Configuration des Interfaces

Lors du premier démarrage, via la console, nous avons assigné les interfaces :

- **WAN (vtnet0) :**

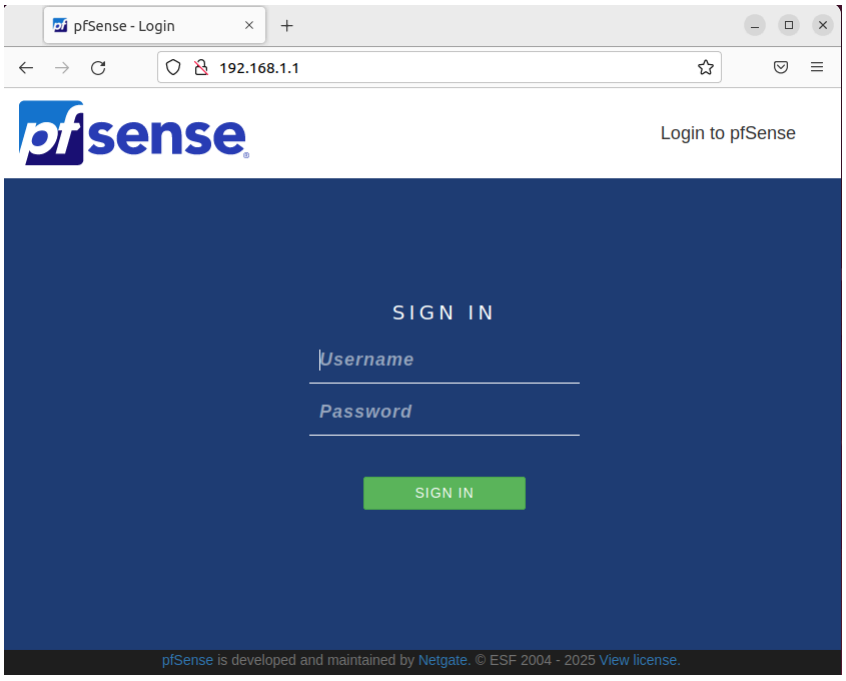
  Configuration en DHCP (Reçoit une IP en 172.31.x.x).

- **LAN (vtnet1) :**

  Configuration statique en **192.168.1.1 / 24**.

```
WAN (wan)       -> em0        -> v4/DHCP4: 172.31.19.1/20
LAN (lan)       -> em1        -> v4: 192.168.1.1/24
```

On lance notre machine cliente Ubuntu qui est sur le réseaux TP (Eth1) et on se connecte au portail PfSense en y mettant l'adresse IP **192.168.1.1 (LAN)** :



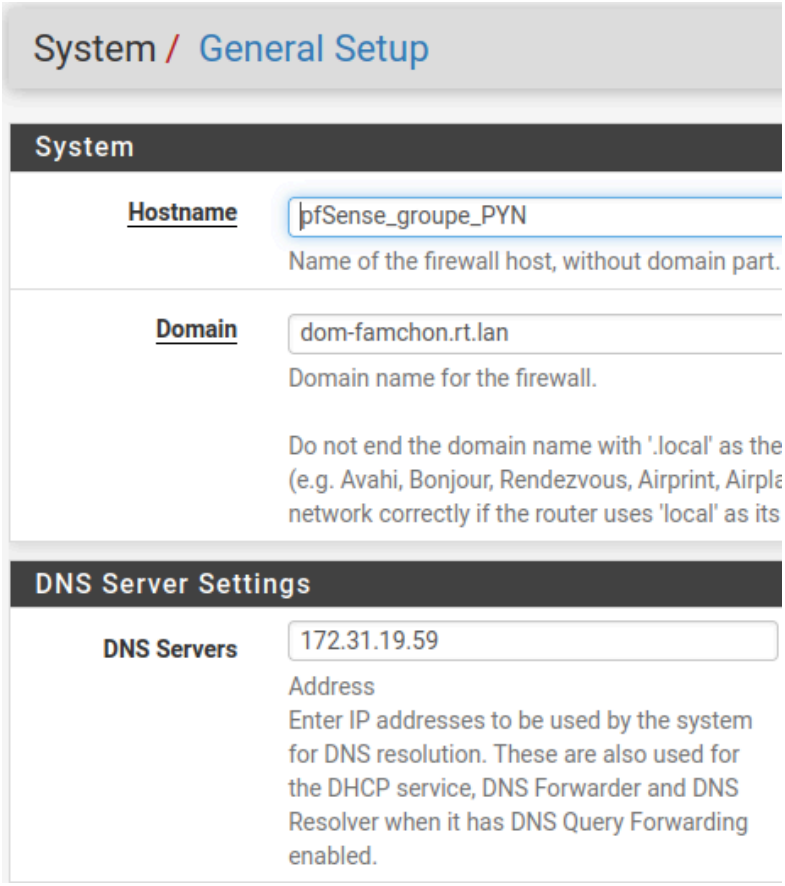
Les identifiants par défaut de pfsense :

- Login : admin
- MDP : pfsense

3.3. Configuration général

Une fois sur le portail captif, on se dirige dans **System** puis **General Setup** et on spécifie :

- Le Hostname : pfSense_groupe_PYN
- Le nom de domaine : dom-famchon.rt.lan
- L'IP du DNS Servers qui est notre serveur Windows (AD) avec l'IP 172.31.19.59.



3.4. Configuration affichage du portail Web

Ici, on y retrouve d'autres paramètres comme la langue d'affichage, la timezone et tout ce qui concerne la configuration et l'affichage du portail web :

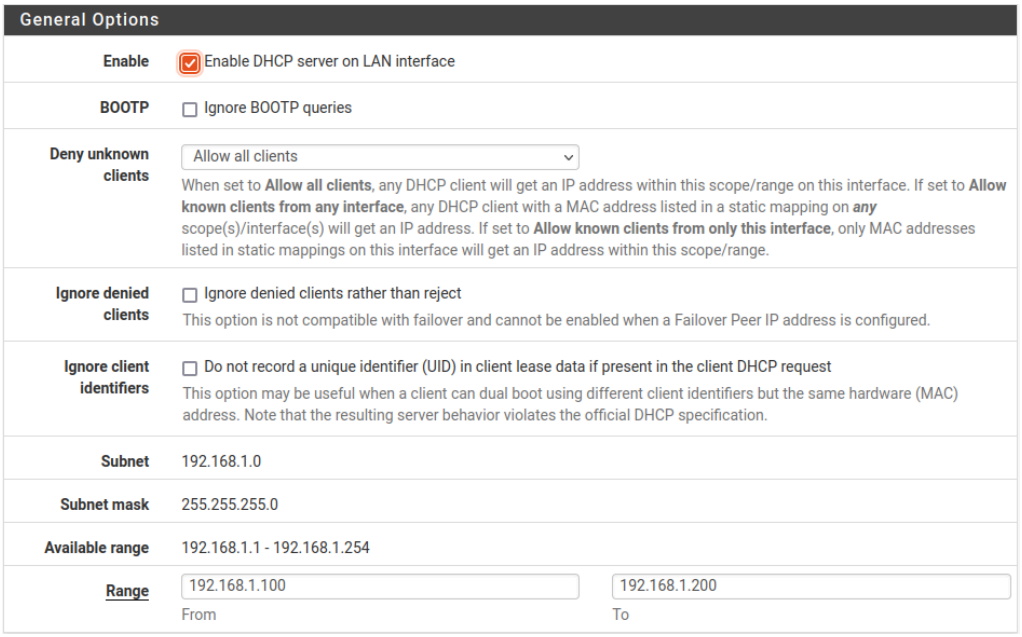
3.5. Services : DHCP et NAT

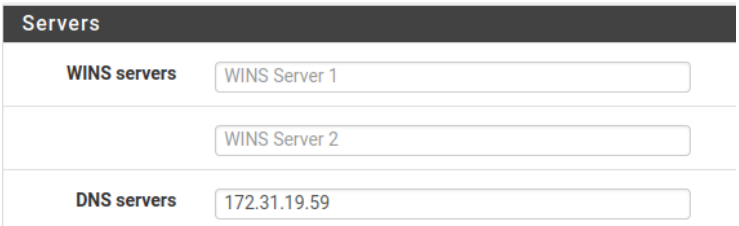
Pour que les VMs puissent communiquer sans configuration manuelle IP :

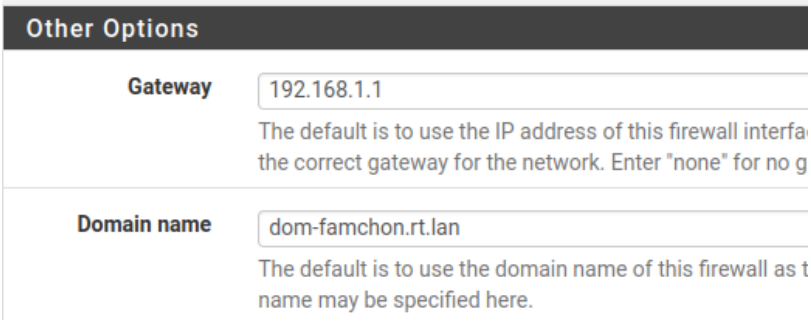
1. **Serveur DHCP :**

   Activé sur l'interface LAN.

   - Plage : 192.168.1.100 à 192.168.1.200.
   - DNS distribué : 192.168.1.254 (IP de notre futur AD).
   - Passerelle distribuée : 192.168.1.1.

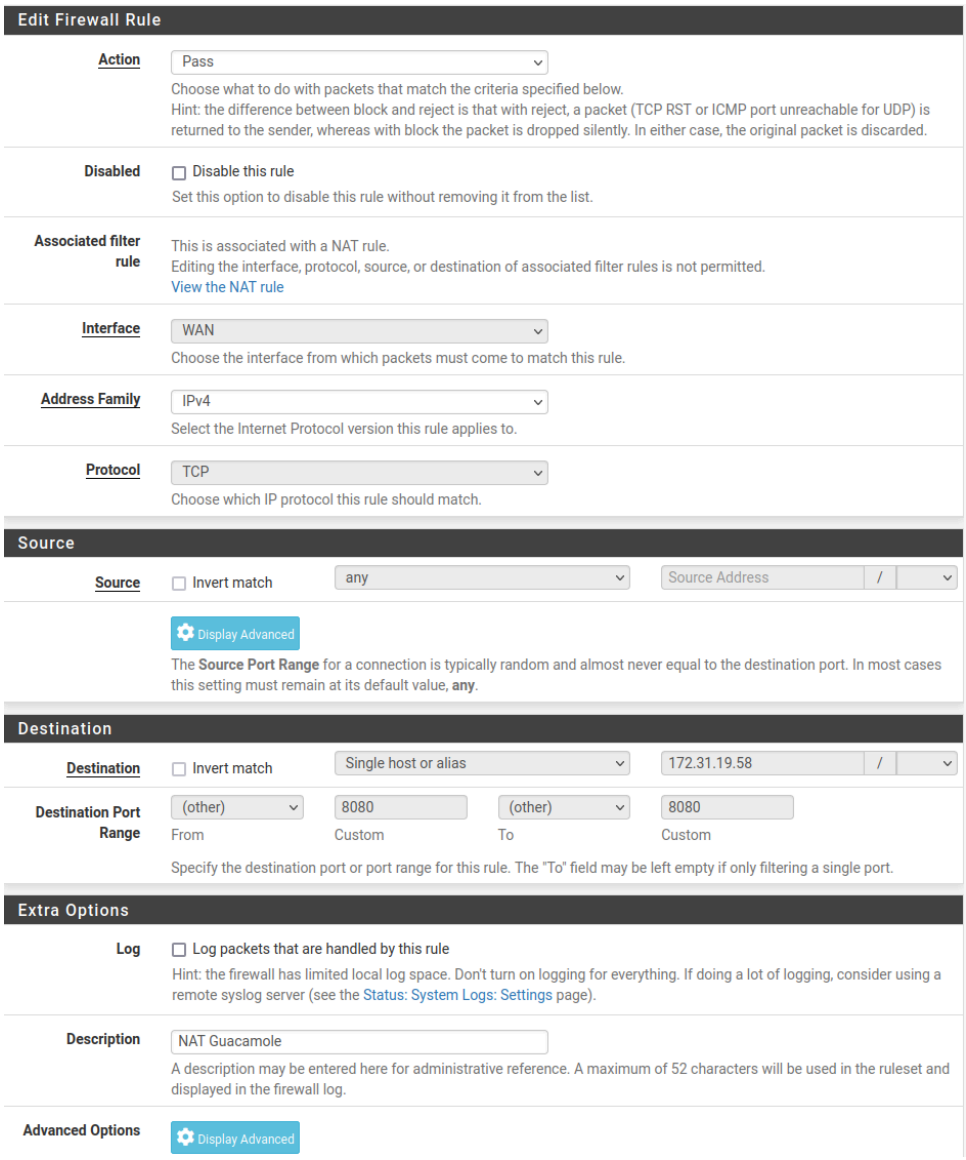


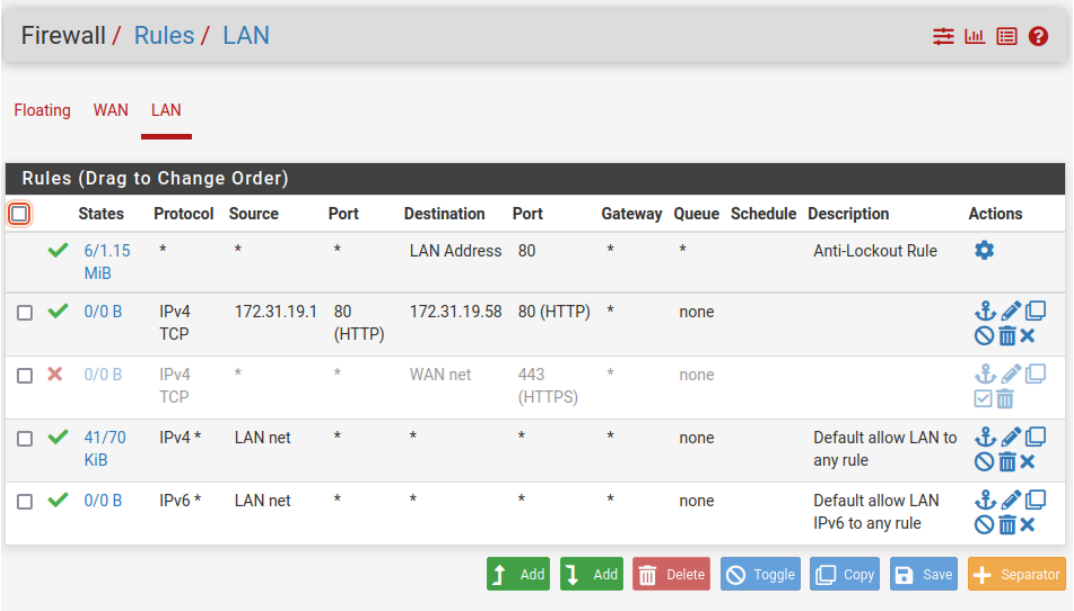




2. NAT Outbound :

Configuré en mode automatique pour permettre aux VMs du réseau 192.168.1.0/24 de sortir sur Internet en utilisant l'IP WAN du pfSense.





3.6. Règles de Pare-feu et Port Forwarding

Par défaut, pfSense bloque tout le trafic entrant.

1. **Règle LAN :**

   Création d'une règle "Allow Any to Any" pour permettre aux VMs de sortir.

Les 2 premières règles ADMIN_TO_PROX_HTTP/HTTPS et permettent d'autoriser uniquement le poste d'administration avec l'IP 192.168.1.10 à accéder à **Proxmox** ayant l'IP 192.168.1.251 aux ports 80 et 443 (HTTP et HTTPS).

LAN_TO_WAN_PF_HTTPS :
Cette 5ème règle permet de bloquer toutes les sources voulant accéder à l'extérieur (WAN) par le port 443 (HTTPS).

ADMIN_TO_PF_HTTPS :
Cette 6ème règle permet d'autoriser seulement le poste d'administration 192.168.1.10 à accéder au serveur **PfSense** et donc à l'interface web.

2. **Accès Guacamole (NAT Port Forward) :**
   - Nous avons redirigé le port **8080** de l'interface WAN vers l'IP interne de Guacamole (172.31.19.58).
   - Cela permet d'accéder à l'interface de gestion depuis une salle de TP via l'URL : http://172.31.19.58:8080/guacamole.

Edit Firewall Rule

Action — Pass

Choose what to do with packets that match the criteria specified below.
Hint: the difference between block and reject is that with reject, a packet (TCP RST or ICMP port unreachable for UDP) is returned to the sender, whereas with block the packet is dropped silently. In either case, the original packet is discarded.

Disabled — ☐ Disable this rule

Set this option to disable this rule without removing it from the list.

Interface — LAN

Choose the interface from which packets must come to match this rule.

Address Family — IPv4

Select the Internet Protocol version this rule applies to.

Protocol — TCP

Choose which IP protocol this rule should match.

Source

Source — ☐ Invert match | Single host or alias | 172.31.19.1 /

Hide Advanced

The **Source Port Range** for a connection is typically random and almost never equal to the destination port. In most cases this setting must remain at its default value, **any**.

Source Port Range

HTTP (80)		HTTP (80)	
From	Custom	To	Custom

Specify the source port or port range for this rule. The "To" field may be left empty if only filtering a single port.

Destination

Destination — ☐ Invert match | Single host or alias | 172.31.19.58 /

Destination Port Range

HTTP (80)		HTTP (80)	
From	Custom	To	Custom

Specify the destination port or port range for this rule. The "To" field may be left empty if only filtering a single port.

Extra Options

Log — ☐ Log packets that are handled by this rule

Hint: the firewall has limited local log space. Don't turn on logging for everything. If doing a lot of logging, consider using a remote syslog server (see the Status: System Logs: Settings page).

Description

A description may be entered here for administrative reference. A maximum of 52 characters will be used in the ruleset and displayed in the firewall log.

Advanced Options — Display Advanced

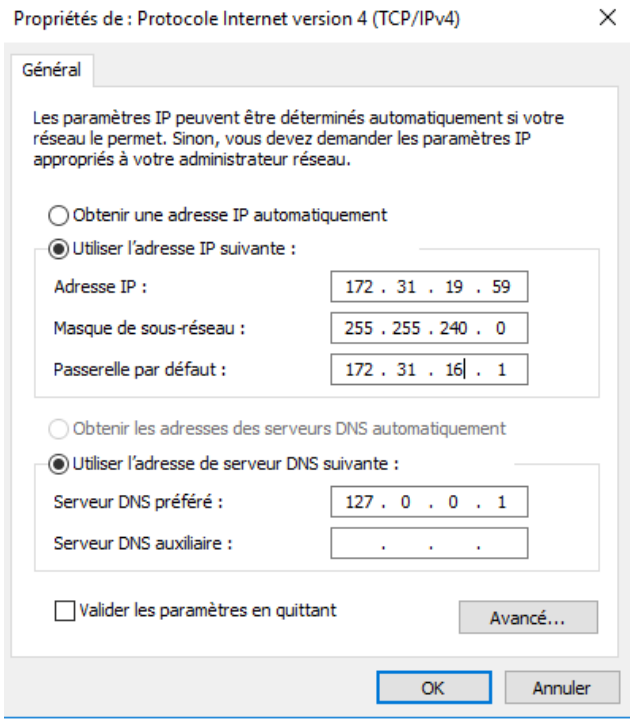
IV. PHASE 3 : SERVICES D'ANNUAIRE (WINDOWS SERVER AD)

4.1. Installation

Nous avons déployé une VM Windows Server 2016 pour centraliser la gestion des identités.

- **Nom de domaine :** dom-famchon.rt.lan.

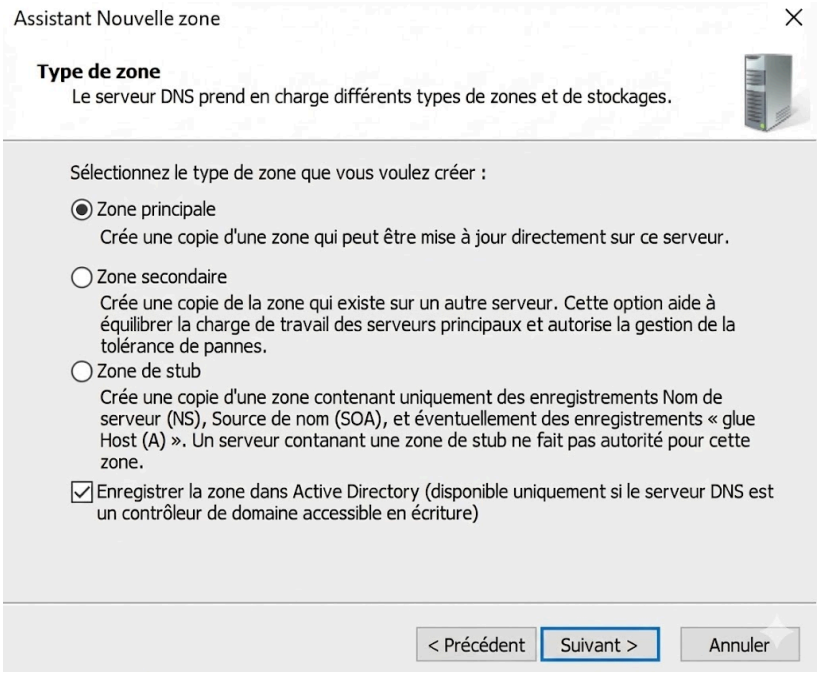
- **IP Statique :** 172.31.19.59.

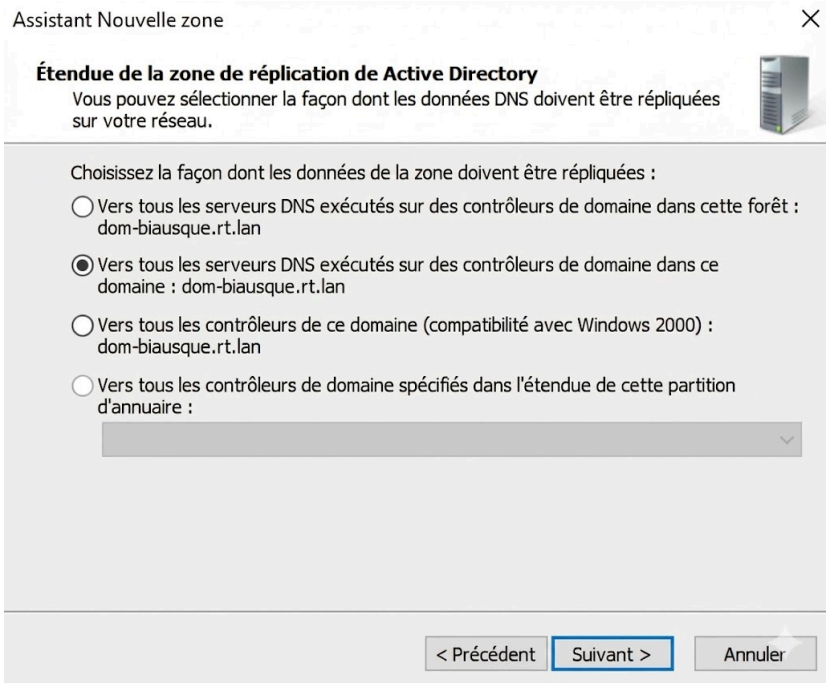


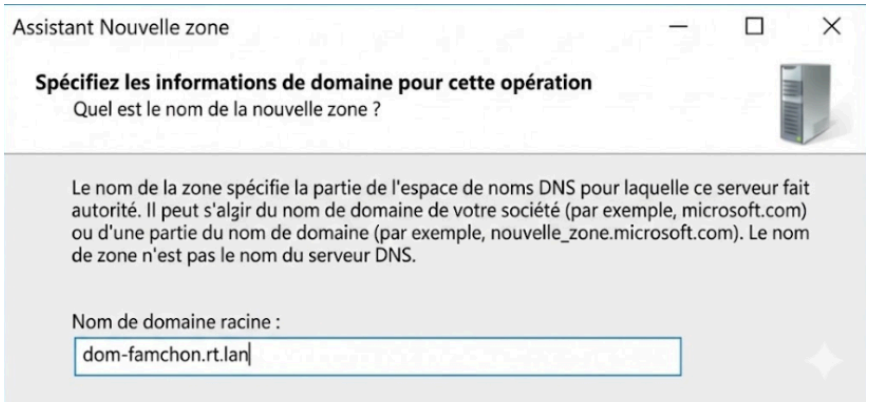
4.2. Configuration DNS (Zones Directes et Inversées)

Le DNS est vital pour notre script d'automatisation Python qui se base sur les noms de machines.

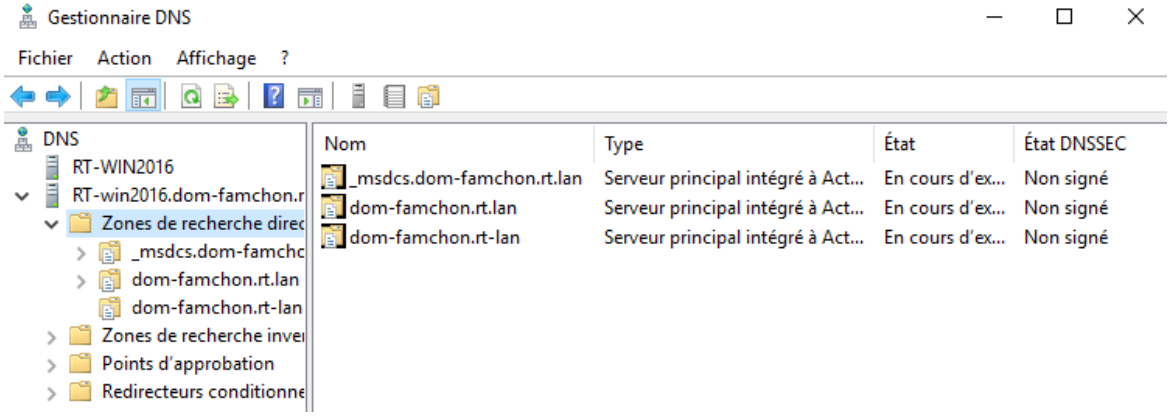
- **Zone Directe :** Créée automatiquement (résolution nom → IP).



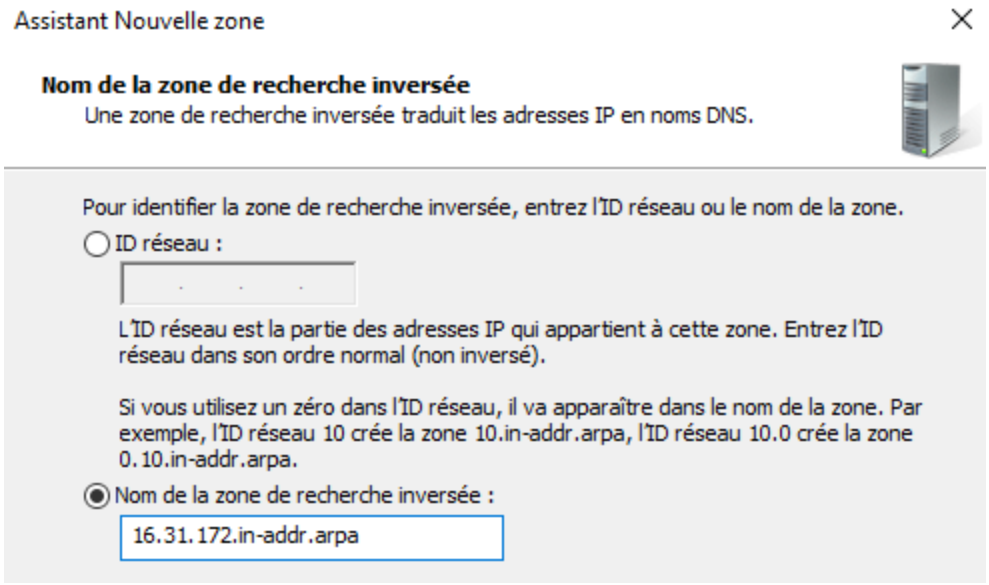




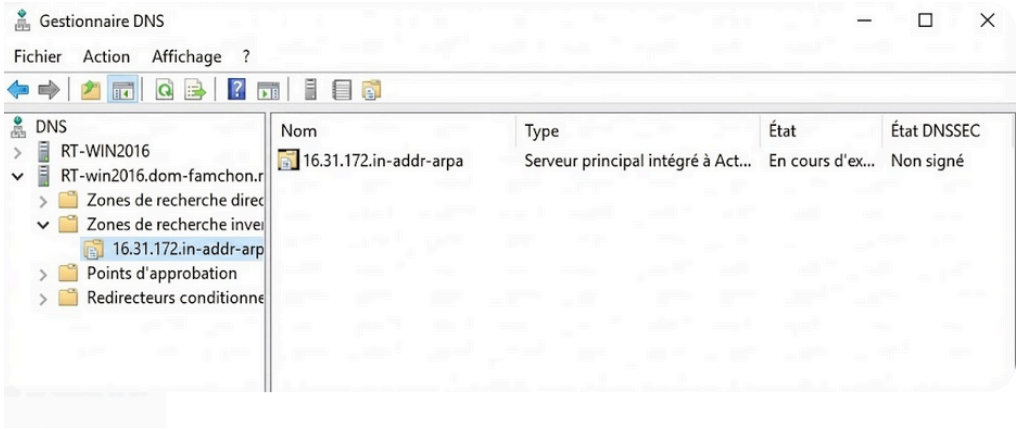
On constate la bonne création de la zones de recherche directe :



- **Zone Inversée :** Création manuelle de la zone pour le réseau 172.31.16.x. Cela permet la résolution IP → nom.

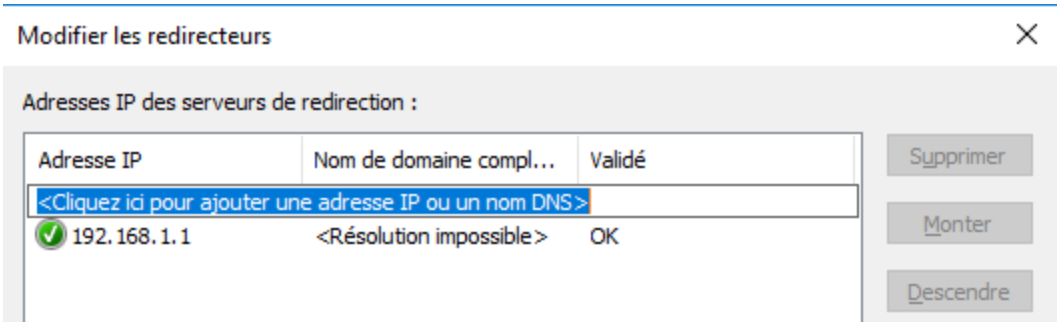


On constate la bonne création de la zone de recherche inversé :



- **Redirecteurs :**

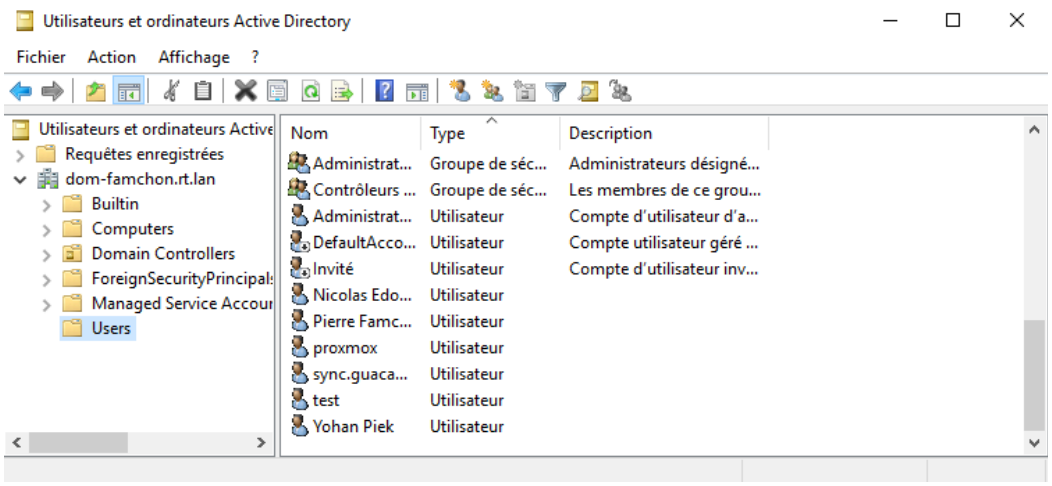
  Ajout de l'adresse de pfsense, qui joue le rôle de passerelle virtuel, permettant la résolution des noms internet pour les VMs du domaine.



4.3. Organisation

Nous avons structuré l'AD avec des Unités d'Organisation (OU) pour séparer les étudiants, les profs et les machines techniques (Proxmox, Guacamole).

- Utilisateurs de service créés : sync.guacamole, sync.proxmox (pour les liaisons LDAP).



V. PHASE 4 : PASSERELLE D'ACCÈS (APACHE GUACAMOLE)

5.1. Installation des prérequis et du serveur Guacamole

- **Environnement :**
  Machine Virtuelle Ubuntu.
- **Adresse IP :**
  172.31.19.58.
- **Composants clés :**
  Tomcat 9, Guacd (Proxy daemon), bibliothèques clientes (RDP/SSH).

Installation des dépendances de compilation :
Nous commençons par mettre à jour le système et installer les outils nécessaires à la compilation des sources de Guacamole (compilateur GCC, librairies graphiques, outils de développement).

Bash :

```
apt-get update
apt-get install build-essential libcairo2-dev libjpeg62-turbo-dev libpng-dev libtool-bin uuid-dev libossp-uuid-dev libavcodec-dev libavformat-dev libavutil-dev libswscale-dev freerdp2-dev libpango1.0-dev libssh2-1-dev libtelnet-dev libvncserver-dev libwebsockets-dev libpulse-dev libssl-dev libvorbis-dev libwebp-dev
```

Installation de Guacamole Server (guacd) :
Une fois les dépendances installées, nous téléchargeons, compilons et installons le cœur du système.

Bash :

```
cd /tmp
wget https://downloads.apache.org/guacamole/1.5.5/source/guacamole-server-1.5.5.tar.gz
tar -xzf guacamole-server-1.5.5.tar.gz
```

```
cd guacamole-server-1.5.5/


# Configuration et compilation
./configure --with-systemd-dir=/etc/systemd/system/
make
make install

# Mise à jour des liens dynamiques
ldconfig

# Activation du service
systemctl daemon-reload
systemctl enable --now guacd
```

Installation du Client Web (Tomcat 9) :
Pour l'interface web, nous utilisons Tomcat 9.

Bash :

```
apt-get install tomcat9 tomcat9-admin tomcat9-common tomcat9-user
# Déploiement du fichier .war
cd /var/lib/tomcat9/webapps/
wget https://downloads.apache.org/guacamole/1.5.5/binary/guacamole-1.5.5.war
-O guacamole.war
```

5.2. Authentification Hybride

Pour garantir sécurité et flexibilité, nous avons mis en place une double authentification :

- **MySQL (MariaDB) :**
  Utilisé pour stocker la configuration technique des connexions (quelle VM a quelle IP, quel protocole, paramètres d'affichage).
- **LDAP (Active Directory) :**
  Utilisé pour l'authentification des utilisateurs. Cela évite de recréer les comptes en double et permet aux étudiants d'utiliser leurs identifiants habituels.

Mise en place de la base de données :

Bash

```
apt-get install mariadb-server
mysql_secure_installation

# Création de la base
mysql -u root -p
CREATE DATABASE guacadb;
CREATE USER 'guaca'@'localhost' IDENTIFIED BY 'P@ssword!';
GRANT SELECT, INSERT, UPDATE, DELETE ON guacadb.* TO 'guaca'@'localhost';
FLUSH PRIVILEGES;
EXIT;
```

5.3. Configuration LDAP et Résolution de problème

Lors de l'intégration, nous avons rencontré des erreurs d'authentification.

- **Problème :**

  L'extension LDAP n'est pas fournie par défaut avec le paquet APT.

- **Solution :**

  Téléchargement manuel du fichier .jar (version 1.5.5) et placement dans /etc/guacamole/extensions/.

Bash

```
mkdir -p /etc/guacamole/extensions
cd /etc/guacamole/extensions
wget
https://downloads.apache.org/guacamole/1.5.5/binary/guacamole-auth-ldap-1.5.5.tar.gz
tar -xzf guacamole-auth-ldap-1.5.5.tar.gz
mv guacamole-auth-ldap-1.5.5/guacamole-auth-ldap-1.5.5.jar .
```

Fichier de configuration /etc/guacamole/guacamole.properties :
Nous avons configuré la liaison avec notre contrôleur de domaine (172.31.19.59).

- **Configuration LDAP (guacamole.properties) :**

  Properties

  ```
  ldap-hostname: 172.31.19.59
  ldap-user-base-dn: DC=dom-famchon,DC=rt,DC=lan
  ldap-username-attribute: sAMAccountName
  ldap-user-search-filter: (objectClass=user)
  ```

- **Configuration MySQL (Rappel) :**

  ```
  mysql-hostname: 127.0.0.1
  mysql-port: 3306
  mysql-database: guacadb
  ```

```
mysql-username: guaca
mysql-password: progtr00
```

Fichier de configuration du démon /etc/guacamole/guacd.conf :
Nous forçons l'écoute sur toutes les interfaces pour éviter les problèmes de liaison locale.

- **Configuration (guacd.conf) :**

```
[server]
blind_host = 0.0.0.0
blind_port = 4822
```

5.4. Finalisation et Tests

Nous enregistrons les configurations et redémarrons l'ensemble de la stack logicielle :

```
Bash

systemctl restart tomcat9 guacd mariadb
```

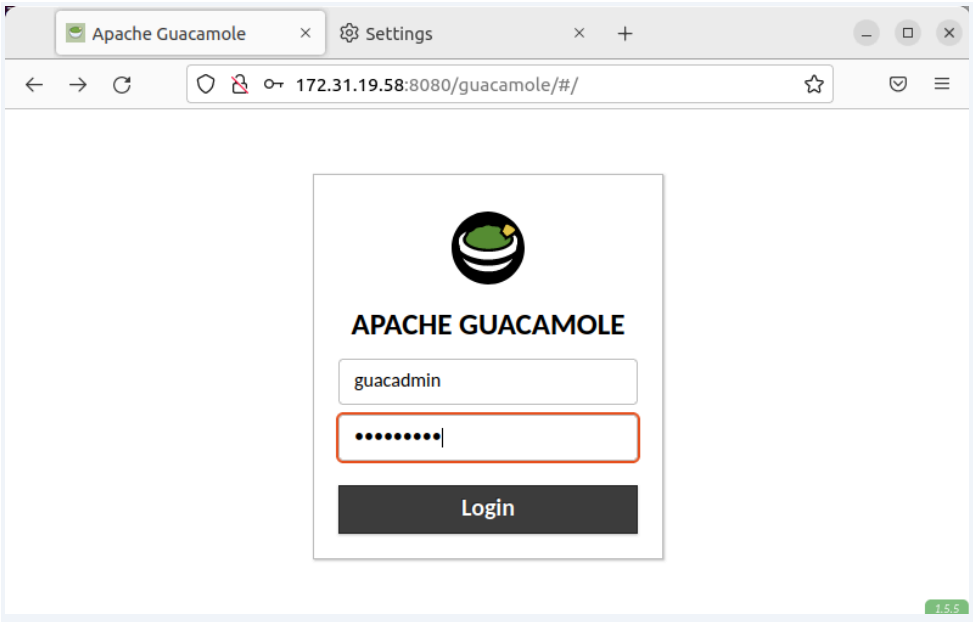
Accès à l'interface :

L'application est désormais accessible via l'URL :

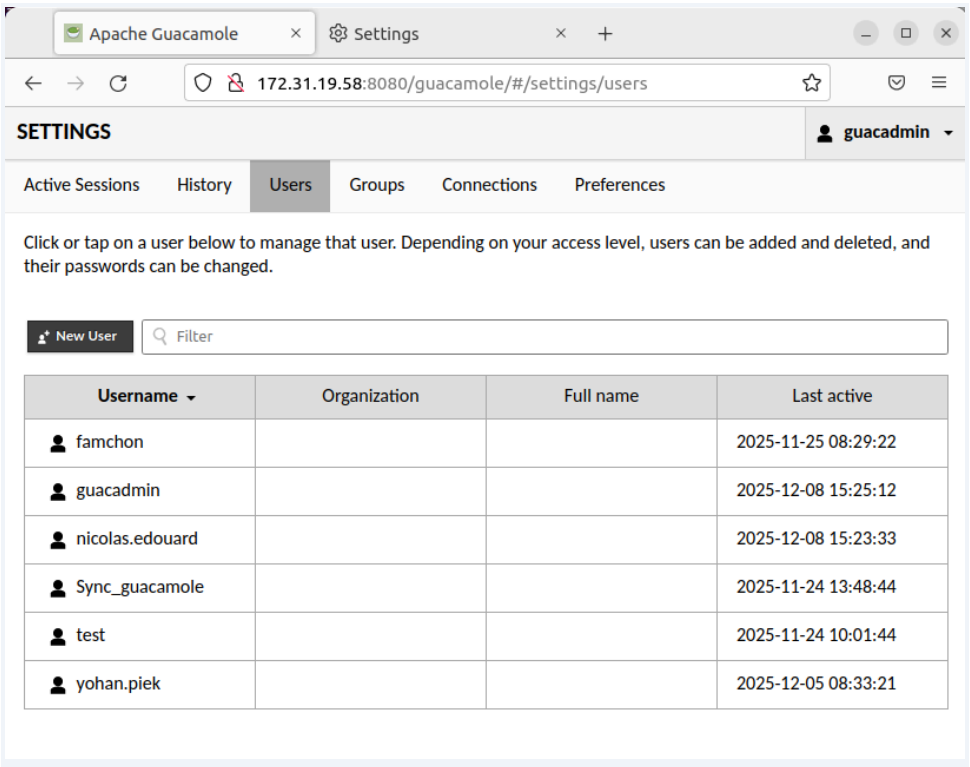
http://172.31.19.58:8080/guacamole

Nous utilisons le compte administrateur local par défaut pour la première configuration :

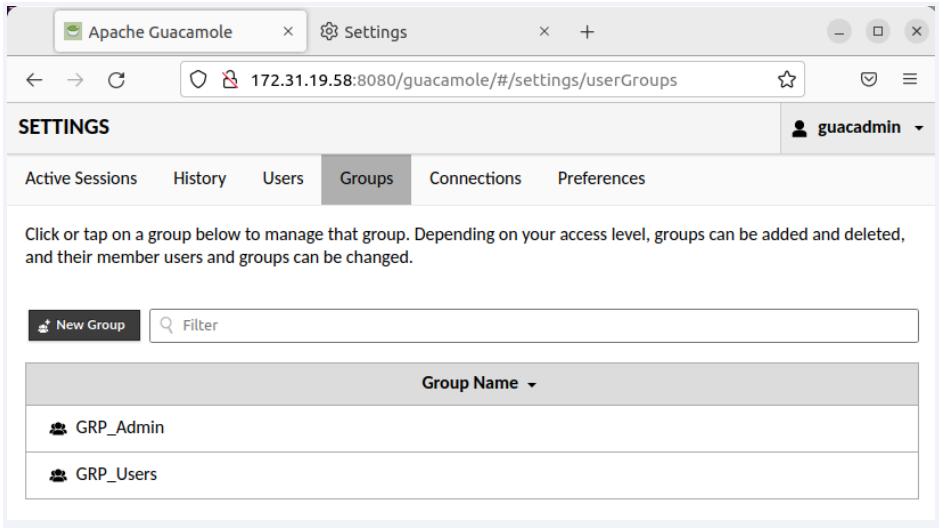
- **Utilisateur :**
  guacadmin
- **Mot de passe :**
  guacadmin



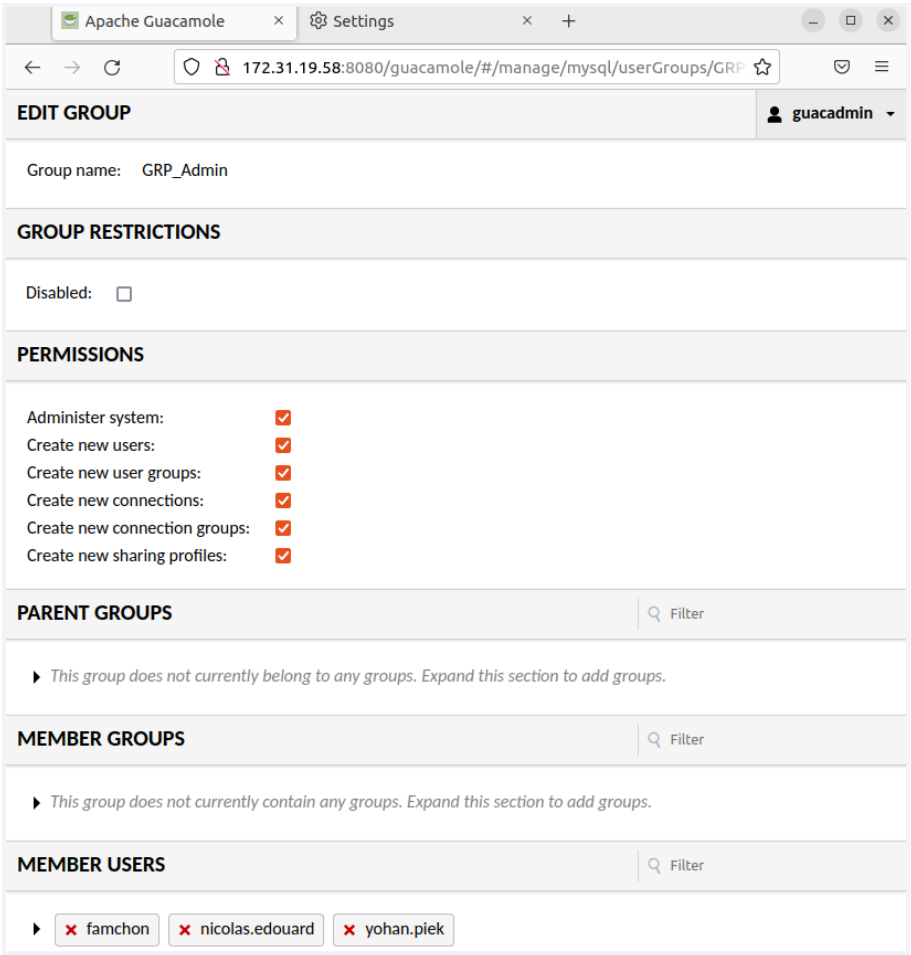
Portail de connexion Guacamole.



La liaison avec l'AD fonctionne, nous pouvons observer nos Users.



Nous pouvons observer, nos groupes.



Les permissions associés à ces groupes ainsi que les users qui le composent.

VI. PHASE 5 : AUTOMATISATION ET PORTAIL WEB (PYTHON/FLASK)

6.1. Architecture et Environnement

L'application est développée en Python 3 en utilisant le micro-framework Flask, choisi pour sa légèreté et sa facilité de déploiement.
Préparation de l'environnement sur le serveur :
Nous avons isolé les dépendances du projet dans un environnement virtuel (venv) pour ne pas polluer le système hôte.

Bash

```
# Installation des paquets systèmes
sudo apt update
sudo apt install python3-pip python3-venv

# Création de l'environnement virtuel et installation des librairies
mkdir mon_portail_vm
cd mon_portail_vm
python3 -m venv venv
source venv/bin/activate
pip install flask requests
```

Structure du projet :

```
mon_portail_vm/
├── app.py            ← Le cerveau (Ton ancien script bash traduit en Python)
├── config.py         ← Tes mots de passe et Tokens (Sécurisé)
└── templates/
    ├── login.html    ← Page de connexion
    └── dashboard.html ← Liste des VMs et boutons
```

- app.py : Le cœur de l'application (Logique métier, Routes).
- config.py : Fichier contenant les secrets (Tokens API, URLs, Mots de passe) pour ne pas les coder en dur.
- templates/ : Dossier contenant les pages HTML (login.html, dashboard.html).

6.2. Logique Backend : Intégration des API (Proxmox/Guacamole)

Récupération de l'API Guacamole :

```
curl -X POST \
  -d "username=nicolas.edouard" \
  -d "password=Progtr00#" \
  "http://172.31.19.58:8080/guacamole/api/tokens"
```

Récupération de l'API Proxmox :

Générer le Token (Interface Proxmox)

1. On se connecte à Proxmox (https://172.31.28.255:8006) en root.
2. On va dans Datacenter (colonne de gauche tout en haut).
3. Puis dans Permissions > API Tokens.
4. Clique sur Add (Ajouter).
    - User : root@pam
    - Token ID : guacamole
    - Privilege Separation : Décoche-le (ce sera plus simple pour commencer, ça donne au token les mêmes droits que root).
5. Et pour finir on clique sur Add.

Vérifier les Permissions

Même si on a décoché "Privilege Separation", on vérifie que le token a le droit d'agir.

1. On va dans Datacenter > Permissions.
2. On clique sur Add > API Token Permission.
3. API Token : Choisis root@pam!guacamole.
4. Path : / (La racine).
5. Rôle : Administrator (ou PVEVMAdmin si tu veux être plus restrictif, mais Administrator évite les bloquages pour l'instant).
6. Ensuite on clique sur Add.

Le portail agit comme un chef d'orchestre entre l'utilisateur, Proxmox et Guacamole.

Authentification via Guacamole :

Le portail ne gère pas les mots de passe. Il les transmet à l'API de Guacamole. Si Guacamole (relié à l'AD) valide, il renvoie un token.

Python

```
def get_guacamole_token(username, password):
    url = f"{config.GUACAMOLE_URL}/tokens"
    data = {"username": username, "password": password}
    # Appel à l'API Guacamole
    resp = requests.post(url, data=data, proxies=NO_PROXY)
    if resp.status_code == 200:
        return resp.json().get("authToken")
    return None
```

Gestion des VMs via Proxmox :
Nous utilisons un Token API (root@pam!guacamole) avec des droits administrateurs limités pour piloter l'hyperviseur.

- **Listing :**
  Récupération de la liste des VMs JSON et filtrage par nom d'utilisateur.
- **Clonage :**
  Détection automatique du prochain ID libre (max_id + 1) et clonage du template choisi (Windows ou Linux).

6.3 Contrainte Réseau : Le Défi du Proxy

Durant le développement, nous avons rencontré un blocage majeur : le script Python, bien que hébergé sur le même réseau que Guacamole, passait par le proxy de l'université pour tenter de joindre localhost ou l'IP locale, causant des erreurs 403/503.

Solution implémentée :

Nous avons injecté un dictionnaire de configuration spécifique dans les requêtes requests pour forcer le script à ignorer le proxy système pour les appels API internes.

Python

```
# Configuration pour forcer le trafic local (Bypass Proxy)
NO_PROXY = {
   "http": None,
   "https": None,
}

# Utilisation dans chaque appel API
requests.post(url, data=data, proxies=NO_PROXY)
```

6.4. Configuration du Script Python (`app.py` & `config.py`)

APP.PY :

```
from flask import Flask, render_template, request, redirect, session, url_for
import requests
import config  # Assure-toi que config.py est bien rempli !
import json
import time
import urllib3

# On désactive les avertissements de sécurité SSL pour Proxmox (car certificats auto-signés)
urllib3.disable_warnings(urllib3.exceptions.InsecureRequestWarning)

app = Flask(__name__)
app.secret_key = "SUPER_SECRET_KEY_A_CHANGER"  # Nécessaire pour gérer les sessions utilisateurs

# --- CONFIGURATION PROXY (LE FIX IMPORTANT) ---
# On définit un dictionnaire vide pour forcer requests à NE PAS utiliser de proxy
NO_PROXY = {
   "http": None,
   "https": None,
}

# --- FONCTIONS API ---

def get_guacamole_token(username, password):
   """Authentifie l'utilisateur sur Guacamole et récupère son token"""
   url = f"{config.GUACAMOLE_URL}/tokens"
   data = {"username": username, "password": password}

   print(f"\n--- [DEBUG] AUTHENTIFICATION ---")
   print(f"Vers URL: {url}")
   print(f"User: {username}")

   try:
      # On ajoute proxies=NO_PROXY pour éviter l'erreur "Proxy URL had no scheme"
      resp = requests.post(url, data=data, proxies=NO_PROXY)

      print(f"Code Retour: {resp.status_code}")
```

```
        if resp.status_code == 200:
            token = resp.json().get("authToken")
            print(f"Token reçu: {token[:10]}...") # On affiche le début du token
            return token
        else:
            print(f"Erreur Auth: {resp.text}")

    except Exception as e:
        print(f"CRASH RESEAU (Auth): {e}")
    return None

def get_admin_guac_token():
    """Récupère un token ADMIN pour créer des connexions (utilise le compte de
service)"""
    return get_guacamole_token(config.GUAC_ADMIN_USER, config.GUAC_ADMIN_PASS)

def proxmox_get_vms(username_prefix):
    """Lister les VMs Proxmox qui commencent par le nom de l'utilisateur"""
    url = f"{config.PROXMOX_URL}/api2/json/nodes/{config.PROXMOX_NODE}/qemu"
    headers = {"Authorization": config.PROXMOX_TOKEN}

    print(f"\n--- [DEBUG] LISTING PROXMOX ---")

    try:
        # verify=False pour le SSL, proxies=NO_PROXY pour le réseau
        resp = requests.get(url, headers=headers, verify=False, proxies=NO_PROXY)

        if resp.status_code == 200:
            data = resp.json().get('data', [])
            # Filtrage : On garde uniquement les VMs qui commencent par "nicolas.edouard-"
            user_vms = [vm for vm in data if vm.get('name', '').startswith(username_prefix)]
            return user_vms
        else:
            print(f"Erreur Proxmox: {resp.status_code} - {resp.text}")

    except Exception as e:
        print(f"CRASH RESEAU (Proxmox): {e}")
    return []

def create_guacamole_connection(vm_name, vm_ip, rdp_user, rdp_pass):
    """Crée la connexion dans Guacamole via l'API (nécessite token admin)"""
    admin_token = get_admin_guac_token()
```

```
    if not admin_token:
        print("Impossible d'obtenir le token admin Guacamole")
        return False

    url =
f"{config.GUACAMOLE_URL}/session/data/mysql/connections?token={admin_token}"

    # Configuration de la connexion RDP
    payload = {
        "name": f"{vm_name}-rdp",
        "protocol": "rdp",
        "parentIdentifier": "ROOT",
        "parameters": {
            "hostname": vm_ip,
            "username": rdp_user,
            "password": rdp_pass,
            "security": "any",       # (Ou "nla" si tu as changé)
            "ignore-cert": "true",
            "port": "3389"
        }
    }
    try:
        resp = requests.post(url, json=payload, proxies=NO_PROXY)
        if resp.status_code == 200:
            print(f"Connexion Guacamole créée pour {vm_name}")
            return True
        else:
            print(f"Erreur Création Guacamole: {resp.text}")
    except Exception as e:
        print(f"Crash Création Guacamole: {e}")
    return False

def create_vm_logic(user_prefix, template_id, vm_name_suffix, rdp_user, rdp_pass):
    """Logique optimisée : Utilise le DNS au lieu de l'agent QEMU"""

    # 1. Trouver un ID libre
    url_list = f"{config.PROXMOX_URL}/api2/json/nodes/{config.PROXMOX_NODE}/qemu"
    headers = {"Authorization": config.PROXMOX_TOKEN}

    try:
        vms = requests.get(url_list, headers=headers, verify=False,
proxies=NO_PROXY).json()['data']
        # On récupère les IDs existants pour prendre le suivant
```

```
        ids = [int(vm['vmid']) for vm in vms if str(vm.get('vmid')).isdigit()]
        new_id = max(ids) + 1 if ids else 150
    except:
        new_id = 150

    full_name = f"{user_prefix}-{vm_name_suffix}"
    print(f"Création de la VM {new_id} ({full_name})...")

    # 2. Cloner le template
    url_clone =
f"{config.PROXMOX_URL}/api2/json/nodes/{config.PROXMOX_NODE}/qemu/{template_id
}/clone"
    payload = {"newid": new_id, "name": full_name}
    requests.post(url_clone, headers=headers, data=payload, verify=False,
proxies=NO_PROXY)

    # 3. Démarrer la VM
    time.sleep(2)
    url_start =
f"{config.PROXMOX_URL}/api2/json/nodes/{config.PROXMOX_NODE}/qemu/{new_id}/sta
tus/start"
    requests.post(url_start, headers=headers, verify=False, proxies=NO_PROXY)

    print("VM démarrée. Configuration DNS en cours...")

    # 4. Construction du Nom de Domaine (FQDN)
    # C'est ici qu'on utilise ton DNS qui marche !
    DOMAINE = "dom-famchon.rt.lan"
    vm_fqdn = f"{full_name}.{DOMAINE}"

    print(f"Liaison Guacamole vers : {vm_fqdn}")

    # On crée la connexion Guacamole avec le NOM (vm_fqdn) au lieu de l'IP
    # Plus besoin d'attendre, c'est instantané !
    create_guacamole_connection(full_name, vm_fqdn, rdp_user, rdp_pass)

    return new_id

# --- ROUTES DU SITE WEB ---

@app.route('/', methods=['GET', 'POST'])
def login():
    error = None
```

```
    if request.method == 'POST':
        user = request.form['username']
        pwd = request.form['password']

        # Test authentification
        token = get_guacamole_token(user, pwd)

        if token:
            session['user'] = user
            session['guac_token'] = token
            return redirect(url_for('dashboard'))
        else:
            error = "Identifiants incorrects ou Erreur Guacamole (Voir Terminal)"

    return render_template('login.html', error=error)

@app.route('/dashboard')
def dashboard():
    if 'user' not in session:
        return redirect(url_for('login'))

    # Récupérer les VMs
    mes_vms = proxmox_get_vms(session['user'])
    return render_template('dashboard.html', vms=mes_vms, user=session['user'])

@app.route('/create_vm', methods=['POST'])
def create_vm():
    if 'user' not in session: return redirect(url_for('login'))

    nom_vm = request.form['vm_name']
    template_id = request.form['template_id']

    # On utilise les infos de session ou un formulaire étendu pour le user RDP
    # Pour l'instant on simplifie
    create_vm_logic(session['user'], template_id, nom_vm, "user", "password")

    return redirect(url_for('dashboard'))

@app.route('/logout')
def logout():
    session.clear()
    return redirect(url_for('login'))
```

```
if __name__ == '__main__':
    # On lance sur toutes les interfaces (0.0.0.0) port 5000
    app.run(host='0.0.0.0', port=5000, debug=True)
```

CONFIG.PY :

```
# config.py

# --- PROXMOX (Pour créer/gérer les VMs) ---
PROXMOX_URL = "https://172.31.28.255:8006"
PROXMOX_NODE = "pve"
# Ton token root Proxmox (celui de ton script)
PROXMOX_TOKEN =
"PVEAPIToken=root@pam!guacamole=fd1e9c7c-8128-4937-a8e2-3601a848c15e"
# --- GUACAMOLE (Pour l'admin et l'auth) ---
GUACAMOLE_URL = "http://172.31.19.58:8080/guacamole/api"
# Token admin pour créer les connexions dans Guacamole (si besoin)
GUAC_ADMIN_USER = "sync.guacamole"
GUAC_ADMIN_PASS = "Progtr00#"
```

6.5. Interface Utilisateur (Code HTML & Rendu Visuel)

login.html :

```
<!DOCTYPE html>
<html>
<head>
   <title>Login - Portail VM</title>
   <link rel="stylesheet" href="{{ url_for('static', filename='style.css') }}">
</head>
<body>
   <div class="container">
      <div class="card">
         <h2>Accès Sécurisé</h2>

         {% if error %}
            <p style="color: red; text-align: center;">{{ error }}</p>
         {% endif %}

         <form method="POST">
            <label>Utilisateur</label>
            <input type="text" name="username" placeholder="ex:
nicolas.edouard" required>

            <label>Mot de passe</label>
            <input type="password" name="password" required>

            <input type="submit" value="SE CONNECTER">
         </form>
      </div>
   </div>
</body>
</html>
```

dashboard.html :

```
<!DOCTYPE html>
<html>
<head>
   <title>Dashboard - Portail VM</title>
   <link rel="stylesheet" href="{{ url_for('static', filename='style.css') }}">
</head>
<body>
   <div class="container">
      <h1>Bienvenue, {{ user }}</h1>

      <div class="card">
         <h3>🚀 Créer une nouvelle machine</h3>
         <form action="/create_vm" method="POST">
            <label>Nom de la machine (Suffixe)</label>
            <input type="text" name="vm_name" placeholder="ex: Travail_TP1"
required>

            <label>Système d'exploitation</label>
            <select name="template_id">
               <option value="100">Windows 10 (Template 100)</option>
               <option value="103">Linux Ubuntu (Template 103)</option>
               </select>

            <input type="submit" value="Lancer la création">
         </form>
      </div>

      <div class="card">
         <h3>🖥️ Mes Machines Virtuelles</h3>
         {% if vms %}
            <ul>
               {% for vm in vms %}
                  <li>
                     <div>
                        <strong>{{ vm.name }}</strong><br>
                        <small style="color: #888;">ID: {{ vm.vmid }} - Statut: {{
vm.status }}</small>
                     </div>

                     <a href="http://172.31.19.59:8080/guacamole/" target="_blank">
                        <button>Accéder</button>
                     </a>
                  </li>
```

```
            {% endfor %}
          </ul>
        {% else %}
          <p style="text-align: center; color: #888;">Aucune machine virtuelle
active.</p>
        {% endif %}
      </div>

      <a href="/logout" class="logout-link">Se déconnecter</a>
  </div>
</body>
</html>
```

6.6. Workflow de création et Innovation DNS

Pour optimiser l'expérience utilisateur, nous avons amélioré le processus de création standard.

- **L'étudiant valide le formulaire :**
  Choix du nom de la VM et du Template (OS).
- **Clonage et Démarrage :**
  Le script ordonne à Proxmox de cloner le template et de démarrer la nouvelle VM.
- **Innovation (Prédiction DNS)**
  Au lieu d'attendre 30 à 60 secondes que l'agent QEMU remonte l'adresse IP de la VM (processus lent et parfois instable), nous utilisons notre infrastructure DNS/DHCP robuste.
  - Nous savons que la VM va s'appeler nom_vm.
  - Nous savons que le domaine est dom-famchon.rt.lan.
  - Le script **construit le FQDN** (nom_vm.dom-famchon.rt.lan) et configure Guacamole avec ce nom de domaine immédiatement.

Python

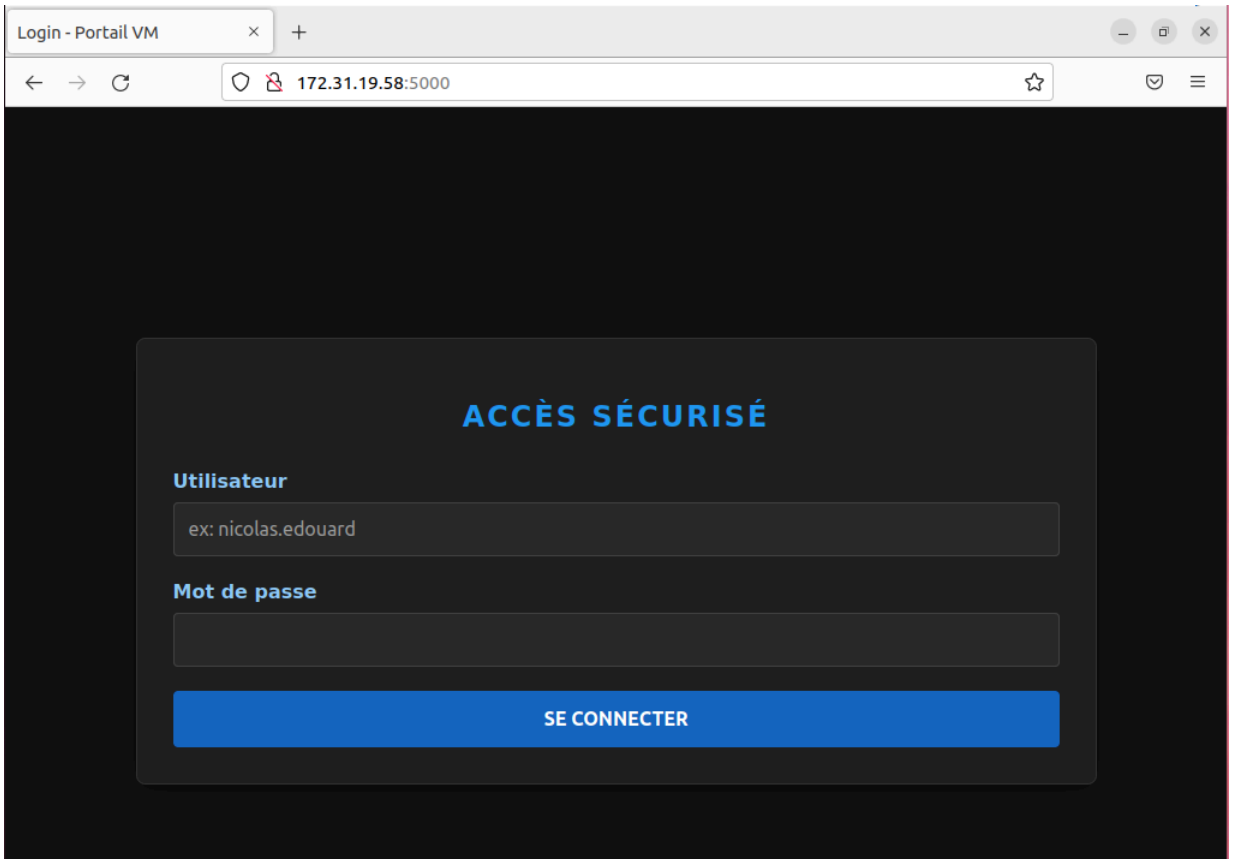
```
# Construction du nom DNS (FQDN)
vm_fqdn = f"{full_name}.dom-famchon.rt.lan"

# Création immédiate de la connexion dans Guacamole
# Plus besoin d'attendre l'IP, la résolution DNS se fera à la connexion
create_guacamole_connection(full_name, vm_fqdn, rdp_user, rdp_pass)
```

Résultat :
L'étudiant voit apparaître le bouton "Se connecter" quasi instantanément après la création, rendant le service fluide et réactif.

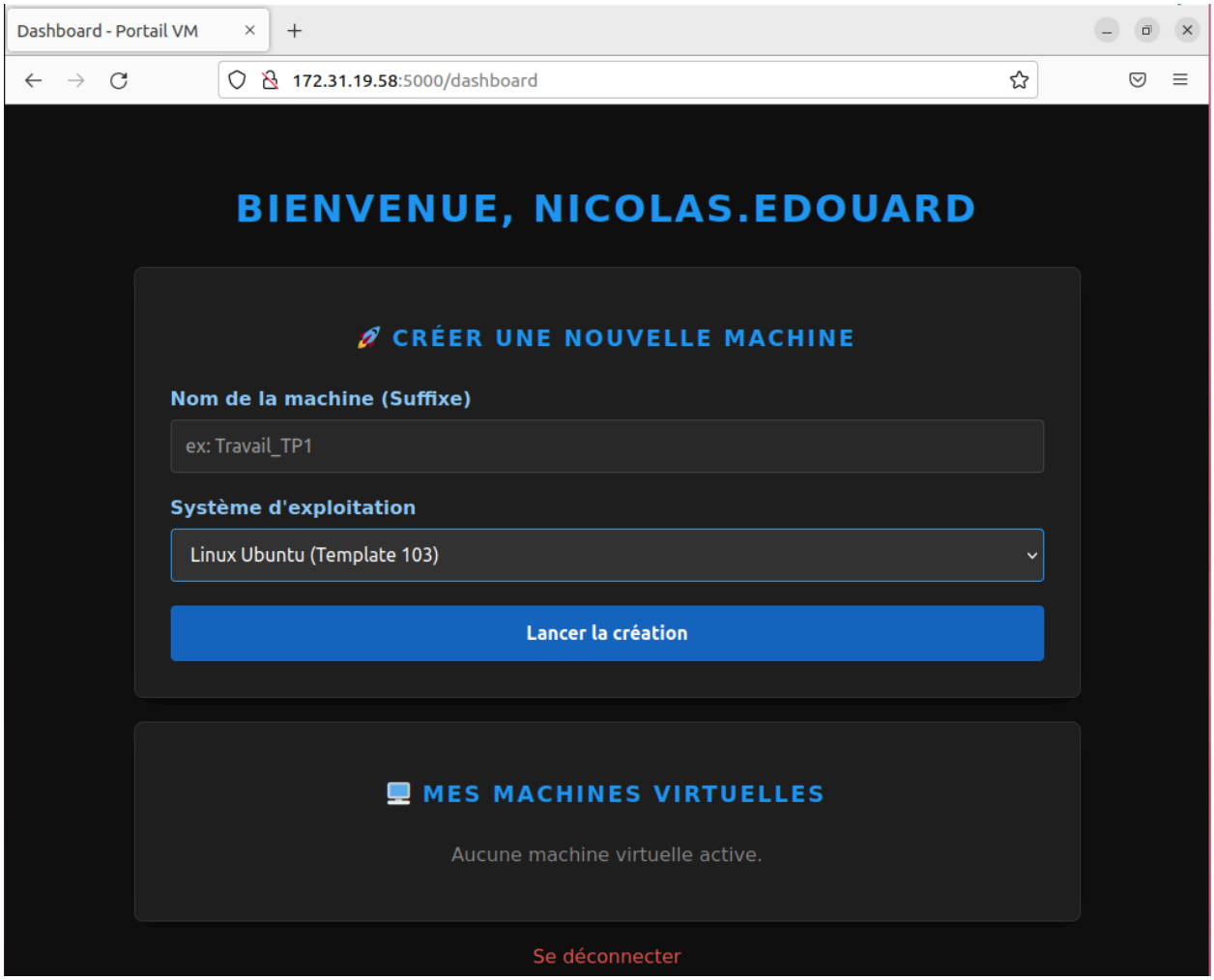
6.7. Interface Utilisateur (Frontend)

L'interface a été conçue avec des templates HTML/CSS simples mais fonctionnels (Thème sombre) :

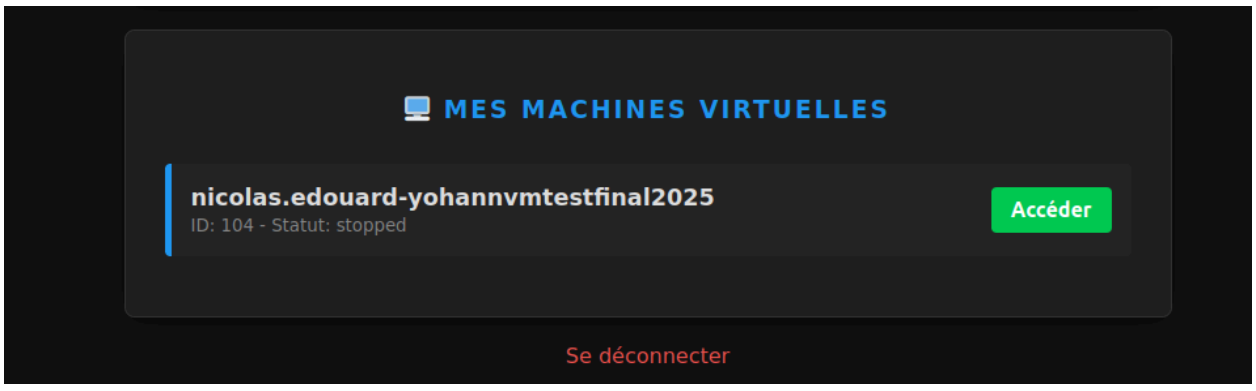
- **Page de Login :**
  Formulaire simple demandant User/Pass AD.
- **Dashboard :**
  - Menu déroulant pour choisir le Template (Windows 10 / Ubuntu).
  - Liste dynamique des VMs actives de l'étudiant.
  - Bouton d'accès direct redirigeant vers le client web Guacamole.



Portail de login.html.



Création d'un VM en utilisant un template.



On voit la VM bien créée.

VII. PHASE 6 : GESTION DES MACHINES VIRTUELLES

7.1. Préparation des "Golden Images"

Nous avons créé des templates (Ubuntu Desktop, Windows 10, Kali).
Pour garantir leur fonctionnement dans notre réseau privé :

- Installation de l'agent qemu-guest-agent.
- Installation des outils de jonction au domaine (realmd, adcli).

7.2. Intégration automatique (Zero Touch)

Nous avons intégré un script au démarrage des templates (/usr/local/bin/join-ad.sh).
Dès que la VM démarre et reçoit une IP du DHCP de pfSense, elle contacte l'AD et s'y inscrit automatiquement. Cela permet à l'étudiant de se connecter avec ses identifiants école directement sur la VM.

VIII. ASPECTS ENVIRONNEMENTAUX ET CONCLUSION

8.1. Green IT et Sobriété Numérique

Dans un contexte où l'empreinte carbone du numérique est une préoccupation majeure, notre projet d'infrastructure VDI s'inscrit pleinement dans une démarche de "Green IT" et de rationalisation des ressources.

- **Consolidation des Serveurs (Hardware) :**

  L'usage de la virtualisation via Proxmox VE permet une mutualisation forte du matériel. Au lieu de mobiliser 30 unités centrales physiques fonctionnant simultanément pour une session de TP, nous concentrons la puissance de calcul sur un unique serveur physique. Cette approche réduit drastiquement la consommation électrique directe, mais également les besoins en climatisation de la salle serveur.

- **Optimisation Énergétique Dynamique :**

  Contrairement à un parc de PC classiques qui consomment de l'énergie même en inactivité, notre hyperviseur alloue les ressources (CPU, RAM) dynamiquement. Une VM éteinte ne consomme rien.

- **Lutte contre le "VM Sprawl" (Gaspillage de stockage) :**

  L'un des points forts de notre solution réside dans l'automatisation via Python. Notre script de gestion intègre une logique de cycle de vie des machines virtuelles. En facilitant la suppression des environnements après usage, nous évitons l'accumulation de "machines zombies" qui consomment inutilement de l'espace disque et des ressources de sauvegarde, prolongeant ainsi la durée de vie des supports de stockage (SSD/HDD).

- **Réduction des Déplacements :**

  En offrant un accès distant performant via Apache Guacamole, nous permettons aux étudiants et enseignants de travailler depuis chez eux, limitant les déplacements physiques et les émissions de CO_2 associées au

transport.

8.2. Bilan du Projet

Ce projet SAE 5.01 a représenté un défi technique stimulant, nécessitant la convergence de compétences transversales : l'administration système (Windows/Linux), l'ingénierie réseau (Routage/Pare-feu) et le développement logiciel (Python/API).

Objectifs Atteints : Nous avons réussi à livrer une plateforme "Clef en main". Le portail web développé offre une abstraction complète de la complexité technique pour l'utilisateur final. L'accès distant est fluide, sécurisé par une double authentification (LDAP + MySQL) et protégé par une segmentation réseau stricte via pfSense. L'automatisation du déploiement (Clonage + DNS + Jonction AD) transforme une tâche de 30 minutes en un processus de quelques secondes.

Difficultés Surmontées et Montée en Compétences : Le parcours a été ponctué d'obstacles techniques qui ont enrichi notre apprentissage :

- **La gestion DNS :**

  Comprendre la nécessité des redirecteurs et des "Domain Overrides" pour faire cohabiter un AD public et un réseau privé NATé a été crucial.

- **Les contraintes réseaux (Proxy) :**

  Le développement du script Python a nécessité une adaptation fine (`NO_PROXY`) pour contourner les restrictions du proxy universitaire lors des appels API locaux.

- **L'interopérabilité :**

  Faire dialoguer des briques hétérogènes (Proxmox en REST, Guacamole en MySQL/API, AD en LDAP) a validé notre capacité à intégrer des systèmes complexes.

Conclusion :

L'infrastructure est aujourd'hui pleinement opérationnelle, résiliente et documentée. Elle répond au cahier des charges initial et est prête à être déployée en production pour assurer des sessions de travaux pratiques.

ENGLISH SYNTHESIS

Project Summary: Configuration and Deployment of a Virtual Classroom Infrastructure

Conclusion :

This project successfully bridges the gap between System Administration, Network Engineering, and Software Development. We have delivered a fully functional, secure, and user-friendly VDI platform. Key achievements include:

- **Full Automation :**

  A custom Python/Flask web portal orchestrates Proxmox and Guacamole APIs to provision VMs in seconds.

- **Security :**

  Network segmentation via pfSense and hybrid authentication (Active Directory/LDAP + MySQL) ensures a secure environment.

- **Problem Solving :**

  We successfully overcame significant technical challenges, particularly regarding complex DNS routing in a NATed environment and handling proxy constraints within API communications.

The infrastructure is now operational and ready for production use in educational settings.