SAE 5.01

Flavien Marchand
Ali Bouziani

Concevoir, réaliser et présenter une solution technique

Sommaire

Conception et planification :

Nous avons donc comme projet de mettre en place une solution d'hypervision et d'y ajouter des fonctionnalités en utilisant d'autres services pour l'accès à distance de machines virtuelles configurées comme machines de TP (dans le cadre d'une formation R&T).

Donc ici, nous avions comme phase de planification la recherche et la répartition des tâches.

Nous avons donc, dans un premier temps, réservé notre début de projet à la recherche des différents services d'hyperviseurs disponibles dans le marché, et leurs pertinences dans notre projet, voir lequel serait le plus efficace et pertinent dans notre projet. Il a aussi fallu voir ses disponibilités et capacités pour nous permettre de savoir la portée de réalisation de notre projet.

Pour ces recherches nous avons décidé de faire un tableau avec les hyperviseurs les plus influents et populaires :

Nom	**Avantages**	**Inconvénients**	**Profil**
Proxmox VE	-> Combine la virtualisation et les conteneurs. -> Gratuit et open source, avec une interface web conviviale. -> Supporte la gestion de clusters et haute disponibilité.	-> Propose une version communautaire payante -> Moins de fonctionnalités que d'autres solutions payantes	Idéal pour une utilisation dans une PME ou personnelle, de par son utilisation simple et gratuite.
Microsoft HyperV	-> Intégré dans Windows Server, facile à configurer et à gérer pour les utilisateurs de cet écosystème. -> Bien intégré avec d'autres produits Microsoft, comme Active Directory et Azure. -> Support natif des environnements Windows.	-> Manque de compatibilité avec d'autres os. -> Administration moins complexe	Idéal pour des environnements principalement Windows ou avec l'utilisation d'outils Windows ou la gestion de groupes / utilisateurs.

VMware ESXi	-> Très fiable et stable, largement utilisé dans les entreprises. -> Large catalogue de fonctionnalités, gestion avancée des ressources et intégration avec des solutions de virtualisation. -> Très populaire et très documenté avec une communauté active.	-> Payant (cher) -> Une consommation élevée des ressources.	Idéal pour les environnements d'entreprise avec des besoins critiques et un budget conséquent pour la virtualisation.
KVM	-> Open source et gratuit. -> Propose d'excellentes performances. -> Support natif sur les systèmes linux.	-> Très complexe (pas user friendly). -> Nécessite des compétences techniques poussées dans l'implémentation et la gestion de MV.	Idéal pour une utilisation plus poussée dans la configuration d'implémentations de machines. Aussi pour une utilisation de machines Linux.

Ici, d'après notre tableau nous en avons conclus que ;

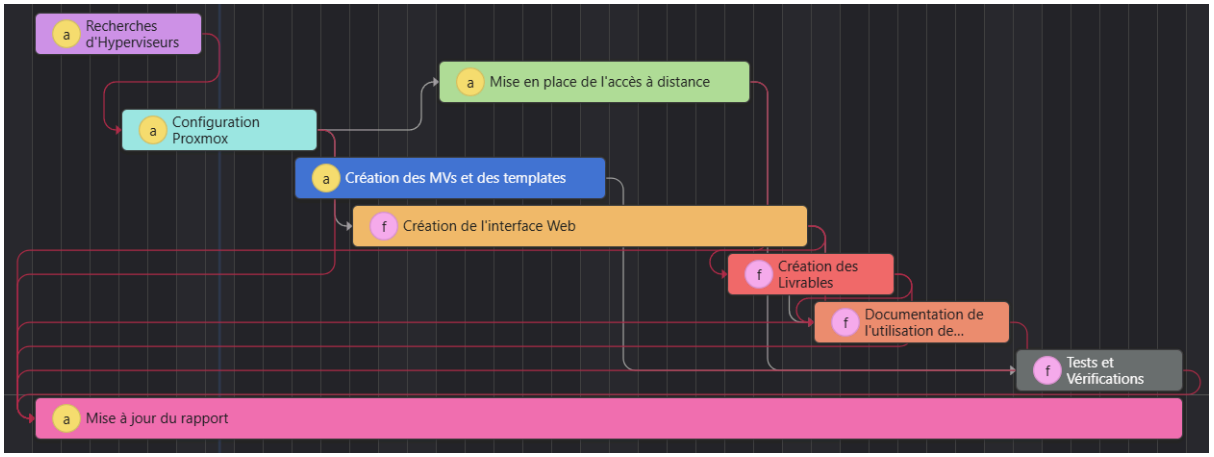
- **VMware ESXi** était adapté pour une utilisation **professionnelle** en entreprise et à **grande échelle**.

- **KMV** était adapté à une utilisation **robuste**, plus **complexe** et **personnalisée** d'hypervision, aussi dans un **déploiement** majoritairement de **machines Linux**.

- **Microsoft Hyper-V** était adapté à une utilisation plus simple **d'intégration** majoritairement **windows**, pour une utilisation unifiée de **services d'administration Windows**.

Donc, suite au dressage des solutions disponibles et leur utilisation générale, nous nous sommes tournés vers Proxmox PE, car plus simple d'utilisation et gratuit, il possède aussi beaucoup de fonctionnalités que nous utiliserons dans notre projet qui sont pertinentes et parfaites pour le déploiement du projet.

Nous avons ensuite décider de planifier et attribuer les tâches pour le projet sous la forme d'un gantt :



La répartition est faite de la manière suivante :

- Le cercle avec la lettre à l'intérieur est pour l'attribution de la tâche : a pour Ali Bouziani et f pour Flavien Marchand.

Il est aussi important de savoir que nous avions initialement planifié le projet de scéances à scéances pour définir clairement l'avancée du projet et permettre de nous attribuer une deadline importante pour respecter les délais des étapes du projet.

Hors quelques séances plus tard, nous avons abandonné cette méthode pour une plus flexible et sans délai pour réaliser les tâches ensembles et pouvoir unifier la configuration (d'où le gantt sans délai avec uniquement les tâches à réaliser et leurs liens et attributions).

Installation et configuration de Proxmox PE :

Donc pour l'installation de Proxmox PE, nous avons décidé d'utiliser les ressources suivantes ;

CPU : 8 coeurs
RAM : 16 Go
Stockage : 240 Go SSD / 1 To HDD

(Il est important de prendre en compte que notre installation diffère car elle a été effectuée sur une machine physique (un serveur personnel)).

Le choix des performances peut varier en fonction de l'utilisation, mais il faut noter que plus le nombre de machines augmente, plus il faudra de performances. Dans le cadre du projet, en virtualisant Proxmox, l'utilisation de celui-ci sera beaucoup plus réduite, étant donné le besoin de ressources (virtualisation dans de la virtualisation).

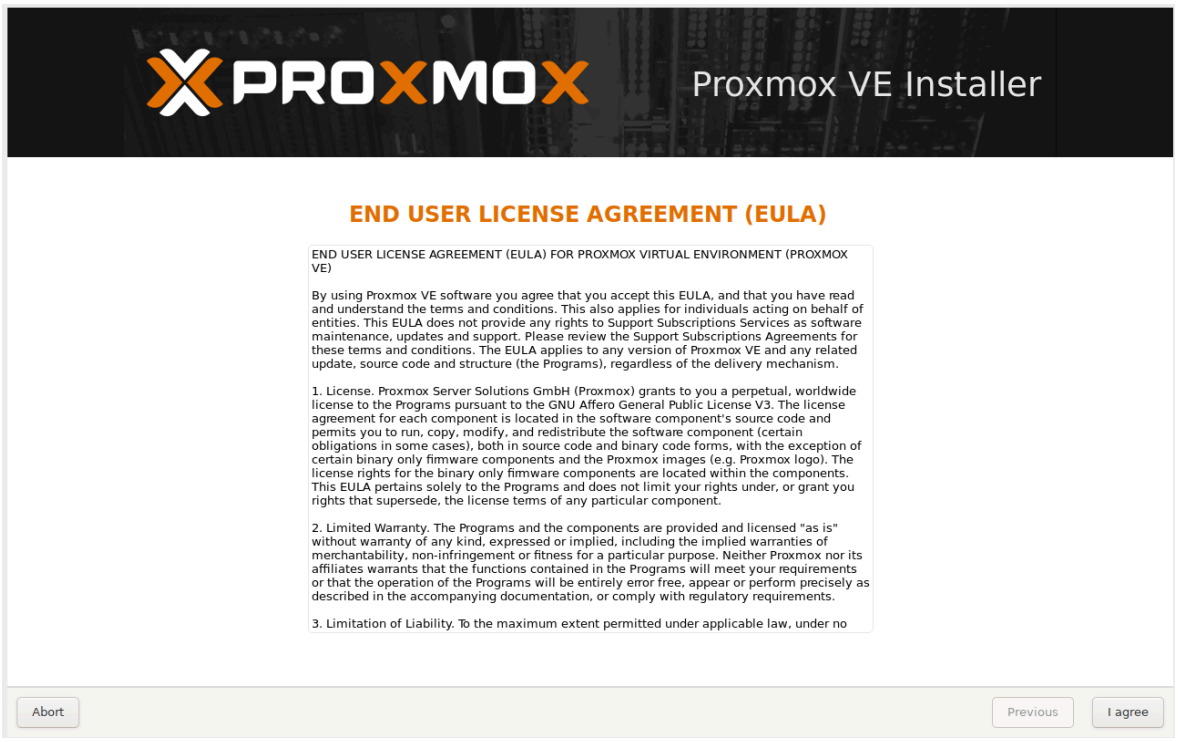
Donc pour l'installation ;

Nous sommes allés chercher un iso de proxmox PE (ici, un iso de la version 8.3-1) sur le site officiel de proxmox.

RT3-FI

On sélectionne l'installation en version graphique pour pouvoir effectuer l'installation avec un environnement graphique (plus simple et rapide).

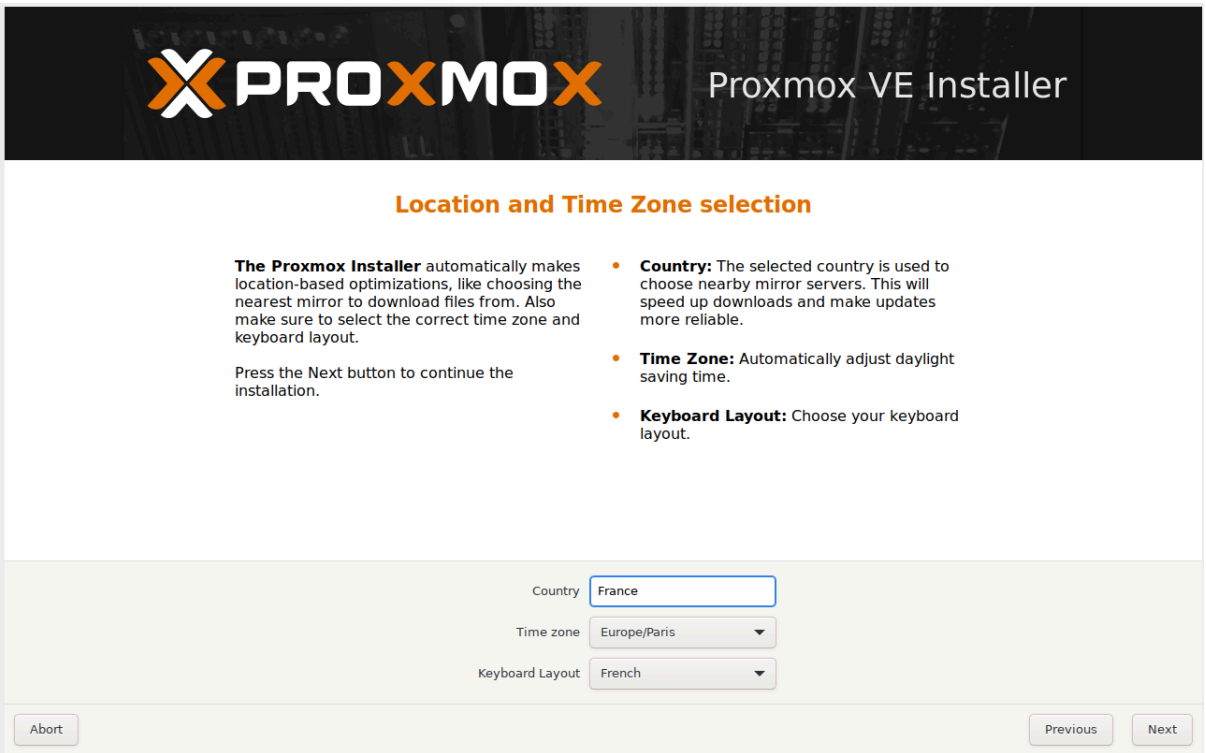
Ensuite, on accepte les conditions d'utilisation :



On vient ensuite sélectionner le disque dans lequel on veut installer proxmox :

RT3-FI

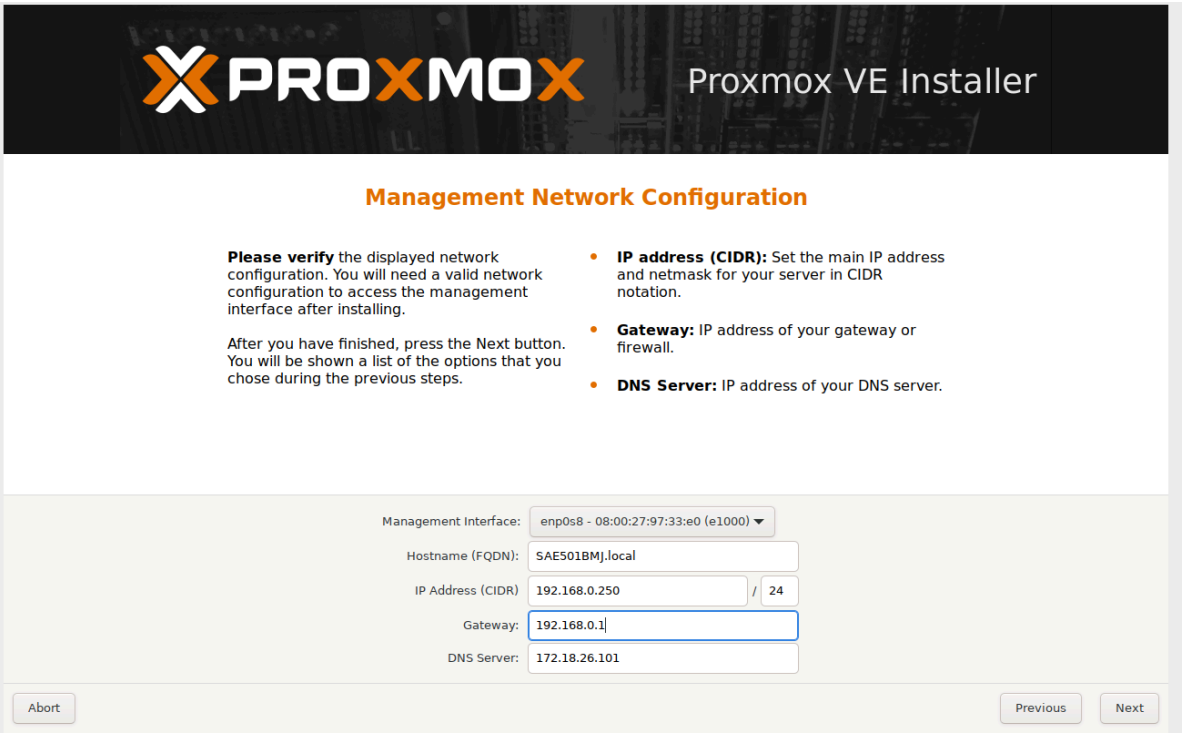
On sélectionne la zone démographique et la disposition du clavier :



Ensuite le mdp du user root :

RT3-FI

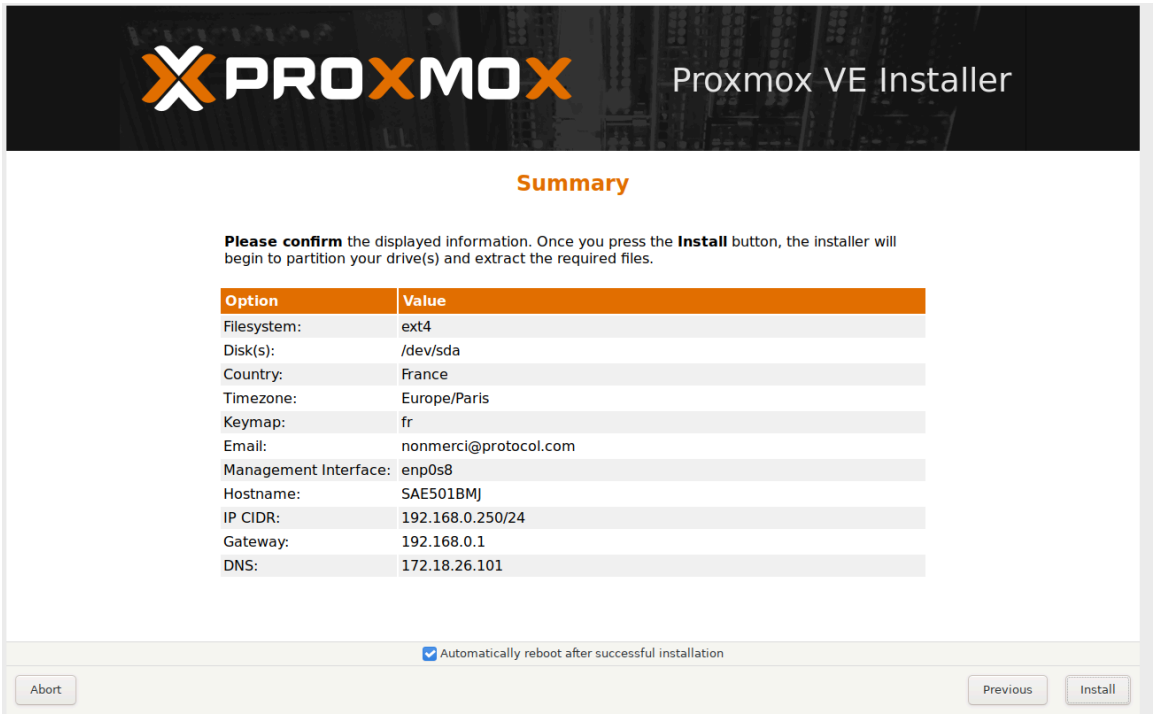
Enfin, la configuration réseau de la machine :



ici, il est intéressant de relever l'ip que l'on met en statique nous permettra d'accéder à l'interface web de proxmox.

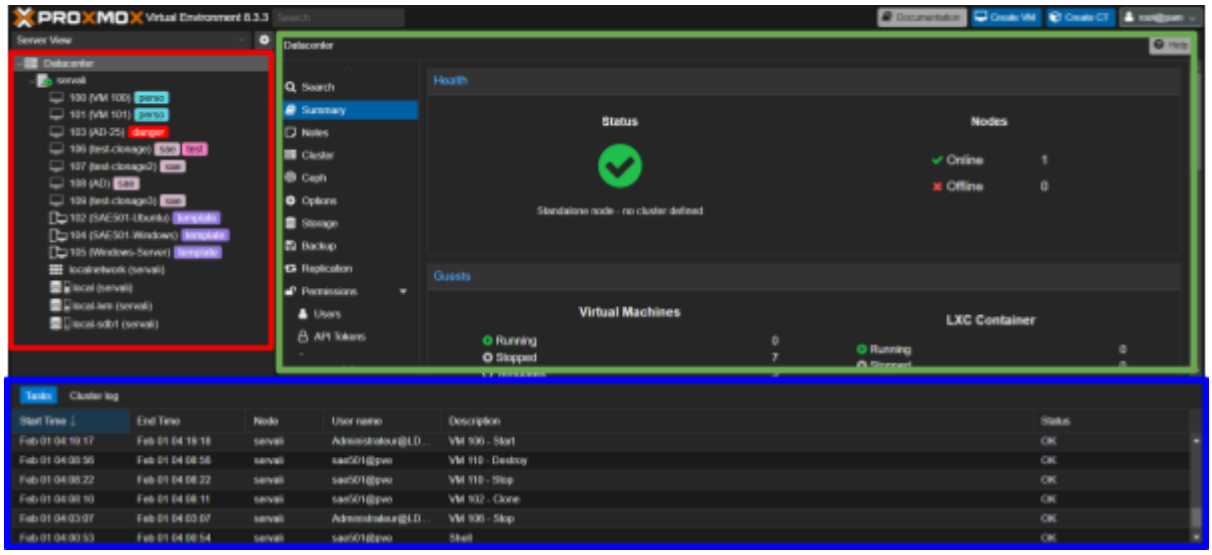
Et enfin, avant d'installer et redémarrer la machine, une page de résumé de notre configuration :

RT3-FI

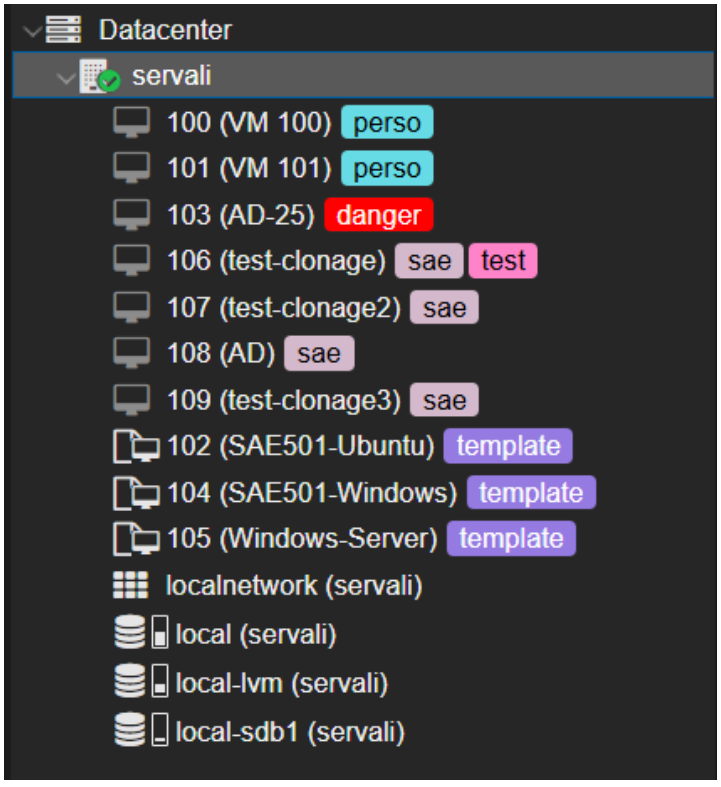

PROXMOX
Proxmox VE Installer
Summary
Please confirm the displayed information. Once you press the Install button, the installer will begin to partition your drive(s) and extract the required files.
Option
Value
Filesystem:
ext4
Disk(s):
/dev/sda
Country:
France
Timezone:
Europe/Paris
Keymap:
fr
Email:
nonmerci@protocol.com
Management Interface:
enp0s8
Hostname:
SAE501BMJ
IP CIDR:
192.168.0.250/24
Gateway:
192.168.0.1
DNS:
172.18.26.101
Automatically reboot after successful installation
Abort
Previous
Install

Utilisation de Proxmox PE :

Une fois proxmox installé et la machine redémarrée, on peut accéder à l'interface web (http://IPMachine:8006) :



Sur l'interface web, la partie **rouge** permet de naviguer entre les machines, et la partie **verte** d'avoir accès aux différentes informations ou configurations des machines, et en **bleu** les logs, l'historique des dernières actions faites sur le serveur .

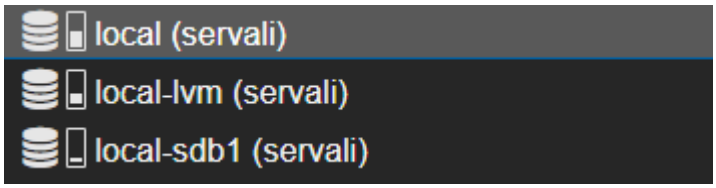


-> ici la node servali

Ici, il y a la notion de node qui est importante, un node (noeud) désigne un serveur, proxmox définis les serveurs de cette façon car on peut implémenter simplement un cluster de serveurs proxmox (pour la répartition des charges ou la haute disponibilité).

RT3-FI

Nous avons donc en premier, la partie de stockage local :



Il est premièrement utilisé pour y déposer des isos pour une installation locale, c'est notre cas ici ;

Il peut aussi être utilisé pour y déposer différents fichiers, ses utilisations les plus courantes sont pour effectuer des sauvegardes de machines, stocker des isos ou des templates de conteneurs.

Il est aussi important de noter qu'il est possible de partager ces répertoires en fonction de groupes ou de users spécifiques (ceux que l'on définis).

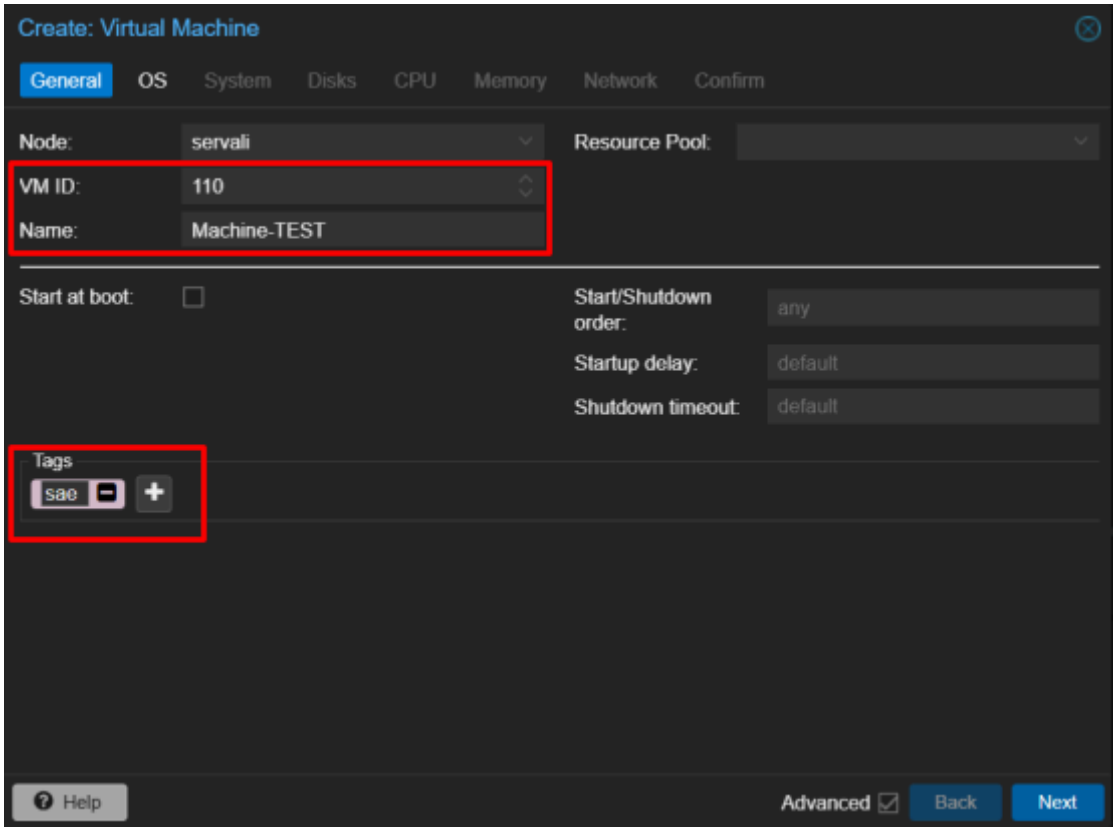
Dans notre cas, on l'utilise principalement pour les iso, comme le montre la capture ci-dessus, nous avons installé plusieurs iso.

Créer une machine virtuelle sur Proxmox PE

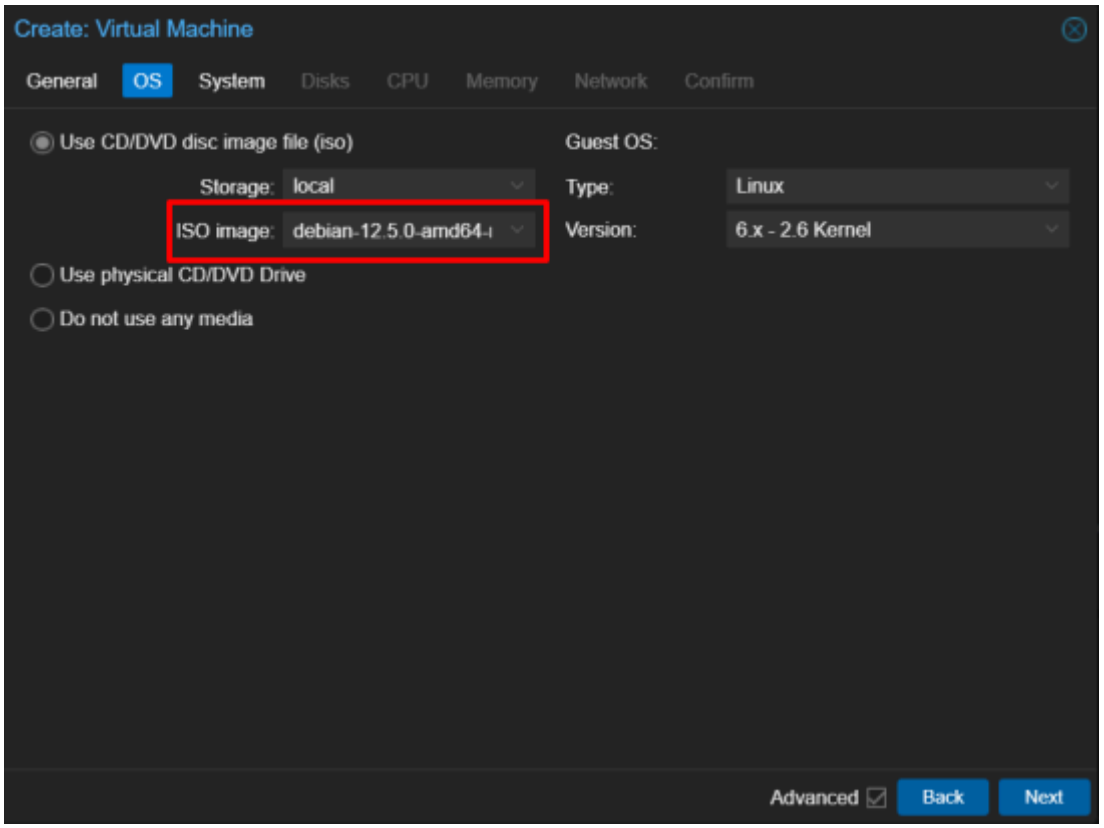
Lors de la création d'une MV, on va définir sa configuration aux besoins nécessaires :

On passe par plusieurs onglets de configuration des MVs :

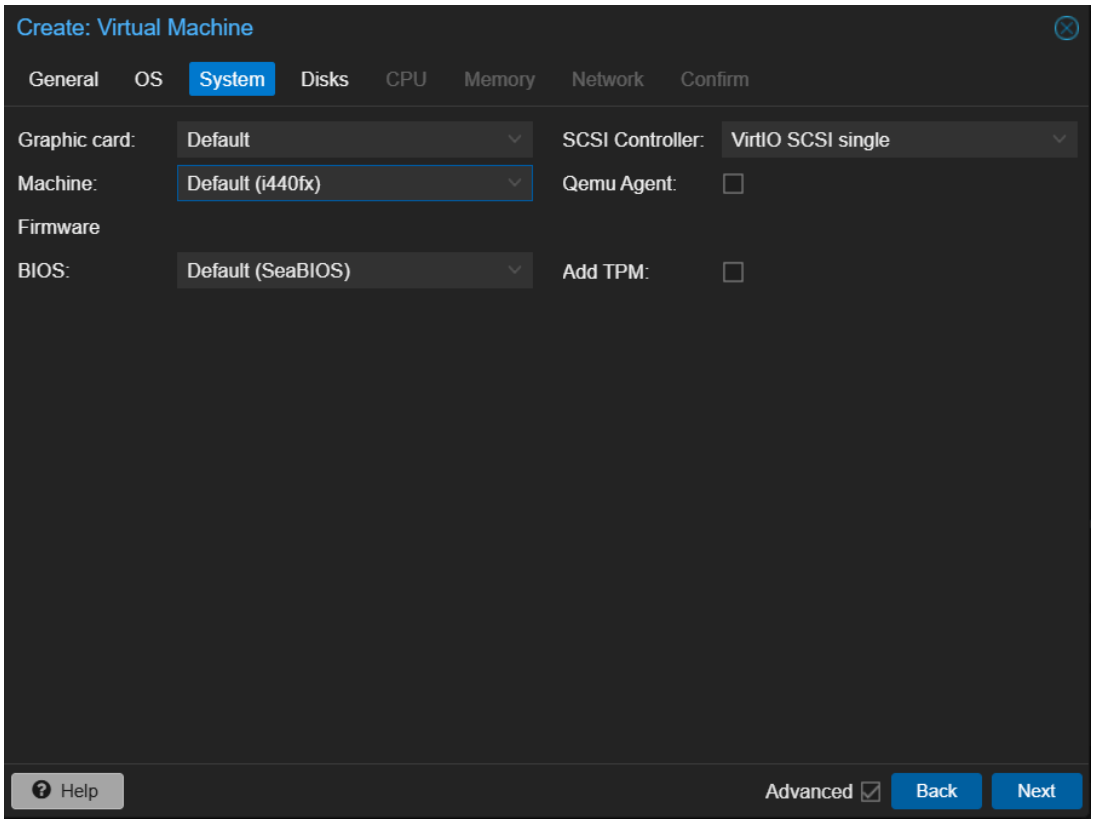
Ici, on donne un nom et un ID à notre machine, on peut aussi sélectionner un tag pour notre machine :



On vient ensuite choisir l'iso que l'on veut installer :



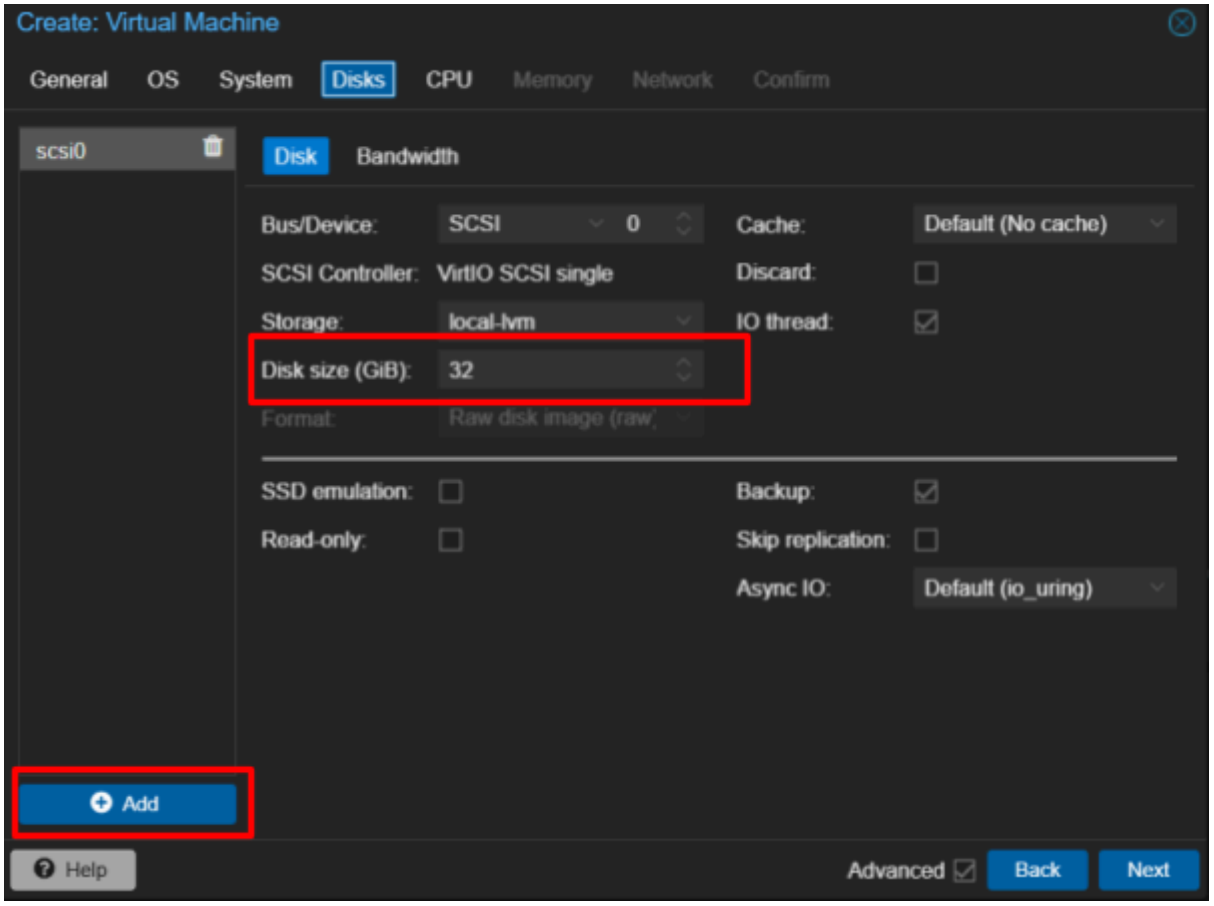
Ici, on laisse les paramètres par défaut :



Ces paramètres permettent de configurer les différents comportements, et manières de se comporter avec les iso. Notamment avec le choix de différents BIOS, ou le paramètre “Machine” qui permet de choisir le différent type de matériel virtuel émulé (la manière dont la machine va se comporter).

Dans notre cas, nous utilisons des iso classiques, qui ne nécessitent aucun changement dans ces paramètres de configuration.

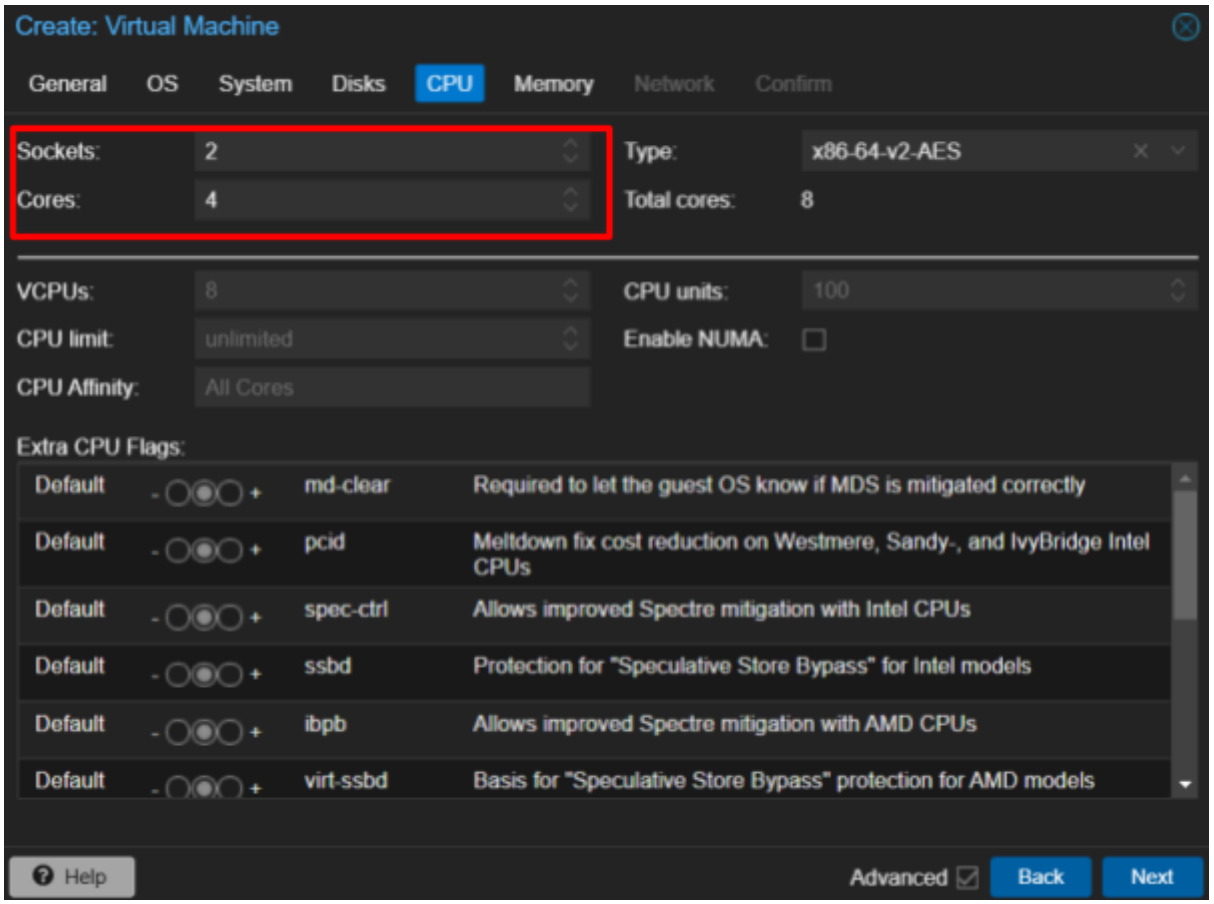
Puis ensuite, on va choisir le stockage de notre machine :



On peut décider d’ajouter une autre partition de stockage, ou encore le comportement du stockage.

RT3-FI

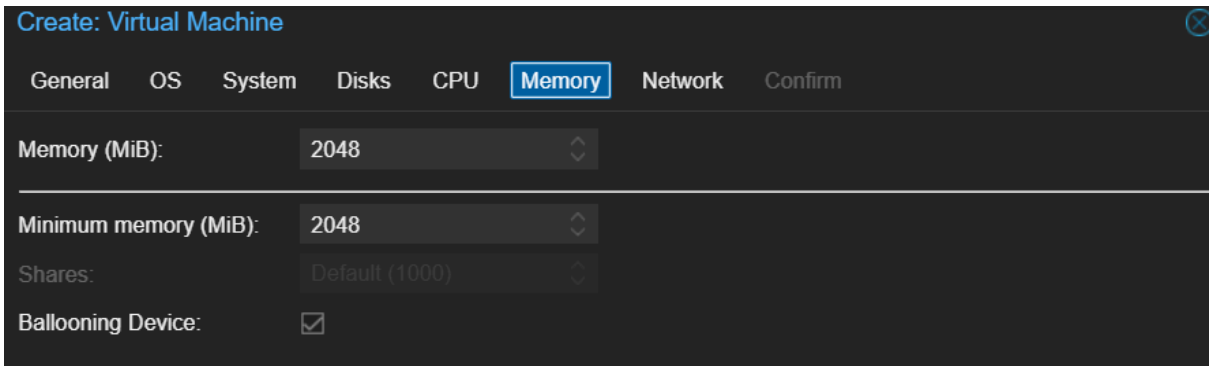
On définit par la suite le nombre de coeurs à allouer à notre machine :



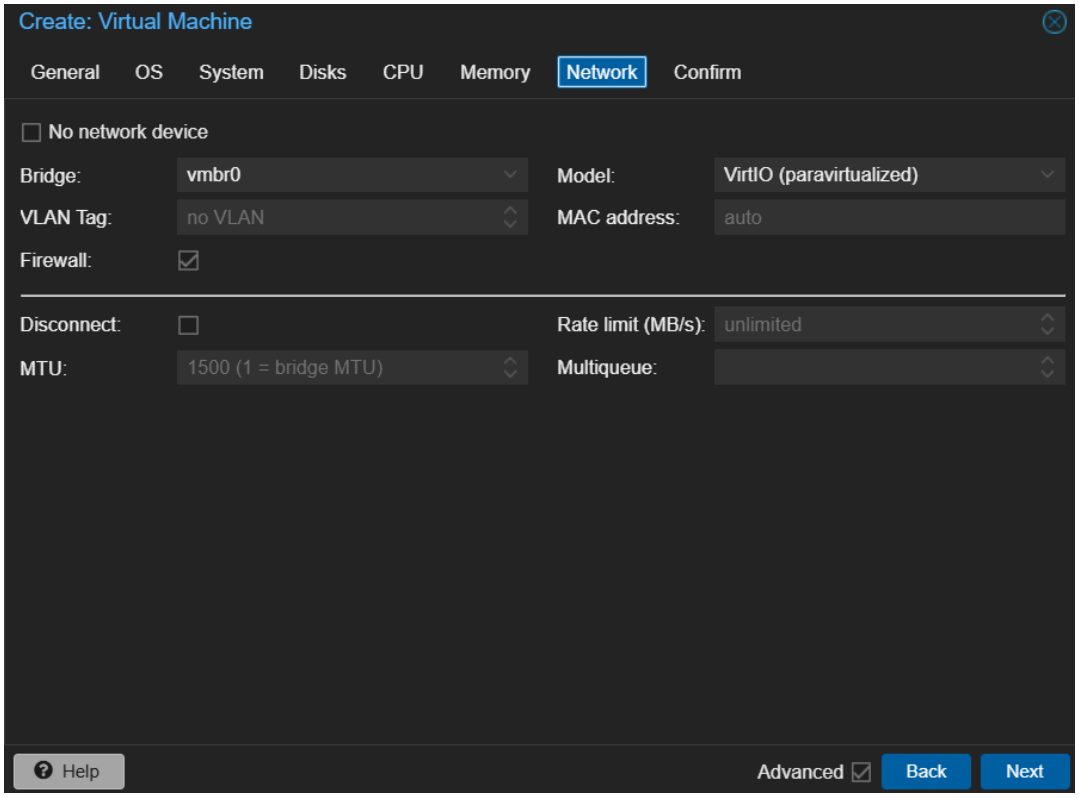
Ces nombreux paramètres permettent d'agir sur le fonctionnement du CPU, en le limitant ou en activant des fonctions spécifiques ou le type du CPU.

Dans notre cas, nous n'avons aucun besoin spécifique à ces changements.

Par la suite, on sélectionne la RAM à allouer à la machine :

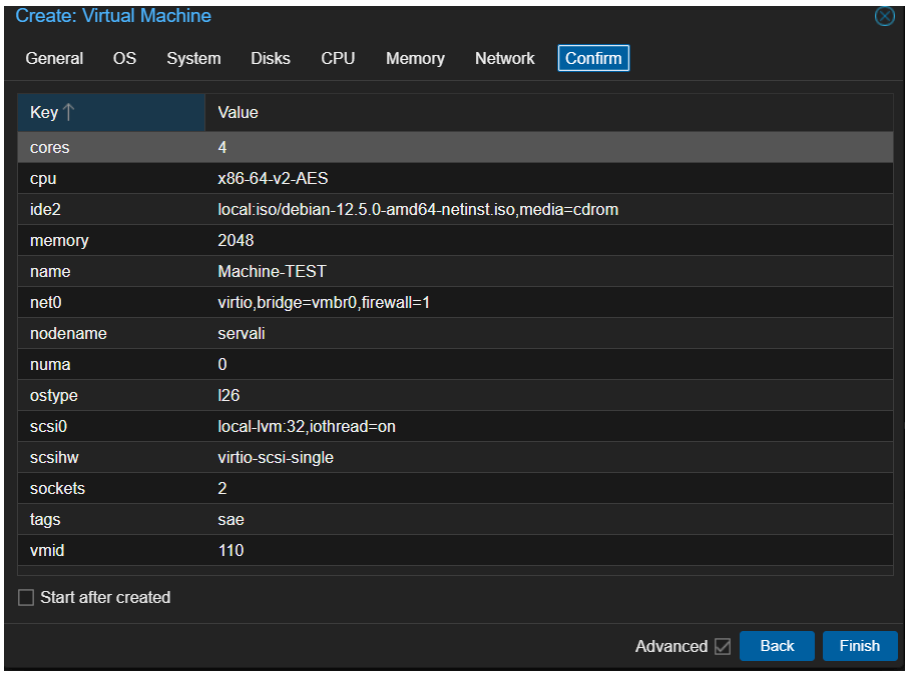


Puis, enfin, on vient définir l'interface réseau :

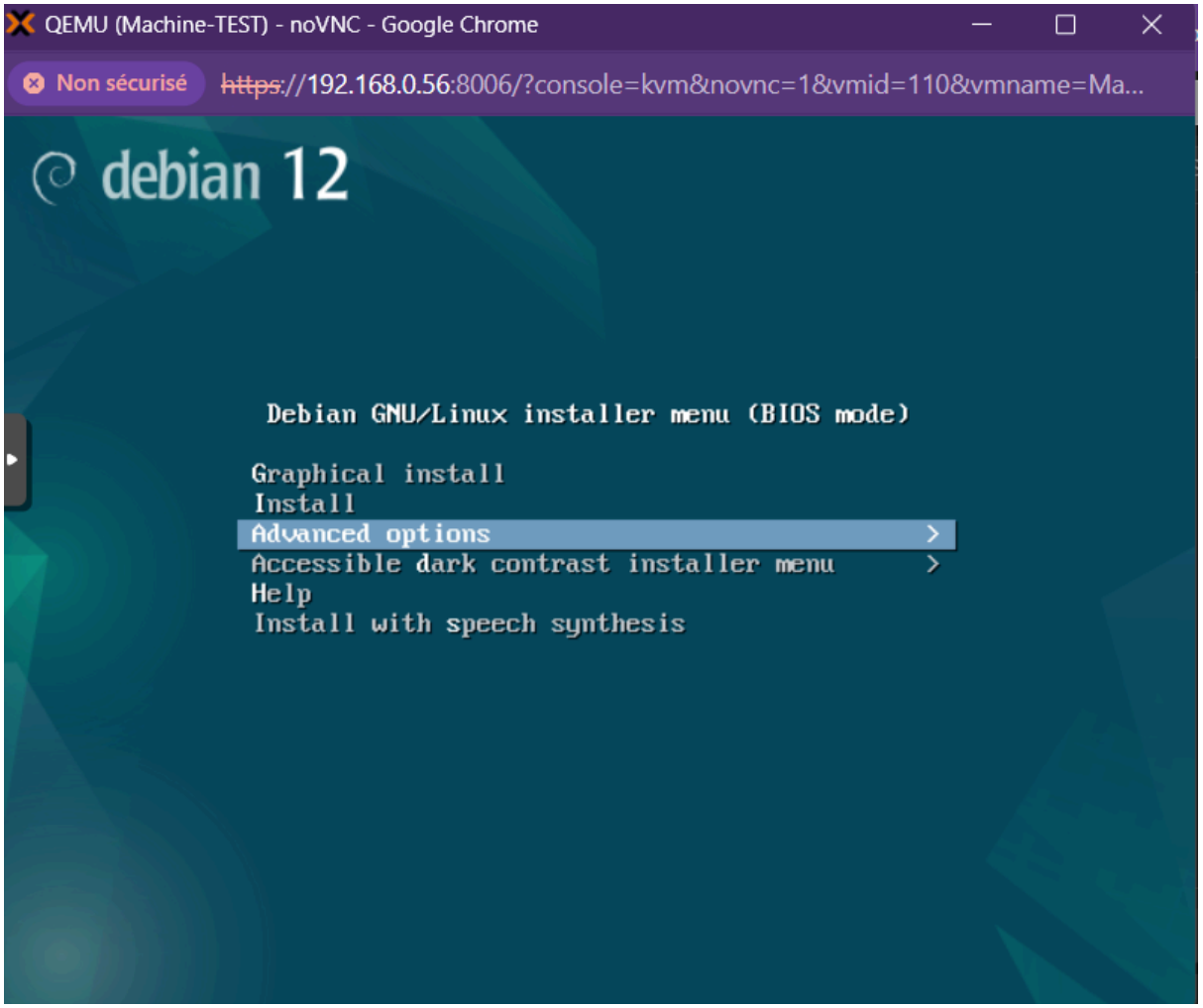


On peut choisir différents modèles de cartes réseau, la manière dont on connecte la machine au réseau, des paramètres de firewall et de vlans pour une configuration dans un environnement plus poussé.

Enfin on valide et crée notre MV :



Une fois la machine crée, il faut la démarrer et lancer le programme d'installation de l'iso :



(Pour se faire, il faut démarrer la machine et cliquer sur le bouton en haut à droite "console", on peut choisir entre une interface graphique et un shell via ssh.)

RT3-FI

Définition des besoins pour des machines de TP :

Une fois avoir créé les machines, nous avons eu comme réflexion quels seraient les besoins d'étudiants en BUT Réseaux et Télécommunications dans le cadre de TPs.

Nous avons donc dressé un tableau des différents services à installer sur nos machines :

Services et Logiciels :

services de base	ssh, nano, curl, wget
services réseau	wireshark, bind9, dhcpd, tftpd, nginx, apache2, iptables, ufw, wireguard
services développement	git, python3, vscode
services de virtualisation	docker, virtualbox, gns3
services gestion à distance	zabbix, nagios
services télécoms	gns3, scapy
services base de données	mysql, mariadb, postgresql, phpmyadmin, sqlite
services sécurité / cyber	nmap, fail2ban

Mais aussi des utilitaires tels que firefox, libreoffice.

Donc, la définition de ces services est à installer dans un environnement Linux, une fois ces services définis, nous avons l'environnement de travail destiné aux étudiants.

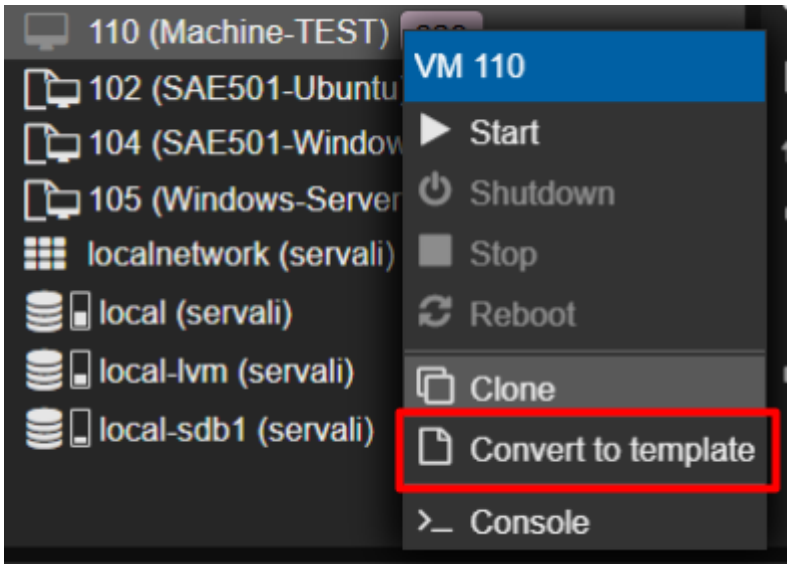
Création de templates :

Donc une fois la machine créée et configurée avec tous les services installés nous avons enfin une machine prête. Hors cette méthode est longue et minutieuse, il faut aussi passer par l'installation complète de l'os ce qui rend l'utilisation de machines neuves dans le cadre de tps ardue et peu fluide.

Pour régler ce problème, nous pouvons créer des templates, qui sont des versions vierges de machine déjà installées avec tous les services déjà prêts.

En fait, une template est juste une machine prête dont on va se servir pour cloner d'autres versions prêtes à être utilisées.

Pour créer une template, nous allons devoir sélectionner une machine prête :



On vient faire un clic droit sur la machine que l'on veut convertir en template, et on vient sélectionner l'option "Convert to template".

Il est important de noter qu'il sera alors impossible de récupérer ou modifier la machine une fois convertie en template, il sera alors possible uniquement de la cloner pour créer une nouvelle machine déjà prête.

Accès à distance :

Il a fallu ensuite permettre d'avoir accès à l'interface web depuis l'extérieur. Alors nous avons décidé de mettre en place un service VPN qui permettrait d'avoir accès au réseau privé à distance et de pouvoir utiliser l'interface de proxmox ou l'interface web pour travailler.

Pour configurer Wireguard, nous avons décidé d'utiliser wg-easy et de le faire tourner sur un docker via un docker-compose :

On crée le fichier docker-compose.yml :

```
version: '3.3'
services:
  wg-easy:
    image: ghcr.io/wg-easy/wg-easy
    container_name: wg-easy
    environment:
      - WG_HOST=192.168.0.174
      - PASSWORD_HASH=$2a$12$193MkEVpxfLVmQZy1ZwgIe7bGhLfeunwBNli4Tz0LKiTvc4tpH042
    volumes:
      - ~/.wg-easy:/etc/wireguard
    ports:
      - "16444:51820/udp"
      - "16445:51821/tcp"
    restart: unless-stopped
    cap_add:
      - NET_ADMIN
      - SYS_MODULE
    sysctls:
      - net.ipv4.ip_forward=1
      - net.ipv4.conf.all.src_valid_mark=1
```

Dans ce fichier on vient changer :

- l'ip de l'host (on met celle du serveur)

- le mdp (on génère un hash du mot de passe avec la commande suivante :)

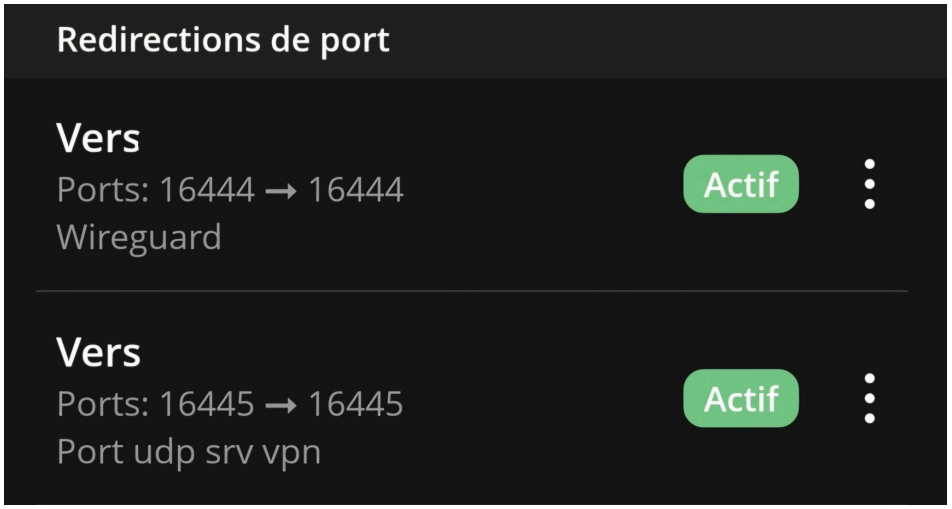
```
docker run --rm -it ghcr.io/wg-easy/wg-easy wgpw 'YOUR_PASSWORD'
PASSWORD_HASH='$2b$12$coPqCsPtcFO.Ab99xylBNOW4.Iu7OOA2/ZIboHN6/oyxca3MWo7fW' // literally YOUR_PASSWORD
```

- les ports (on ouvre les ports réseau pour wireguard en UDP et TCP)

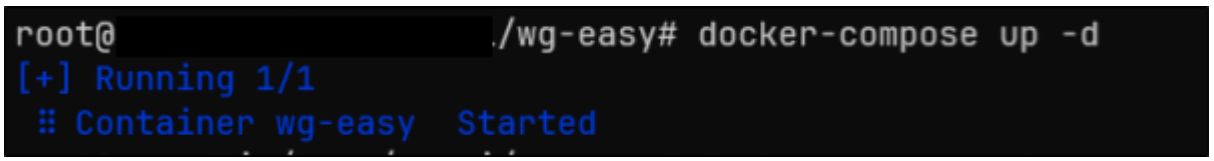
Dans mon cas les ports de ma box

RT3-FI

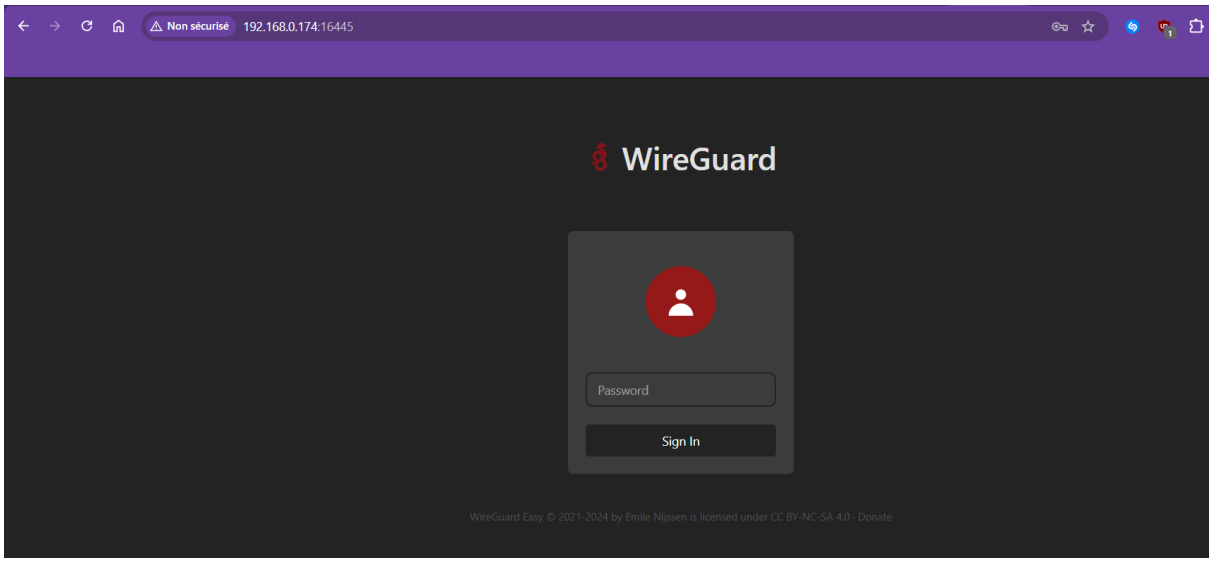
On va par la suite ouvrir les ports du réseau :



Puis on lance le fichier docker-compose.yml :

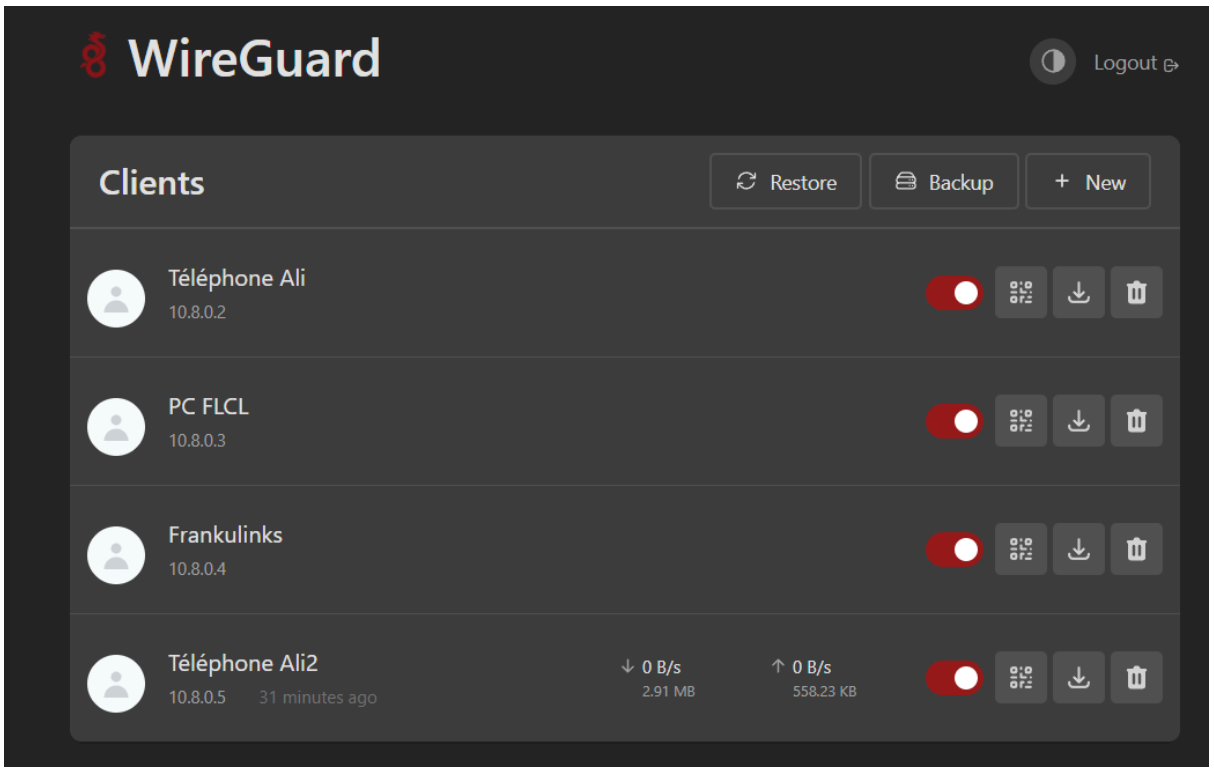


Et on va sur l'interface web :



On se connecte et on peut envoyer des configurations de vpns à toute personne qui en veut ou en a besoin.

RT3-FI



On peut interagir avec une interface web simple et rapide pour la configuration d'ajout d'utilisateurs pour le vpn. On peut aussi générer un QR code pour partager la configuration du VPN et la télécharger sous forme de fichier texte. On peut aussi gérer la connexion des utilisateurs et la couper si besoin.

Nous pouvions aussi avoir comme autre solution d'ouvrir les ports web http et https pour accéder directement à l'interface web et utiliser un nom de domaine pour y accéder. Hors il semblait plus sécurisé et sensé d'utiliser l'option d'un VPN, une solution que l'on retrouverait en entreprise par exemple.

RT3-FI

Mettre en place le LDAP et le sécuriser

1. Installation des rôles Active Directory et LDAP

1. **Ouvrir le Gestionnaire de serveur** (`Server Manager`).
2. Aller dans **"Gérer" > "Ajouter des rôles et fonctionnalités"**.
3. Sélectionner **"Installation basée sur un rôle ou une fonctionnalité"** et cliquer sur **"Suivant"**.
4. Choisir le serveur correspondant et cliquer sur **"Suivant"**.
5. **Sélectionner "Services AD DS"** (Active Directory Domain Services).
6. Ajouter également **"Outils de gestion Active Directory"**.
7. Confirmer l'installation et attendre la fin du processus.

---

2. Configuration du contrôleur de domaine

1. Une fois AD DS installé, ouvrir **"Gestionnaire de serveur"**.
2. Dans la notification en haut à droite, cliquer sur **"Promouvoir ce serveur en contrôleur de domaine"**.
3. Sélectionner **"Ajouter une nouvelle forêt"** (si c'est un nouveau domaine).
4. Entrer un **nom de domaine** (exemple : `mon-domaine.local`).
5. Définir un **mot de passe** pour le Mode de récupération des services d'annuaire (DSRM).
6. Configurer les autres options et terminer l'installation.
7. Redémarrer le serveur une fois la configuration terminée.

---

3. Configuration de LDAP en mode sécurisé (LDAPS)

LDAPS nécessite un **certificat SSL/TLS**. Pour cela, on va générer un certificat avec l'Autorité de Certification (CA) de Windows Server.

3.1 Installation de l'Autorité de Certification

1. Dans **Gestionnaire de serveur**, aller à **"Gérer" > "Ajouter des rôles et fonctionnalités"**.
2. Ajouter le rôle **"Services de certificats Active Directory (AD CS)"**.
3. Sélectionner **"Autorité de certification"** et **"Inscription de certificats sur le Web"** (facultatif).
4. Installer le rôle et ouvrir **"Gestionnaire des services de certificats"** (`certsrv.msc`).
5. Choisir **"Autorité de certification d'entreprise"** et **"Autorité de certification racine"**.
6. Générer un certificat et finaliser l'installation.

3.2 Génération et attribution du certificat pour LDAPS

RT3-FI

1. Ouvrir **MMC** (`mmc.exe`).
2. Ajouter le composant logiciel enfichable **"Certificats"** > **"Compte ordinateur"** > **"Ordinateur local"**.
3. Aller dans **"Certificats (Ordinateur local)"** > **"Autorités de certification racines de confiance"** > **"Certificats"**.
4. Vérifier que l'**autorité de certification** est bien installée.
5. Aller dans **"Certificats" > "Personnel" > "Certificats"**.
6. Faire un **clic droit** > **"Toutes les tâches" > "Demander un nouveau certificat"**.
7. Suivre l'assistant et choisir un modèle de certificat compatible avec **"Authentification de serveur"**.
8. Une fois généré, ouvrir le certificat et **noter son empreinte**.

---

4. Activation de LDAPS

1. Ouvrir **l'Éditeur de registre** (`regedit.exe`).
2. Aller à : `HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\NTDS\Parameters`
3. Créer une nouvelle valeur **DWORD (32-bit)** nommée : `LdapEnforceChannelBinding`
4. Définir sa valeur à `0` pour désactiver la vérification stricte des canaux LDAPS.
5. Créer une autre valeur **DWORD (32-bit)** nommée : `LdapServerIntegrity`
6. Définir sa valeur à `2` pour forcer LDAPS.

---

5. Redémarrage

Redémarrer le service AD DS :

```
net stop ntds
net start ntds
```

---

6. Tester LDAPS avec LDP.exe

1. Ouvrir **LDP.exe** (`ldp.exe`).
2. Aller dans **"Connexion" > "Se connecter"**.
3. Entrer **"localhost"** et sélectionner **Port 636** (LDAPS).
4. Cocher **SSL** et cliquer sur **OK**.
5. Si la connexion réussit, LDAPS est bien configuré.

RT3-FI

Lier le LDAP au Proxmox

Créer ou modifier ce fichier sur le Proxmox :

`/etc/pve/domains.cfg`

Et y mettre :

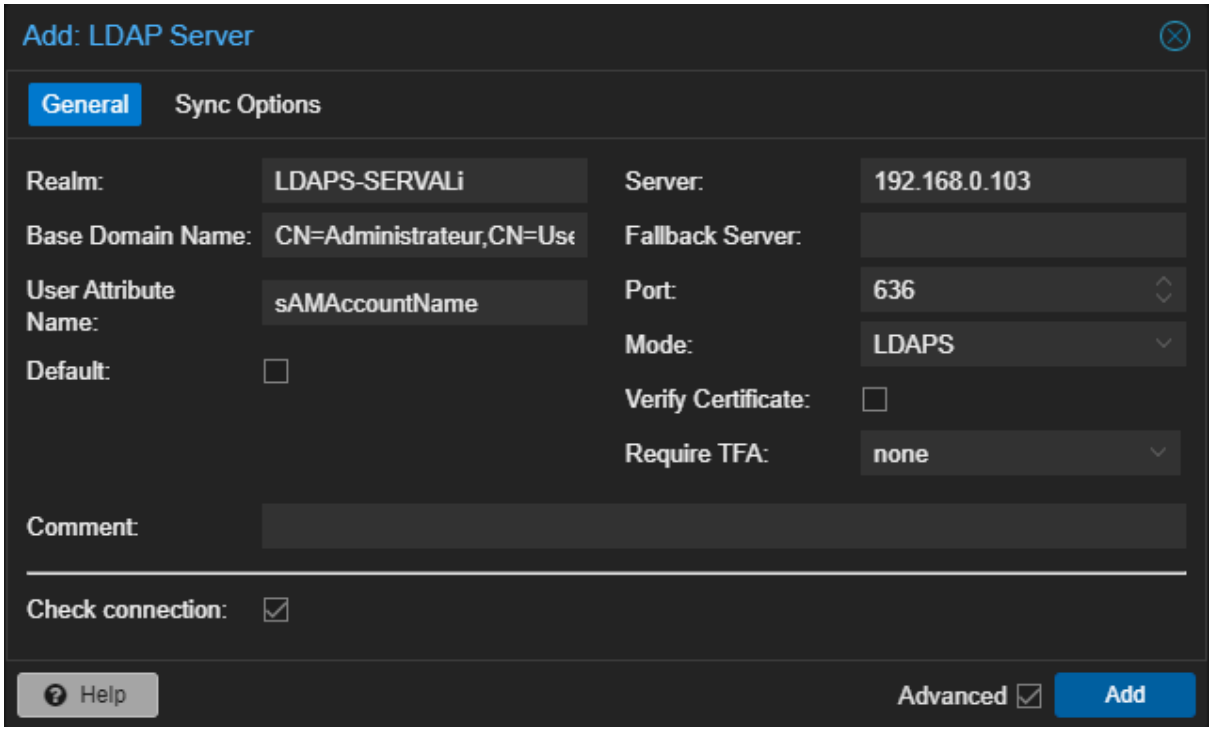
```
ldap: AD-LDAPS
    server1 192.168.0.103
    port 636
    secure 1
    base_dn DC=servali,DC=local
    bind_dn CN=Administrateur,CN=Users,DC=servali,DC=local
    bind_pw "Progtr00"
    capath none
    user_attr sAMAccountName
```

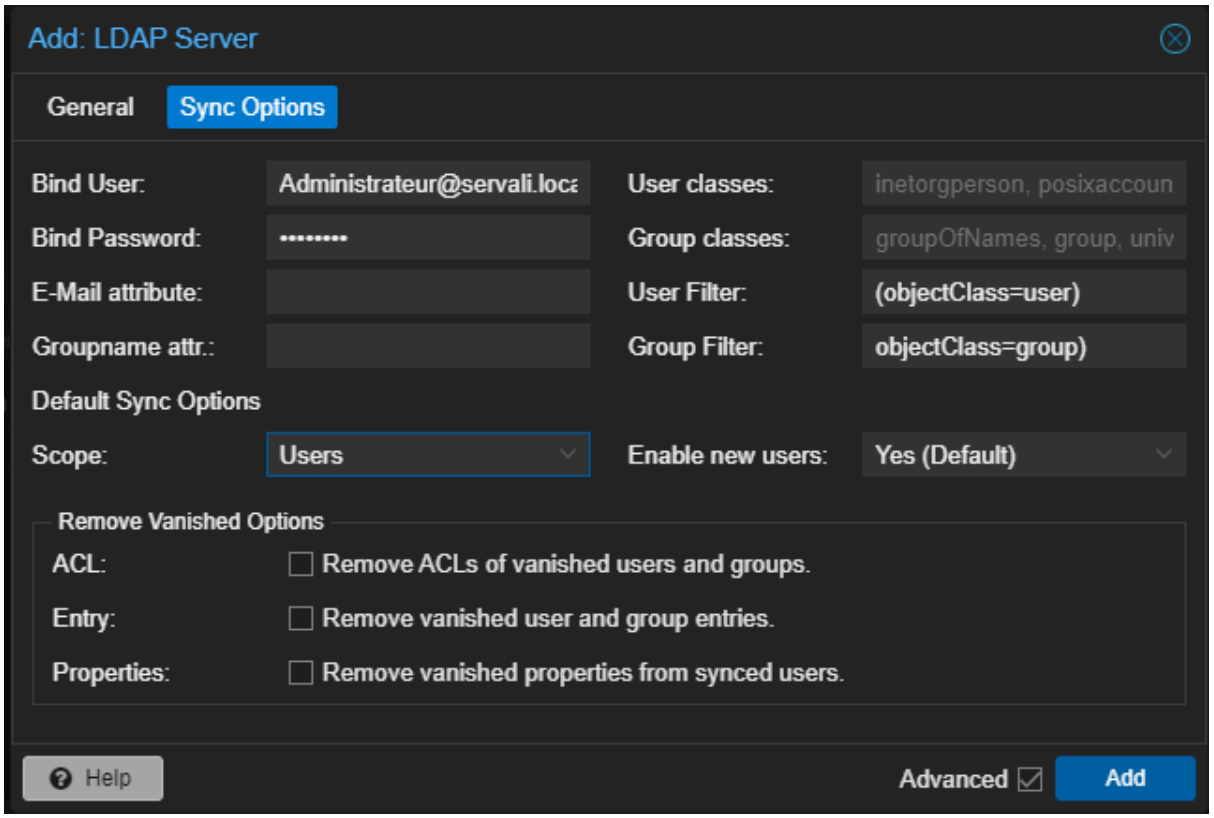
redémarrer le service :

```
systemctl restart pveproxy
```

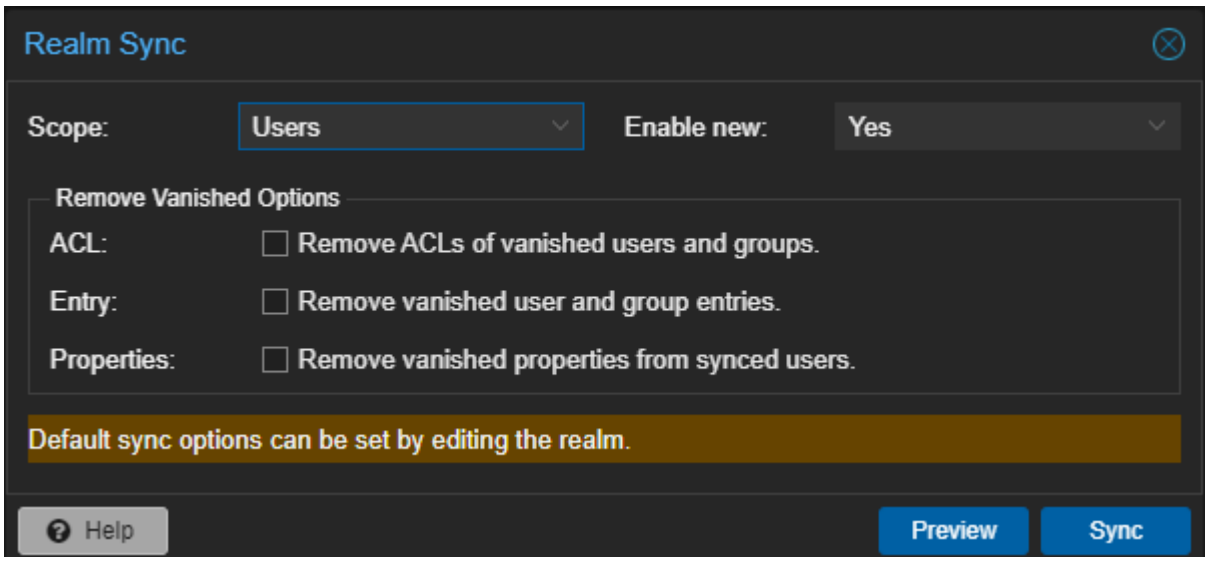
RT3-FI

Maintenant se rendre sur l'interface web du proxmox dans Datacenter > Permissions > Realms, faire "Add" et mettre ces informations :



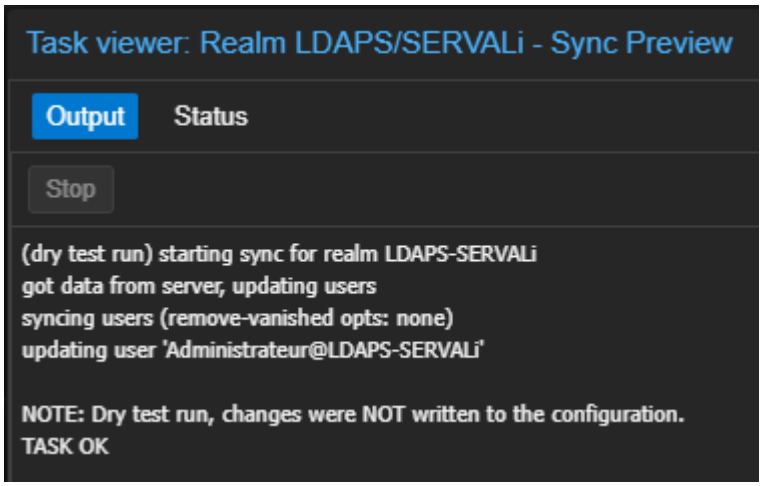


Une fois le realm créé, on fait un clic gauche sur ce nouveau realm puis on clique sur "Sync":



On met “Users” en scope pour ne récupérer que les utilisateurs.

Puis on fait “Preview” pour prévisualiser les utilisateurs qui seront ajoutés à Proxmox :



Si les utilisateurs à ajoutés sont corrects, on peut retourner en arrière et faire “Sync” pour ajouter les utilisateurs du LDAP au Proxmox.

RT3-FI

Interface web et API nodeJS

Pour **déployer une interface web** permettant l'accès à distance aux machines virtuelles nous avons **développé un site web** qui utilise **l'api de proxmox** pour **récupérer, créer, supprimer, cloner** des machines virtuelles.

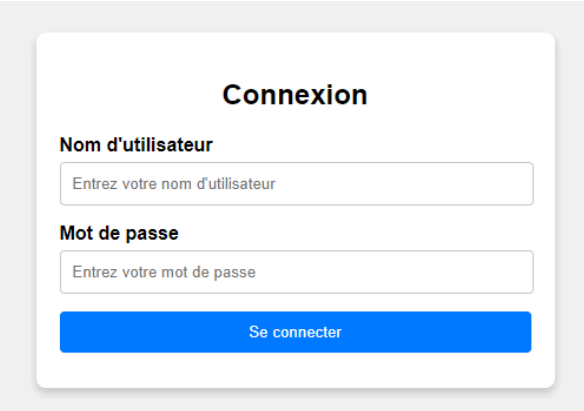
[Les différents codes du site web et de l'api NodeJS se trouveront en annexe à la fin de ce compte rendu à partir de la page 39]

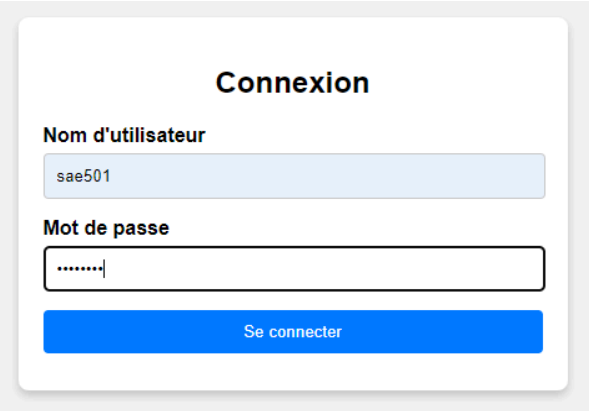
Je tiens à préciser que le développement de cette interface web à en partie été fait à l'aide de chatGPT pour tout ce qui touche au Javascript pour faute de ne pas connaître suffisamment ce langage.

Pour lancer l'api nodeJS et le site web il faut avoir nodeJS d'installer et lancer le fichier app.js avec la commande `node app.js`.

Le site web sera donc disponible à l'adresse **localhost:4000**.

Une fois sur le site web on arrive sur la page de **Login**, sur cette page il suffit d'entrer un identifiant et un mot de passe correct pour pouvoir accéder au site web.





Une fois connecté, l'utilisateur arrive directement sur son **dashboard** sur lequel **il voit les machines virtuelles (VM) auxquelles il a accès**, cette page dashboard met aussi à disposition de l'utilisateur les différentes **fonctionnalités de création, suppression et clonage de VM** dans la barre de navigation (navbar).

La récupération des Machines Virtuelles

La récupération des machines virtuelles est faite en fonction du **groupe** attribué à l'utilisateur connecté. Par exemple, si un utilisateur à le **groupe "sae"**, alors notre api nodeJS va récupérer toutes les machines virtuelles qui ont un **tag "sae"** et les afficher pour cet utilisateur. Pour pouvoir **récupérer toutes les machines virtuelles** disponibles sur le proxmox l'utilisateur doit avoir le **groupe "all"**.

Les cartes des Machines Virtuelles

Que retrouve-t-on sur la carte d'une machine virtuelle ?

Chaque carte affiche l'état de la VM (démarrée ou arrêtée), son ID, la ram (Memory), le nombre de cœurs et la taille du stockage du disque de la VM.

Démarrer une VM

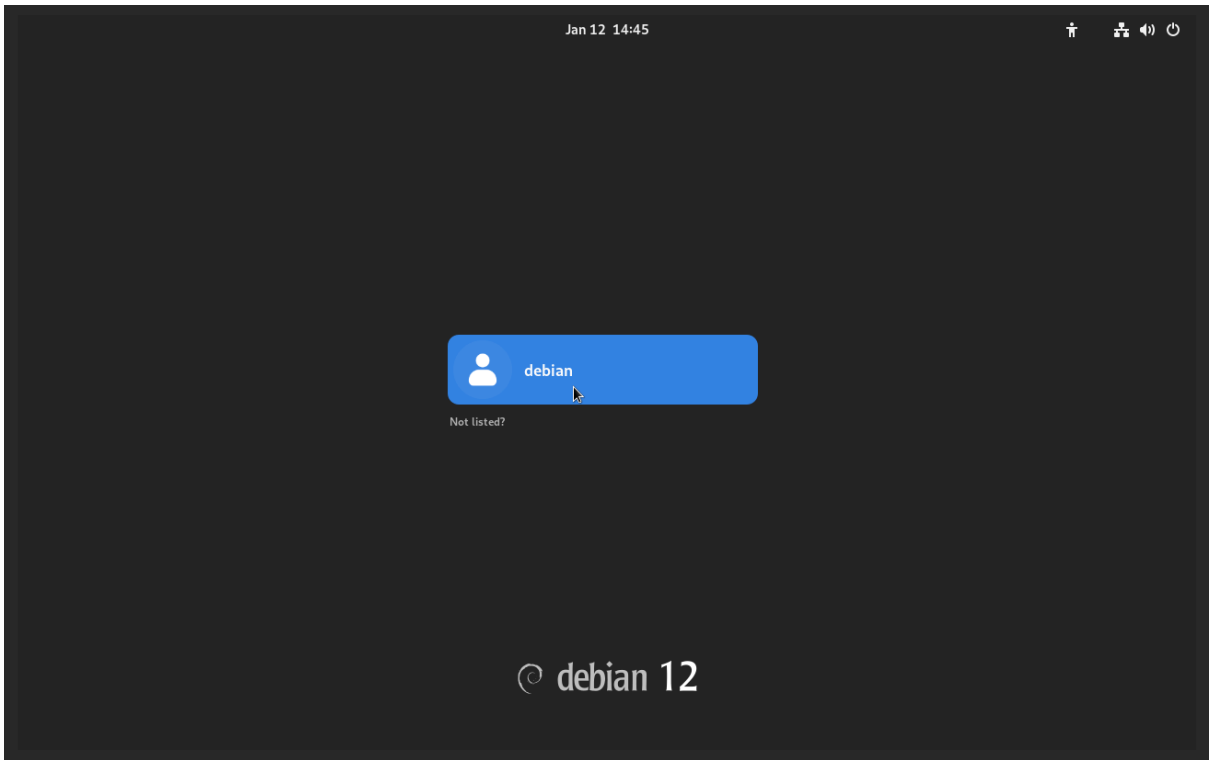
Il y a aussi 3 boutons à disposition, le premier bouton en vert permet de démarrer une VM :

Arrêter une VM

Le 2e bouton permet de l'arrêter :

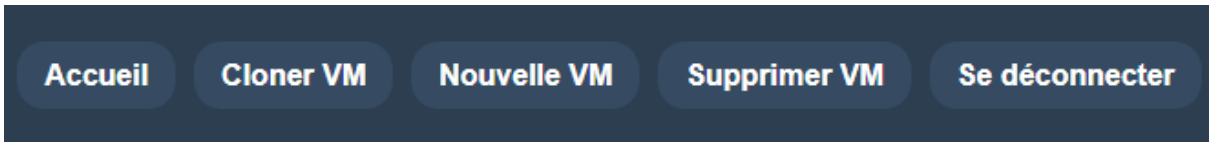
Afficher l'écran d'une VM

et le 3e bouton permet d'ouvrir un nouvel onglet qui affiche l'écran de la VM :



Cloner une VM

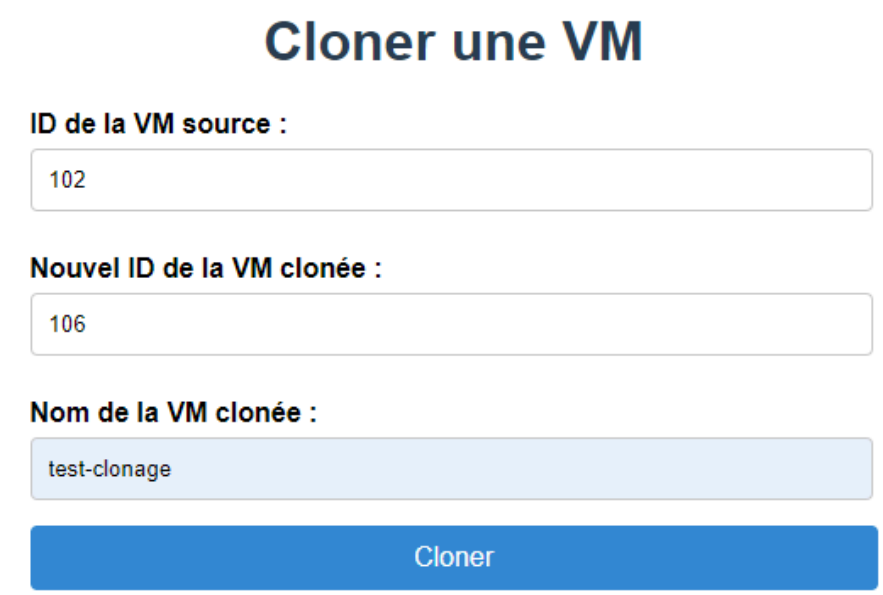
Pour cloner une VM il suffit de cliquer sur le bouton "Cloner VM" :



On arrivera sur cette page :

Il suffit de renseigner l'id de la vm que l'on veut cloner et le nom de la nouvelle vm, l'id de la nouvelle vm est ajouté automatiquement.

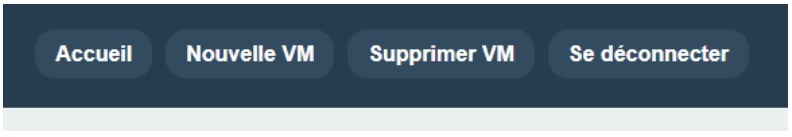
Une fois cela fait :



On peut cliquer sur cloner et on sera renvoyé sur le dashboard et la vm clonée apparaîtra quelques secondes après.

Créer une nouvelle VM

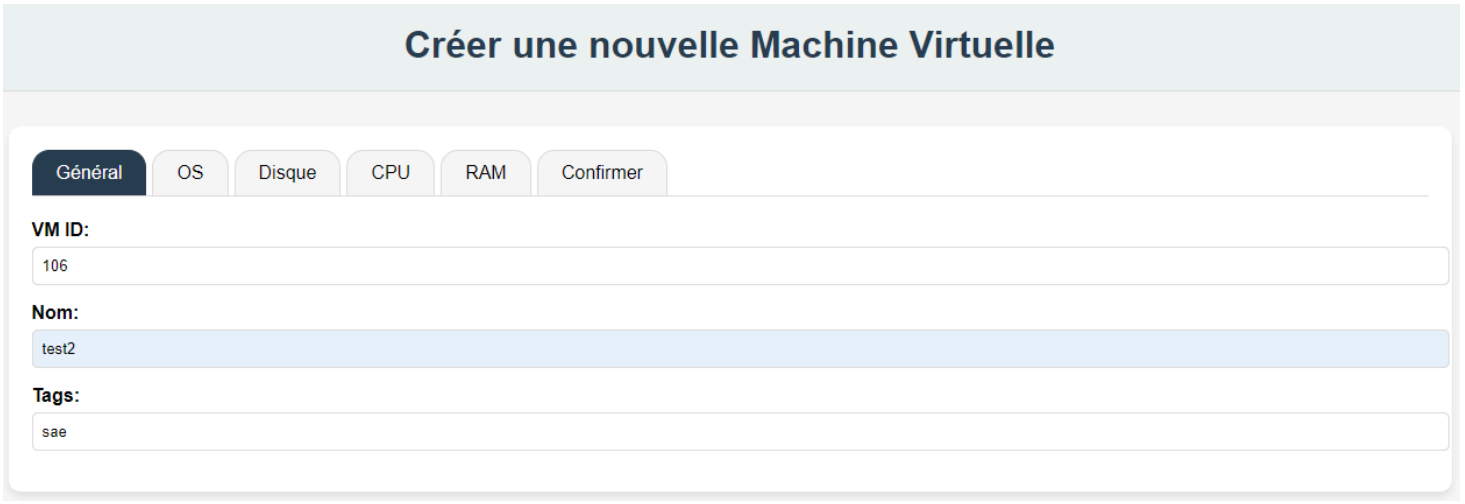
Pour créer une nouvelle VM il suffit de cliquer sur le bouton "Nouvelle VM" :



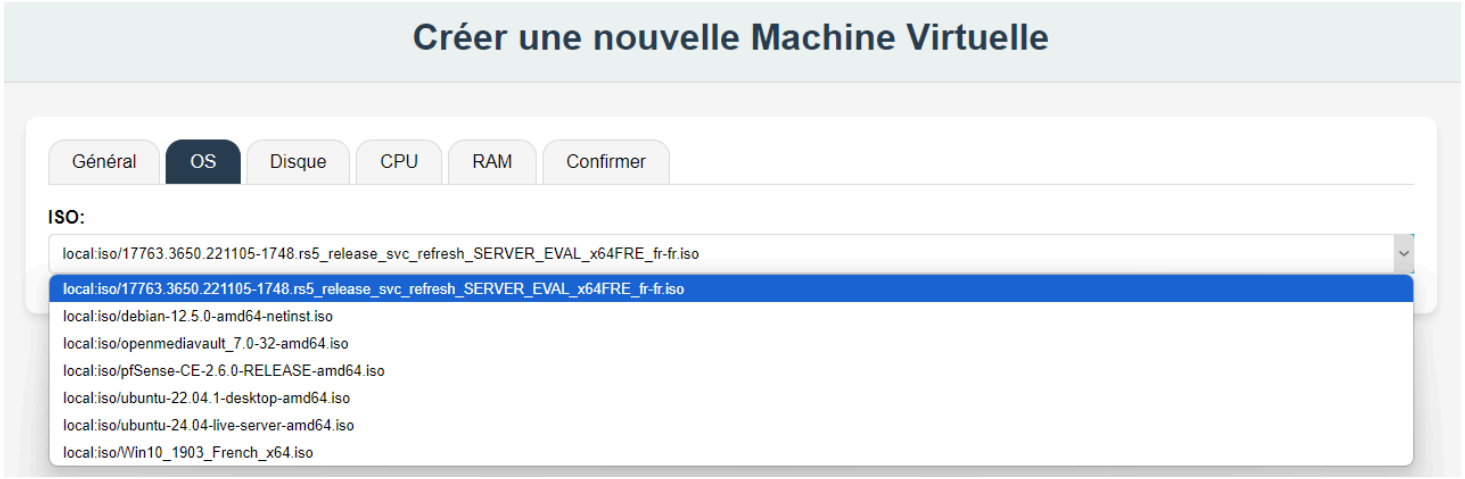
RT3-FI

On arrive sur cette page :

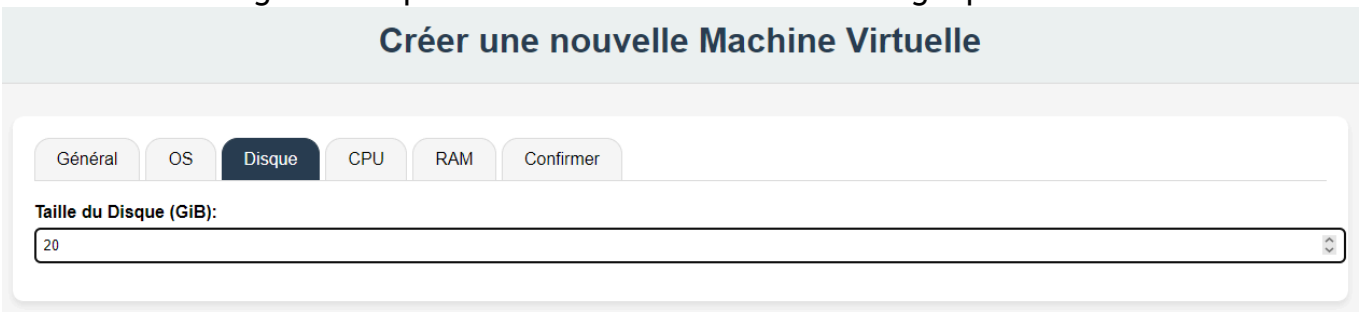
Il suffit de renseigner dans la catégorie “Général” un nom pour la nouvelle VM et un tag (qui n’est pas obligatoire) :



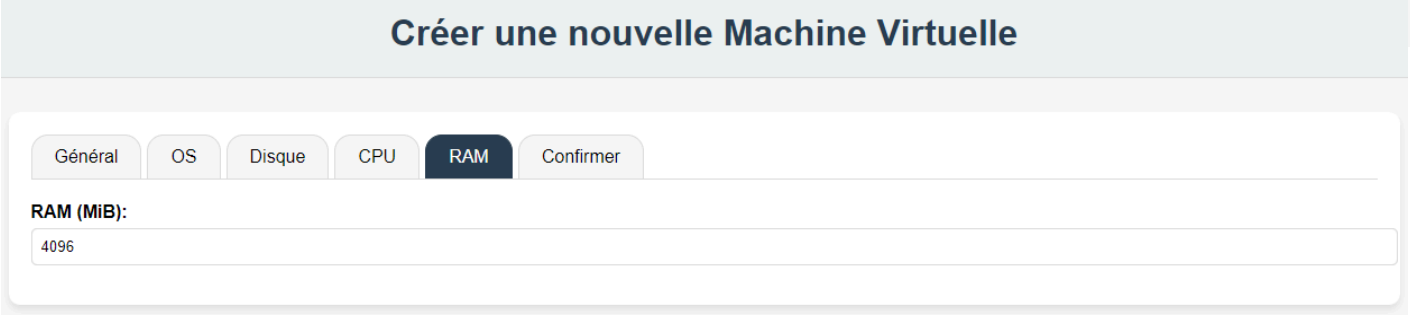
Dans la catégorie “OS” il suffit de choisir l’iso que l’on souhaite mettre sur le VM:



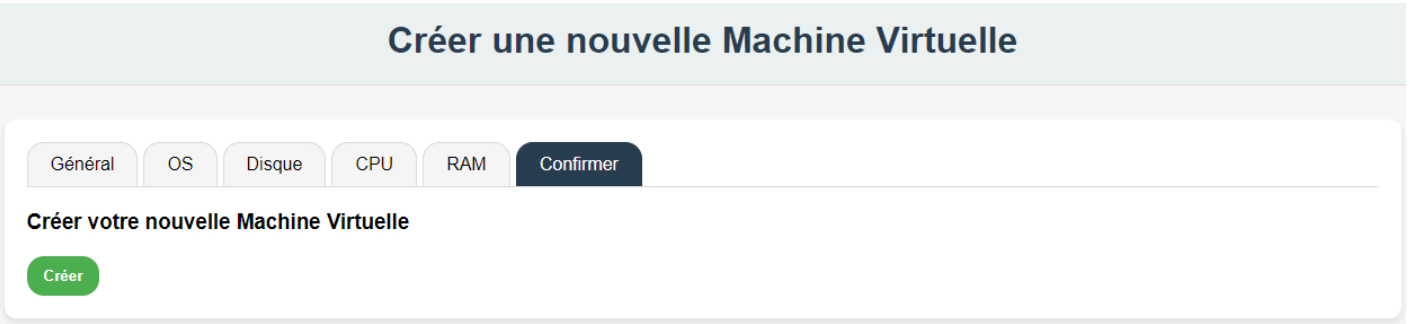
Dans la catégorie “Disque” il faut mettre la taille du stockage que le l’on veut allouer

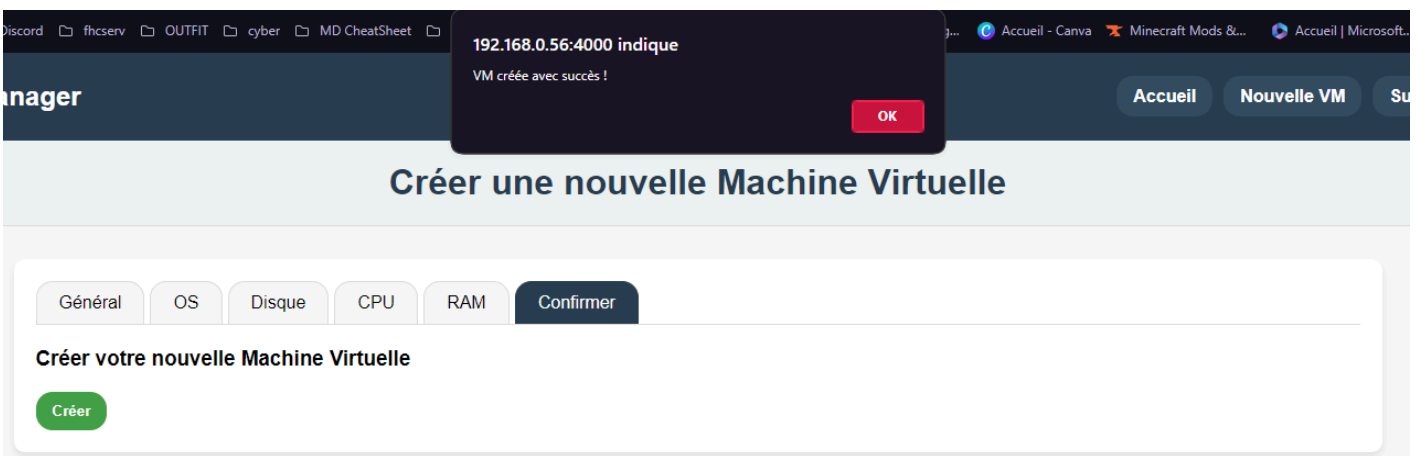


Dans la catégorie “CPU” il faut renseigner le nombre de coeurs et de sockets :



Puis il suffit de cliquer sur créer pour finaliser la création de la VM :

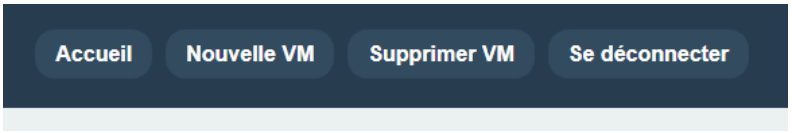




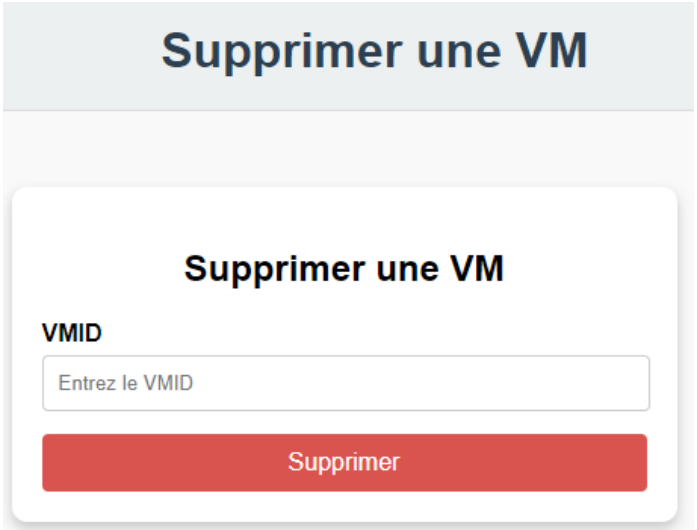
On est donc renvoyés ensuite sur le dashboard et en fonction du tag mis à la machine si il correspond au rôle de l'utilisateur alors celui-ci voit la nouvelle VM sur son dashboard :

Supprimer une VM

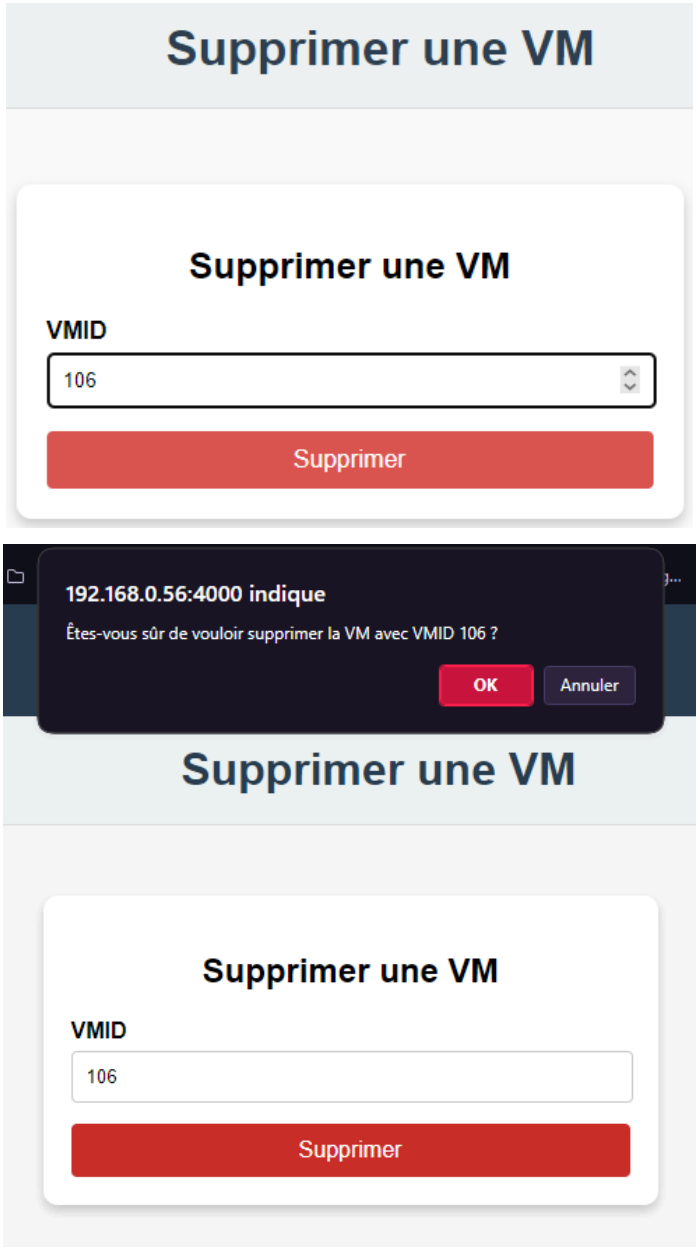
Pour supprimer une machine virtuelle il suffit de cliquer sur le bouton "Supprimer VM" :



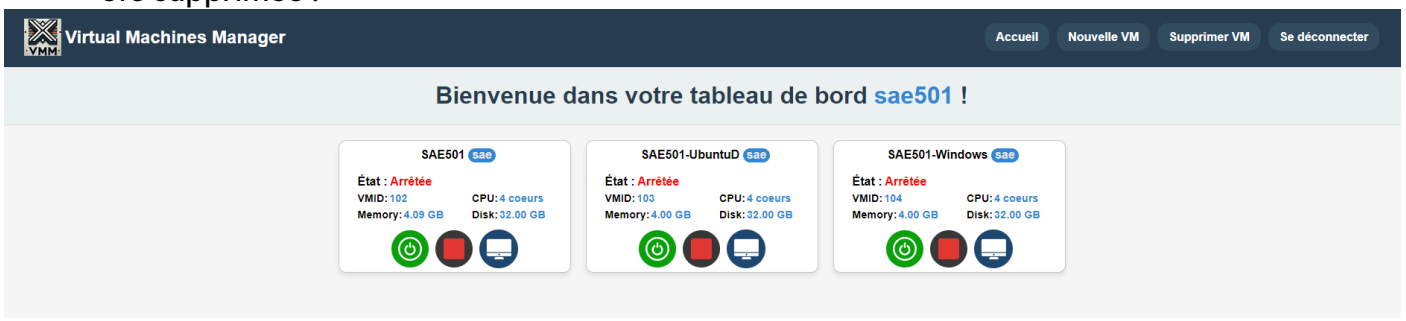
On est donc renvoyé vers cette page où il suffit de rentrer le VMID de la VM que l'on veut supprimer :



RT3-FI



Une fois la VM supprimé on est renvoyé sur le dashboard et on voit bien que la VM a été supprimée :



RT3-FI

Développement durable ?

Nous sommes constamment portés à nous poser la question de manière plus en plus fréquente quant à notre empreinte écologique, celle-ci fait même partie intégrante des politiques d'entreprises de nos jours. Nous sommes donc confrontés à ces problèmes importants jusqu'à notre travail et nos projets. Ici dans le cadre de cette SAE, nous avons été amenés à travailler avec des hyperviseurs, donc des machines qui permettent de virtualiser à grande échelle et uniformément des machines. Le lien entre les hyperviseurs et l'empreinte carbone est que les hyperviseurs permettent d'optimiser et réduire les ressources nécessaires à une machine, il faut prendre en compte par exemple qu'en virtualisant toute une salle de travail(15 postes par exemple) sur un hyperviseur, les coûts en ressources seraient beaucoup moins élevés et permettraient de réduire la consommation d'énergie. De plus, en virtualisant, on réduit la création d'ordinateurs, la production, l'emballage et la transportation de ceux-ci, ce qui permettrait de réduire indirectement la production et l'énergie à produire pour construire ces équipements. Ils permettent aussi une flexibilité qui s'adapterait à tous les besoins grâce à leurs configurations flexibles.

Pour appuyer notre point, nous pouvons citer que leurs utilisation s'aligne avec les objectifs du développement durable pour 2030, on peut citer les OOD 7 et 13 (**ODD7 - Garantir l'accès de tous à des services énergétiques fiables, durables et modernes, à un coût abordable & ODD13 - Prendre d'urgence des mesures pour lutter contre les changements climatiques et leurs répercussions**)

En somme, les hyperviseurs jouent un rôle clé dans le développement durable en optimisant l'utilisation des ressources matérielles, en réduisant l'empreinte carbone grâce à la consolidation des serveurs, en prolongeant la durée de vie du matériel pour limiter les déchets électroniques, et en soutenant des infrastructures cloud plus efficaces et flexibles. Ainsi, ils contribuent à une gestion plus responsable et éco-responsable des technologies informatiques.

Ces question sur le développement durable peuvent nous ouvrir la porte vers une ouverture sur l'évolution de technologies durables dont on peut constater l'évolution, comme avec l'avènement du cloud computing et de la centralisation des services qui permettent de limiter et unifier les coûts ce qui s'aligne parfaitement dans le cadre du développement durable. On peut citer Azure AD et la démocratisation des Hyperviseurs en entreprises et aux universités.

Conclusion

Durant ce projet, nous avons été confrontés à une problématique claire mais peu précise, similaire à ce que l'on pourrait rencontrer en entreprise. Bien que le cadre initial manquait de précision, cela nous a obligés à faire preuve de minutie, de rigueur et à justifier chacun de nos choix. À chaque étape, nous avons dû nous interroger sur la pertinence de nos décisions, ce qui a renforcé notre capacité à réfléchir de manière critique et structurée.

Ce projet nous a permis de mobiliser un large panel de compétences acquises tout au long de notre formation. En plus de nous plonger dans un cadre proche du monde professionnel, il a consolidé nos bases en réseaux et en administration système. J'ai été agréablement surpris par l'ampleur du projet, qui nous a demandé un investissement total et nous a permis de développer une réelle autonomie dans nos démarches. J'ai beaucoup appris, tant sur le plan de l'organisation que sur l'acquisition de nouvelles compétences.

Pour conclure, ce projet a été bénéfique à plusieurs niveaux : professionnel, technique et personnel. Il nous a permis de transformer un besoin simple en un projet concret, tout en nous incitant à une réflexion constante sur nos choix et nos méthodes. Nous avons touché à divers domaines, tels que le codage, la configuration système, le back-end et l'administration système. Enfin, ce projet représente, à mon avis, une concrétisation de notre formation en nous poussant à acquérir une méthodologie solide et à perfectionner nos compétences.

Annexe

app.js

```
// Importation des modules
const express = require('express');
const axios = require('axios');
const cors = require('cors');
const https = require('https');
const fs = require('fs');
const cookieParser = require('cookie-parser');
const fetch = require('node-fetch');
const path = require('path');

const app = express();
const PORT = 4000;

// Charger le certificat auto-signé pour HTTPS
const httpsOptions = {
  key: fs.readFileSync('key.pem'),
  cert: fs.readFileSync('cert.pem'),
};

// Middleware
app.use(cors({
  origin: 'https://192.168.0.56', // Origine autorisée (frontend)
  credentials: true, // Autoriser les cookies
}));
app.use(express.json()); // Permet de traiter les requêtes avec un corps JSON
app.use(cookieParser()); // Middleware pour gérer les cookies

// Agent HTTPS pour accepter les certificats auto-signés
const httpsAgent = new https.Agent({
  rejectUnauthorized: false, // Accepter les certificats non validés
});

// Rediriger la route de base vers la page de connexion
app.get('/', (req, res) => {
  res.redirect('/login');
});

// Servir les fichiers statiques (HTML, CSS, JS)
app.use(express.static(path.join(__dirname, 'public')));

// Routes pour servir les pages HTML
app.get('/login', (req, res) => {
  res.sendFile(path.join(__dirname, 'public', 'login.html'));
});

app.get('/dashboard', (req, res) => {
  res.sendFile(path.join(__dirname, 'public', 'index.html'));
});
```

RT3-FI

```
app.get('/clone-vm', (req, res) => {
  res.sendFile(path.join(__dirname, 'public', 'clone_vm.html'));
});

app.get('/new-vm', (req, res) => {
  res.sendFile(path.join(__dirname, 'public', 'new_vm.html'));
});

app.get('/del-vm', (req, res) => {
  res.sendFile(path.join(__dirname, 'public', 'del_vm.html'));
});

// Route d'authentification
app.post('/authenticate', async (req, res) => {
  const { username, password } = req.body;

  const PROXMOX_API_URL =
'https://192.168.0.56:8006/api2/json/access/ticket';

  try {
    const response = await fetch(PROXMOX_API_URL, {
      method: 'POST',
      headers: { 'Content-Type': 'application/x-www-form-urlencoded' },
      body: new URLSearchParams({ username, password }),
      agent: new https.Agent({ rejectUnauthorized: false }),
    });

    if (response.ok) {
      const data = await response.json();

      // Créer un cookie contenant les identifiants encodés en base64
      const credentials =
Buffer.from(`${username}:${password}`).toString('base64');
      res.cookie('userCredentials', credentials, {
        httpOnly: false, // Empêcher l'accès via JavaScript côté client
        secure: true, // Nécessite HTTPS
        sameSite: 'Strict', // Protéger contre les attaques CSRF
        maxAge: 86400 * 1000, // Valide pendant 1 jour
        path: '/',
      });

      res.status(200).json({ message: 'Connexion réussie', ticket:
data.data.ticket });
    } else {
      const error = await response.json();
      res.status(401).json({ message: 'Authentification échouée', error });
    }
  } catch (error) {
    res.status(500).json({ message: 'Erreur réseau ou serveur', error:
error.message });
  }
});
```

RT3-FI

```
// Route pour récupérer les permissions depuis l'API Proxmox
app.get('/permissions', async (req, res) => {
  const { username, password } = req.query;

  if (!username || !password) {
    return res.status(400).send("Erreur : Identifiant ou mot de passe
manquant.");
  }

  try {
    // Étape 1 : Authentification auprès de Proxmox
    const authResponse = await axios.post(
      'https://192.168.0.56:8006/api2/json/access/ticket',
      new URLSearchParams({ username, password }),
      { httpsAgent }
    );

    const { ticket, CSRFPreventionToken } = authResponse.data.data;

    // Étape 2 : Récupérer les permissions de l'utilisateur
    const permissionsResponse = await axios.get(
      'https://192.168.0.56:8006/api2/json/access/permissions',
      {
        headers: {
          Cookie: `PVEAuthCookie=${ticket}`,
          CSRFPreventionToken,
        },
        httpsAgent,
      }
    );

    permissions = permissionsResponse.data.data;

    // Étape 3 : Retourner les permissions
    res.json(permissions);
  } catch (error) {
    console.error('Erreur lors de la récupération des permissions :',
error.response?.data || error.message);
    console.error('Erreur complète :', error.toJSON ? error.toJSON() :
error);
    res.status(500).json({
      message: 'Une erreur est survenue lors de la récupération des
permissions.',
      error: error.response?.data || error.message,
    });
  }
});

// Route pour récupérer les VMs accessibles par l'utilisateur
app.get('/vms', async (req, res) => {
  const { username, password } = req.query;
```

RT3-FI

```
  if (!username || !password) {
    return res.status(400).send('Erreur : Identifiant ou mot de passe
manquant.');
  }

  try {
    // Authentification utilisateur
    const ticketResponse = await axios.post(
      'https://192.168.0.56:8006/api2/json/access/ticket',
      new URLSearchParams({ username, password }),
      { httpsAgent }
    );

    const ticket = ticketResponse.data.data.ticket;
    const csrfToken = ticketResponse.data.data.CSRFPreventionToken;

    // Ajouter un cookie pour le ticket PVE
    res.cookie('PVEAuthCookie', ticket, {
      path: '/',
      httpOnly: false,
      secure: true,
      maxAge: 1000 * 60 * 60 * 24, // 24 heures
    });

    // Récupérer les groupes de l'utilisateur
    const groupResponse = await axios.get(
      'https://192.168.0.56:8006/api2/json/access/groups',
      {
        headers: {
          Cookie: `PVEAuthCookie=${ticket}`,
          CSRFPreventionToken: csrfToken,
        },
        httpsAgent,
      }
    );

    const groups = groupResponse.data.data;
    const userGroups = groups
      .filter(group => group.users.includes(username))
      .map(group => group.groupid);

    if (!userGroups === 0) {
      console.error("Erreur : Aucun groupe trouvé pour l'utilisateur.");
      return res.status(403).send("Erreur : Accès interdit. Aucun groupe
associé.");
    }

    const hasAccessToAll = userGroups.includes("all");

    // Récupération des ressources (VMs)
    const resourcesResponse = await axios.get(
      "https://192.168.0.56:8006/api2/json/cluster/resources",
      {
```

RT3-FI

```
        headers: {
          Cookie: `PVEAuthCookie=${ticket}`,
          CSRFPreventionToken: csrfToken,
        },
        httpsAgent,
      }
    );

    const allResources = resourcesResponse.data.data;

    let vms;

    if (hasAccessToAll) {
      // Si l'utilisateur a accès à toutes les VMs
      vms = allResources
        .filter(vm => vm.type === "qemu")
        .map(vm => ({
          vmid: vm.vmid,
          name: vm.name,
          maxcpu: vm.maxcpu,
          maxmem: (vm.maxmem / 1024 / 1024 / 1024).toFixed(2),
          maxdisk: (vm.maxdisk / 1024 / 1024 / 1024).toFixed(2),
          status: vm.status,
          tags: vm.tags,
        }));
    } else {
      // Filtrer les VMs en fonction des tags multiples et des groupes de
l'utilisateur
      vms = allResources
        .filter(vm => {
          if (vm.type !== "qemu" || !vm.tags) return false;

          // Diviser les tags multiples de la VM (ex. séparés par ";")
          const vmTags = vm.tags.split(';').map(tag => tag.trim());

          // Vérifier si au moins un des tags correspond à un groupe de
l'utilisateur
          return vmTags.some(tag => userGroups.includes(tag));
        })
        .map(vm => ({
          vmid: vm.vmid,
          name: vm.name,
          maxcpu: vm.maxcpu,
          maxmem: (vm.maxmem / 1024 / 1024 / 1024).toFixed(2),
          maxdisk: (vm.maxdisk / 1024 / 1024 / 1024).toFixed(2),
          status: vm.status,
          tags: vm.tags,
        }));
    }

    res.json(vms);
  } catch (error) {
```

RT3-FI

```
    console.error("Erreur lors de la récupération des VMs :",
error.response?.data || error.message);
    res.status(500).send("Erreur lors de la récupération des données des
VMs");
  }
});

// Route pour démarrer une VM
app.post('/vms/:vmid/start', async (req, res) => {
  const { vmid } = req.params;
  const { username, password } = req.query;

  if (!username || !password) {
    return res.status(400).send('Erreur : Identifiant ou mot de passe
manquant.');
  }

  try {
    const ticketResponse = await axios.post(
      'https://192.168.0.56:8006/api2/json/access/ticket',
      new URLSearchParams({
        username,
        password
      }),
      { httpsAgent }
    );

    const { ticket, CSRFPreventionToken } = ticketResponse.data.data;

    await axios.post(

`https://192.168.0.56:8006/api2/json/nodes/servali/qemu/${vmid}/status/start`
,
      {},
      {
        headers: {
          Cookie: `PVEAuthCookie=${ticket}`,
          CSRFPreventionToken,
        },
        httpsAgent,
      }
    );

    res.status(200).send(`VM ${vmid} démarrée avec succès.`);
  } catch (error) {
    console.error(`Erreur lors du démarrage de la VM ${vmid}:`,
error.message);
    res.status(500).send(`Erreur lors du démarrage de la VM ${vmid}.`);
  }
});

// Route pour arrêter une VM
app.post('/vms/:vmid/stop', async (req, res) => {
```

RT3-FI

```
  const { vmid } = req.params;
  const { username, password } = req.query;

  if (!username || !password) {
    return res.status(400).send('Erreur : Identifiant ou mot de passe
manquant.');
  }

  try {
    const ticketResponse = await axios.post(
      'https://192.168.0.56:8006/api2/json/access/ticket',
      new URLSearchParams({
        username,
        password
      }),
      { httpsAgent }
    );

    const { ticket, CSRFPreventionToken } = ticketResponse.data.data;

    await axios.post(

`https://192.168.0.56:8006/api2/json/nodes/servali/qemu/${vmid}/status/stop`,
      {},
      {
        headers: {
          Cookie: `PVEAuthCookie=${ticket}`,
          CSRFPreventionToken,
        },
        httpsAgent,
      }
    );

    res.status(200).send(`VM ${vmid} arrêtée avec succès.`);
  } catch (error) {
    console.error(`Erreur lors de l'arrêt de la VM ${vmid}:`, error.message);
    res.status(500).send(`Erreur lors de l'arrêt de la VM ${vmid}.`);
  }
});

// Route pour récupérer toutes les Vms
app.get('/all-vms', async (req, res) => {
  const { username, password } = req.query;

  if (!username || !password) {
      return res.status(400).json({ message: 'Identifiant ou mot de passe
manquant.' });
  }

  try {
      const ticketResponse = await axios.post(
          "https://192.168.0.56:8006/api2/json/access/ticket",
          new URLSearchParams({ username, password }),
```

```
          { httpsAgent }
      );

      const ticket = ticketResponse.data.data.ticket;

      const resourcesResponse = await axios.get(
          "https://192.168.0.56:8006/api2/json/cluster/resources",
          {
              headers: {
                  Cookie: `PVEAuthCookie=${ticket}`,
              },
              httpsAgent,
          }
      );

      const vms = resourcesResponse.data.data
          .filter(vm => vm.type === 'qemu')
          .map(vm => ({
              vmid: vm.vmid,
              name: vm.name,
              maxcpu: vm.maxcpu,
              maxmem: (vm.maxmem / 1024 / 1024 / 1024).toFixed(2),
              maxdisk: (vm.maxdisk / 1024 / 1024 / 1024).toFixed(2),
              status: vm.status,
          }));

      res.json(vms);
  } catch (error) {
      console.error('Erreur lors de la récupération des VMs :',
error.response?.data || error.message);
      res.status(500).send('Erreur lors de la récupération des données des
VMs');
  }
});

// Route pour cloner des VMs
app.post('/clone-vm', async (req, res) => {
  const { sourceVmid, newVmid, newVmName } = req.body;
  const { username, password } = req.query;

  if (!username || !password || !sourceVmid || !newVmid || !newVmName) {
    return res.status(400).json({ message: 'Tous les champs sont requis.' });
  }

  try {
    // Étape 1 : Authentification auprès de Proxmox
    const authResponse = await axios.post(
      'https://192.168.0.56:8006/api2/json/access/ticket',
      new URLSearchParams({ username, password }),
      { httpsAgent }
    );

    const { ticket, CSRFPreventionToken } = authResponse.data.data;
```

RT3-FI

```
    // Étape 2 : Envoyer la requête pour cloner la VM
    const cloneResponse = await axios.post(

`https://192.168.0.56:8006/api2/json/nodes/servali/qemu/${sourceVmid}/clone`,
      new URLSearchParams({
        newid: newVmid,
        name: newVmName,
      }),
      {
        headers: {
          Cookie: `PVEAuthCookie=${ticket}`,
          CSRFPreventionToken,
        },
        httpsAgent,
      }
    );

    // Étape 3 : Modifier les tags de la nouvelle VM
    const updateTagsResponse = await axios.put(

`https://192.168.0.56:8006/api2/json/nodes/servali/qemu/${newVmid}/config`,
      new URLSearchParams({
        tags: 'sae', // Remplacer les tags par 'sae'
      }),
      {
        headers: {
          Cookie: `PVEAuthCookie=${ticket}`,
          CSRFPreventionToken,
        },
        httpsAgent,
      }
    );

    res.status(200).json({
      message: `La VM ${newVmName} (ID: ${newVmid}) a été clonée avec succès
et ses tags ont été mis à jour.`,
      cloneData: cloneResponse.data,
      tagUpdateData: updateTagsResponse.data,
    });
  } catch (error) {
    console.error('Erreur lors du clonage de la VM ou de la mise à jour des
tags :', error.response?.data || error.message);
    res.status(500).json({
      message: 'Une erreur est survenue lors du clonage de la VM ou de la
mise à jour des tags.',
      error: error.response?.data || error.message,
    });
  }
});

// Route pour récupérer les iso disponibles sur le proxmox
app.get('/iso-list', async (req, res) => {
```

RT3-FI

```
  const { username, password } = req.query;

  if (!username || !password) {
      return res.status(400).json({ message: 'Identifiant ou mot de passe
manquant.' });
  }

  try {
      const ticketResponse = await axios.post(
          "https://192.168.0.56:8006/api2/json/access/ticket",
          new URLSearchParams({ username, password }),
          { httpsAgent }
      );

      const ticket = ticketResponse.data.data.ticket;

      const isoUrl =
`https://192.168.0.56:8006/api2/json/nodes/servali/storage/local/content`;

      const isoResponse = await axios.get(isoUrl, {
          headers: {
              Cookie: `PVEAuthCookie=${ticket}`,
          },
          httpsAgent,
      });

      const isoFiles = isoResponse.data.data.filter(file => file.content ===
'iso');
      res.status(200).json(isoFiles);
  } catch (error) {
      console.error('Erreur lors de la récupération des ISO :',
error.response?.data || error.message);
      res.status(500).send('Erreur lors de la récupération des ISO.');
  }
});

// Route pour créer une nouvelle VM
app.post('/new-vm', async (req, res) => {
  const { username, password, vmid, name, diskSize, isoImage, sockets, cores,
memory, tags } = req.body;

  if (!username || !password) {
      return res.status(400).json({ message: 'Identifiant ou mot de passe
manquant.' });
  }

  try {
      const authResponse = await axios.post(
          "https://192.168.0.56:8006/api2/json/access/ticket",
          new URLSearchParams({ username, password }),
          { httpsAgent }
      );
```

```
      const { ticket, CSRFPreventionToken } = authResponse.data.data;

      const createVmUrl =
`https://192.168.0.56:8006/api2/json/nodes/servali/qemu`;

      const params = new URLSearchParams({
          vmid,
          name,
          memory,
          sockets,
          cores,
          net0: `virtio,bridge=vmbr0,firewall=1`,
          cpu: `host`,
      });

      if (isoImage) params.append('cdrom', isoImage);
      if (diskSize) params.append('scsi0', `local-lvm:${diskSize}`);
      if (tags && Array.isArray(tags)) {
        params.append('tags', tags.join(','));
      }

      const createVmResponse = await axios.post(createVmUrl, params, {
          headers: {
              Cookie: `PVEAuthCookie=${ticket}`,
              CSRFPreventionToken,
          },
          httpsAgent,
      });

      res.status(200).json({ message: 'VM créée avec succès.', data:
createVmResponse.data });
  } catch (error) {
      console.error('Erreur lors de la création de la VM :',
error.response?.data || error.message);
      res.status(500).send('Erreur lors de la création de la VM.');
  }
});

// Route pour supprimer une VM
app.delete('/vms/:vmid', async (req, res) => {
  const { username, password } = req.query;
  const { vmid } = req.params;

  if (!username || !password) {
    return res.status(400).send("Erreur : Identifiant ou mot de passe
manquant.");
  }

  try {
    const ticketResponse = await axios.post(
      "https://192.168.0.56:8006/api2/json/access/ticket",
      new URLSearchParams({
        username,
```

```
        password,
      }),
      { httpsAgent }
    );

    const { ticket, CSRFPreventionToken } = ticketResponse.data.data;

    const deleteVmResponse = await axios.delete(
      `https://192.168.0.56:8006/api2/json/nodes/servali/qemu/${vmid}`,
      {
        headers: {
          Cookie: `PVEAuthCookie=${ticket}`,
          CSRFPreventionToken,
        },
        httpsAgent,
      }
    );

    if (deleteVmResponse.status === 200) {
      return res.status(200).send(`VM ${vmid} supprimée avec succès.`);
    } else {
      throw new Error(`Erreur lors de la suppression de la VM ${vmid}.`);
    }
  } catch (error) {
    console.error(`Erreur lors de la suppression de la VM ${vmid}:`,
error.message);
    res.status(500).send(`Erreur lors de la suppression de la VM ${vmid}.`);
  }
});

// Route pour se déconnecter
app.post('/logout', (req, res) => {
  res.clearCookie('PVEAuthCookie', { path: '/', domain: '192.168.0.56',
secure: true });
  res.clearCookie('userCredentials', { path: '/', domain: '192.168.0.56',
secure: true });
  res.redirect('/');
});

// Lancer le serveur HTTPS
https.createServer(httpsOptions, app).listen(PORT, () => {
  console.log(`Server running on https://localhost:${PORT}`);
});
```

login.html

```
<!DOCTYPE html>
<html lang="fr">
<head>
```

RT3-FI

```
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Login</title>
<link rel="icon" type="image/x-icon" href="./VMM.ico">
<style>
  body {
    font-family: Arial, sans-serif;
    background-color: #f4f4f4;
    display: flex;
    justify-content: center;
    align-items: center;
    height: 100vh;
    margin: 0;
  }

  .login-container {
    width: 100%;
    max-width: 400px;
    background: #fff;
    padding: 20px;
    box-shadow: 0 4px 8px rgba(0, 0, 0, 0.2);
    border-radius: 8px;
  }

  .login-container h2 {
    text-align: center;
    margin-bottom: 20px;
  }

  .login-container label {
    font-weight: bold;
    display: block;
    margin-bottom: 5px;
  }

  .login-container input {
    width: 95%;
    padding: 10px;
    margin-bottom: 15px;
    border: 1px solid #ccc;
    border-radius: 4px;
  }

  .login-container button {
    width: 100%;
    padding: 10px;
    background: #007bff;
    color: #fff;
    border: none;
    border-radius: 4px;
    cursor: pointer;
  }
```

RT3-FI

```
    .login-container button:hover {
      background: #0056b3;
    }

    .error-message {
      color: red;
      text-align: center;
      margin-top: 10px;
    }
  </style>
</head>
<body>
  <div class="login-container">
    <h2>Connexion</h2>
    <form id="loginForm">
      <label for="username">Nom d'utilisateur</label>
      <input type="text" id="username" placeholder="Entrez votre nom
d'utilisateur" required>
      <label for="password">Mot de passe</label>
      <input type="password" id="password" placeholder="Entrez votre mot de
passe" required>
      <button type="submit">Se connecter</button>
    </form>
    <div id="result" class="error-message"></div>
  </div>

  <script>
    document.getElementById('loginForm').addEventListener('submit', async (e)
=> {
      e.preventDefault();

      const username = document.getElementById('username').value;
      const password = document.getElementById('password').value;

      const tryLogin = async (userWithRealm) => {
        try {
          const response = await fetch('/authenticate', {
            method: 'POST',
            headers: {
              'Content-Type': 'application/json'
            },
            body: JSON.stringify({ username: userWithRealm, password })
          });

          const result = await response.json();
          if (response.ok) {
            sessionStorage.setItem('token', result.ticket);
            document.cookie =
`credentials=${btoa(`${userWithRealm}:${password}`)}; Secure; HttpOnly;
SameSite=Strict; max-age=86400; path=/`;
            window.location.href = '/dashboard';
          } else {
            throw new Error(result.message);
```

```
          }
        } catch (error) {
          return false; // Retourne false si l'authentification échoue
        }
      };

      // Essayer avec @pam d'abord, puis avec @pve si échec
      const loginWithPam = await tryLogin(`${username}@pam`);
      if (!loginWithPam) {
        const loginWithPve = await tryLogin(`${username}@pve`);
        if (!loginWithPve) {
          const loginWithLdaps = await tryLogin(`${username}@LDAPS-SERVALi`);
          if (!loginWithLdaps) {
            document.getElementById('result').innerText = 'Erreur :
Identifiants invalides.';
          }
        }
      }
    });
  </script>
</body>
</html>
```

index.html

```
<!DOCTYPE html>
<html lang="fr">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Tableau de bord</title>
  <link rel="stylesheet"
href="https://pro.fontawesome.com/releases/v5.12.0/css/all.css">
  <link rel="icon" type="image/x-icon" href="./VMM.ico">
  <style>
    /* Styles généraux */
    body {
      font-family: Arial, sans-serif;
      margin: 0;
      padding: 0;
      background-color: #f9f9f9;
    }

    /* Navbar Styles */
    .modern-navbar {
      display: flex;
      justify-content: space-between;
      align-items: center;
      padding: 15px 30px;
      background-color: #2c3e50;
      color: white;
      box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);
```

RT3-FI

```
      flex-wrap: wrap; /* Permettre le passage à la ligne pour les petits
écrans */
    }

    .modern-navbar .logo {
      font-size: 1.5em;
      font-weight: bold;
      flex: 1 1 auto;
    }

    .modern-navbar .nav-links {
      list-style-type: none;
      margin: 0;
      padding: 0;
      display: flex;
      flex-wrap: wrap; /* Empiler les liens sur les petits écrans */
      gap: 10px;
      justify-content: flex-end; /* Aligner les liens à droite */
      flex: 1 1 auto;
    }

    .modern-navbar .nav-links li {
      display: inline;
    }

    .modern-navbar .nav-links a {
      text-decoration: none;
      font-weight: bold;
      color: white;
      padding: 10px 15px;
      border-radius: 15px;
      background-color: #364b61;
      transition: background-color 0.3s ease;
      text-align: center;
    }

    .modern-navbar .nav-links a:hover {
      background-color: #2c3e50;
    }

    #logout {
      background-color: #991818;
    }

    #logout:hover {
      background-color: #bb2121;
    }

    /* Welcome Section */
    .welcome {
      text-align: center;
      padding: 20px;
      background-color: #ecf0f1;
```

RT3-FI

```
    border-bottom: 1px solid #dcdcdc;
}

.welcome h1 {
    margin: 0;
    font-size: 2em;
    color: #2c3e50;
}

/* Container Styles */
.container {
    display: flex;
    gap: 150px;
    padding: 20px;
}

.column1 {
    width: 340px;
    padding-left: 20px;
    padding-bottom: 20px;
    display: flex;
    flex-direction: column;
    gap: 20px;
    border-radius: 25px;
    background-color: #e7e7e7;
    border: 0.01px solid rgba(0, 0, 0, 0.2);
}

.column1 h1 {
    text-align: center;
    margin-bottom: 10px;
}

.column2 {
    flex: 1;
    display: flex;
    flex-direction: column;
    gap: 20px;
}

.column2 h1 {
    text-align: center;
    margin-bottom: 10px;
}

.templategap {
    display: flex;
    flex-direction: column;
    gap: 10px;
}

.othervms {
    display : flex;
```

RT3-FI

```
    flex-wrap: wrap;
    gap: 10px;
}

.card {
    position: relative;
    width: 300px;
    border: 1px solid #ccc;
    border-radius: 10px;
    box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
    background-color: white;
    overflow: hidden;
    transition: transform 0.2s ease, box-shadow 0.2s ease;
    padding: 10px;
    display: flex;
    flex-direction: column;
}

.card:hover {
    transform: translateY(-5px);
    box-shadow: 0 6px 12px rgba(0, 0, 0, 0.15);
}

.card a {
    text-decoration: none;
    color: inherit;
    display: block;
    padding: 20px;
}

.card p {
    margin: 0;
}

.card .small {
    font-size: 0.9em;
    color: black;
    font-weight: bold;
}

#vmName {
    text-align: center;
}

#vmState {
    margin-bottom: 5px;
    margin-left: 15px;
}

#vmCharacteristics {
    margin-left: 15px;
}
```

RT3-FI

```
    .characteristics {
      display: grid;
      grid-template-columns: auto auto; /* Deux colonnes pour aligner les
éléments */
      column-gap: 5px; /* Espacement horizontal entre les colonnes */
      row-gap: 5px; /* Espacement vertical entre les lignes */
      font-size: 0.9em; /* Ajuster la taille si nécessaire */
      line-height: 1.2; /* Espacement entre les lignes pour améliorer la
lisibilité */
    }

    .characteristics div {
      display: flex;
      justify-content: flex-start; /* Ajuste l'espacement entre le label et
la valeur */
      gap: 2px;
      font-weight: bold;
    }

    .tag {
      background:#328bd4;
      border: none;
      border-radius: 15px;
      padding: 0 5px 2px 5px;
      color: rgb(231, 231, 231);
    }

    .tag.template {
      background: #8e44ad; /* Couleur violette */
      color: white; /* Texte en blanc pour un bon contraste */
    }

    .actions {
      margin-top: 10px;
      display: flex;
      justify-content: space-between; /* Crée un espacement uniforme entre
les éléments */
      align-items: center; /* Aligne les éléments verticalement */
    }

    .button {
      background: none;
      border: none;
      cursor: pointer;
      padding: 5px;
      width: 50px; /* Largeur fixe */
      height: 50px; /* Hauteur fixe */
      display: flex;
      justify-content: center;
      align-items: center; /* Centrage de l'image */
      border-radius: 50%; /* Boutons ronds */
      margin-bottom: 5px;
    }
```

```
.button img {
  width: 80%; /* Ajuster la taille de l'image */
  height: 80%; /* Ajuster la taille de l'image */
  object-fit: contain;
}

.button img:hover {
  transform: scale(1.1);
}

.button1 {
  background-color: rgb(12, 160, 12);
  border-radius: 50%;
  margin-left: 60px;
}

.button2 {
  background-color: rgb(56, 56, 56);
  border-radius: 50%;
}

.button3 {
  width: 12px;
  height: 12px;
  background-color: #204972;
  border-radius: 50%;
  margin-right: 70px;
  margin-bottom: 5px;
}

.button3 img {
  width: 35px;
  height: 35px;
  transform: translate(-11px, -11px);
}

.button3 img:hover {
  width: 40px;
  height: 40px;
  transform: translate(-13px, -13px);
}

/* Responsive Styles */
@media (max-width: 885px) {
  .modern-navbar {
    flex-direction: column; /* Empiler les éléments */
    align-items: flex-start;
  }

  .modern-navbar .nav-links {
    justify-content: flex-start;
    margin-top: 15px;
```

```
      }

      .modern-navbar .nav-links li {
        margin-bottom: 15px;
      }

      .modern-navbar .nav-links a {
        width: 100%; /* Chaque lien prend toute la largeur disponible */
      }
    }
  </style>
</head>
<body>
  <!-- Vérification du token -->
  <script>
    if (!sessionStorage.getItem('token')) {
      window.location.href = '/';
    }
  </script>

  <nav class="modern-navbar">
    <img src="./img/VMM.png" alt="VMM" style="width: 50px; margin-right:
5px;">
    <div class="logo">Virtual Machines Manager</div>
    <ul class="nav-links">
      <li><a href="">Accueil</a></li>
      <li><a
href="https://www.canva.com/design/DAGdy8ez_i0/8oSJ8xHat-2Lvol-OGj4XQ/view?ut
m_content=DAGdy8ez_i0&utm_campaign=designshare&utm_medium=link2&utm_source=un
iquelinks&utlId=hd46a505919#13" target="_blank">Tuto</a></li>
      <li><a href="/clone-vm">Cloner VM</a></li>
      <li><a href="/new-vm">Nouvelle VM</a></li>
      <li><a href="/del-vm">Supprimer VM</a></li>
      <li><a href="#" id="logout">Se déconnecter</a></li>
    </ul>
  </nav>

  <div class="welcome">
    <h1 id="welcome-user">Bienvenue dans votre tableau de bord</h1>
  </div>

  <div class="container">
    <div class="column1">
      <h1>Templates</h1>
      <div id="templateColumn" class="templategap"></div>
    </div>
    <div class="column2">
      <h1>VMs</h1>
      <div id="otherVmsColumn" class="othervms"></div>
    </div>
  </div>

  <script>
```

RT3-FI

```
    async function fetchVMs() {
      // Fonction pour récupérer les cookies
      function getCookie(name) {
          const value = `; ${document.cookie}`;
          const parts = value.split(`; ${name}=`);
          if (parts.length === 2) return parts.pop().split(';').shift();
          return null;
      }

      let username, password;

      try {
        const userCredentials = getCookie('userCredentials');
        if (!userCredentials) throw new Error('Cookie userCredentials
introuvable');

        const decodedUrl = decodeURIComponent(userCredentials);
         const decodedCredentials = atob(decodedUrl);
        [username, password] = decodedCredentials.split(':');

        if (!username || !password) throw new Error('Identifiant ou mot de
passe manquant');
      } catch (error) {
        console.error('Erreur lors du décodage du cookie userCredentials :',
error);
        alert('Erreur avec le cookie d\'authentification. Veuillez vous
reconnecter.');
        window.location.href = '/login';
        return;
      }

      async function fetchPerms() {
        try {

          // Récupérer les permissions de l'utilisateur
          const response = await
fetch(`/permissions?username=${encodeURIComponent(username)}&password=${encod
eURIComponent(password)}`);
          if (!response.ok) throw new Error('Erreur lors de la récupération
des permissions.');

          const permissions = await response.json();

          // Fonction pour vérifier si "VM.Clone" existe dans n'importe
quelle section
          function hasVmClone(permissionsData) {
            if (!permissionsData) {
              return false;
            }

            const vmsSection = permissionsData['/vms'];

            if (vmsSection && vmsSection['VM.Clone'] === 1) {
```

RT3-FI

```
              return true;
            }
            return false;
          }

          function hasVmCreationDeletion(permissionsData) {
            if (!permissionsData) {
              console.log("Les permissions sont nulles ou non définies.");
              return false;
            }

            const permissionsToCheck = [
              "VM.Allocate",
              "VM.Config.CDROM",
              "VM.Config.CPU",
              "VM.Config.Cloudinit",
              "VM.Config.Disk",
              "VM.Config.HWType",
              "VM.Config.Memory",
              "VM.Config.Network",
              "VM.Config.Options",
            ];

            const vmsSection = permissionsData["/vms"];

            if (vmsSection) {
              // Vérifier si la section `/vms` contient toutes les
permissions requises
              const allPermissionsExist = permissionsToCheck.every(
                (permission) => vmsSection[permission] === 1
              );

              if (allPermissionsExist) {
                return true; // Retourner true si toutes les permissions sont
présentes dans `/vms`
              }
            }

            return false; // Retourner false si `/vms` est absent ou ne
contient pas toutes les permissions
          }

          const userHasVmClone = hasVmClone(permissions);
          const userHasVmCreationDeletion =
hasVmCreationDeletion(permissions);

          // Si l'utilisateur n'a pas la permission "/", cacher certains
éléments
          if (!permissions) {
          // Cacher les liens
            document.querySelector('a[href="/clone-vm"]').style.display =
'none';
```

RT3-FI

```
            document.querySelector('a[href="/new-vm"]').style.display =
'none';
            document.querySelector('a[href="/del-vm"]').style.display =
'none';

            // Cacher la colonne Templates
            const column1 = document.querySelector('.column1');
            if (column1) {
              column1.style.display = 'none';
            }
          }

          if (!userHasVmClone) {
            document.querySelector('a[href="/clone-vm"]').style.display =
'none';

            // Cacher la colonne Templates
            const column1 = document.querySelector('.column1');
            if (column1) {
              column1.style.display = 'none';
            }
          }

          if (!userHasVmCreationDeletion) {
            document.querySelector('a[href="/new-vm"]').style.display =
'none';
            document.querySelector('a[href="/del-vm"]').style.display =
'none';
          }
        } catch (error) {
          console.error('Erreur lors de la vérification des permissions :',
error);
          alert('Impossible de vérifier les permissions de l\'utilisateur.
Veuillez réessayer.');
        }
      }

      fetchPerms();

      try {
        // Construire la requête avec les identifiants de l'utilisateur
        const response = await
fetch(`/vms?username=${encodeURIComponent(username)}&password=${encodeURIComp
onent(password)}`, {
            credentials: 'include'
        });

        if (!response.ok) {
            throw new Error(`Erreur HTTP ${response.status} :
${response.statusText}`);
        }

        const vms = await response.json();
```

RT3-FI

```
        const templateColumn = document.getElementById('templateColumn');
        const otherVmsColumn = document.getElementById('otherVmsColumn');

        templateColumn.innerHTML = '';
        otherVmsColumn.innerHTML = '';

        vms.forEach(vm => {
          const card = document.createElement('div');
          card.classList.add('card');

          const tagsHTML = vm.tags
            ? vm.tags
                .split(';')
                .map(tag => `<strong class="tag ${tag.trim() === 'template' ?
'template' : ''}">${tag.trim()}</strong>`)
                .join(' ')
            : '';

          // Condition pour afficher ou non l'état de la machine
          const stateHTML = vm.tags && vm.tags.includes('template')
            ? '' // Pas d'état pour les VMs avec le tag 'template'
            : `
              <p id="vmState" class="state">
                <strong>État : <span style="color: ${vm.status === 'running'
? 'green' : 'red'};">
                    ${vm.status === 'running' ? 'Démarrée' : 'Arrêtée'}
                </span></strong>
              </p>
            `;

          const actionsHTML = vm.tags && vm.tags.includes('template')
            ? '' // Pas d'actions pour les VMs avec le tag 'template'
            : `
              <div class="actions">
                <button class="button button1" onclick="startVM(${vm.vmid})">
                  <img src="./img/start.png" alt="Démarrer">
                </button>
                <button class="button button2" onclick="stopVM(${vm.vmid})">
                  <img src="./img/stop.png" alt="Arrêter">
                </button>
                <a target="_blank"
href="https://192.168.0.56:8006/?console=kvm&novnc=1&vmid=${vm.vmid}&vmname=$
{vm.name}&node=servali&resize=off&cmd=" class="button3">
                  <img src="./img/screen.png" alt="écran">
                </a>
              </div>
            `;

          card.innerHTML = `
            <div>
              <p id="vmName"><strong>${vm.name}</strong> ${tagsHTML}</p>
              <br>
```

RT3-FI

```
              ${stateHTML}
              <div id="vmCharacteristics" class="characteristics">
                <div>
                  <strong>VMID:</strong> <span
style="color:#328bd4">${vm.vmid}</span>
                </div>
                <div>
                  <strong>CPU:</strong> <span
style="color:#328bd4;">${vm.maxcpu} coeurs</span>
                </div>
                <div>
                  <strong>Memory:</strong> <span
style="color:#328bd4">${vm.maxmem} GB</span>
                </div>
                <div>
                  <strong>Disk:</strong> <span
style="color:#328bd4">${vm.maxdisk} GB</span>
                </div>
              </div>
              ${actionsHTML}
            </div>
          `;

          if (vm.tags && vm.tags.includes('template')) {
            templateColumn.appendChild(card);
          } else {
            otherVmsColumn.appendChild(card);
          }
        });
      } catch (error) {
          console.error('Erreur lors de la récupération des VMs :', error);
          alert("Une erreur est survenue lors de la récupération des machines
virtuelles. Veuillez réessayer.");
      }
    }

    // Rafraîchir automatiquement les statuts des VM toutes les 5 secondes
    setInterval(fetchVMs, 5000);

    // Fonction pour récupérer les cookies
    function getCookie(name) {
      const value = `; ${document.cookie}`;
      const parts = value.split(`; ${name}=`);
      if (parts.length === 2) return parts.pop().split(';').shift();
      return null;
    }

    document.addEventListener('DOMContentLoaded', () => {
      let username;

      try {
        // Récupérer le cookie contenant les identifiants
        const userCredentials = getCookie('userCredentials');
```

RT3-FI

```
        if (!userCredentials) {
          throw new Error('Cookie userCredentials introuvable');
        }

        // Décoder le cookie
        const decodedUrl = decodeURIComponent(userCredentials);
        const decodedCredentials = atob(decodedUrl);

        // Extraire le nom d'utilisateur
        [username] = decodedCredentials.split(':');

        if (!username) {
          throw new Error('Nom d\'utilisateur manquant après décodage');
        }

        // Retirer le suffixe (@pve ou @pam) du nom d'utilisateur
        username = username.split('@')[0];

        // Mettre à jour le message de bienvenue avec le pseudo
        const welcomeElement = document.getElementById('welcome-user');
        welcomeElement.innerHTML = `Bienvenue dans votre tableau de bord
<span style="color: #328bd4;">${username}</span> !`;
      } catch (error) {
        console.error('Erreur lors du décodage du cookie userCredentials :',
error);
        alert('Erreur avec le cookie d\'authentification. Veuillez vous
reconnecter.');
        window.location.href = '/login';
      }
    });

    async function startVM(vmid) {
      let username, password;

      try {
        // Récupérer le cookie contenant les identifiants
        const userCredentials = getCookie('userCredentials');

        if (!userCredentials) {
          throw new Error('Cookie userCredentials introuvable');
        }

        // Décoder le cookie
        const decodedUrl = decodeURIComponent(userCredentials);
        const decodedCredentials = atob(decodedUrl);

        // Extraire le nom d'utilisateur et le mot de passe
        [username, password] = decodedCredentials.split(':');

        if (!username || !password) {
          throw new Error('Identifiant ou mot de passe manquant après
décodage');
```

RT3-FI

```
        }
      } catch (error) {
        console.error('Erreur lors du décodage du cookie userCredentials :',
error);
        alert('Erreur avec le cookie d\'authentification. Veuillez vous
reconnecter.');
        window.location.href = '/login';
        return; // Arrêter l'exécution si le cookie est invalide
      }

      try {
        // Démarrer la VM avec les identifiants
        const response = await
fetch(`/vms/${vmid}/start?username=${encodeURIComponent(username)}&password=$
{encodeURIComponent(password)}`, {
          method: 'POST',
          credentials: 'include',
        });

        if (!response.ok) {
          throw new Error(`Erreur HTTP ${response.status} :
${response.statusText}`);
        }

        alert(`VM ${vmid} démarrée.`);
        fetchVMs(); // Rafraîchir la liste des VMs
      } catch (error) {
        console.error(`Erreur lors du démarrage de la VM ${vmid}:`, error);
      }
    }

    async function stopVM(vmid) {
      let username, password;

      try {
        // Récupérer le cookie contenant les identifiants
        const userCredentials = getCookie('userCredentials');

        if (!userCredentials) {
          throw new Error('Cookie userCredentials introuvable');
        }

        // Décoder le cookie
        const decodedUrl = decodeURIComponent(userCredentials);
        const decodedCredentials = atob(decodedUrl);

        // Extraire le nom d'utilisateur et le mot de passe
        [username, password] = decodedCredentials.split(':');

        if (!username || !password) {
          throw new Error('Identifiant ou mot de passe manquant après
décodage');
        }
```

```
      } catch (error) {
        console.error('Erreur lors du décodage du cookie userCredentials :',
error);
        alert('Erreur avec le cookie d\'authentification. Veuillez vous
reconnecter.');
        window.location.href = '/login';
        return; // Arrêter l'exécution si le cookie est invalide
      }

      try {
        // Arrêter la VM avec les identifiants
        const response = await
fetch(`/vms/${vmid}/stop?username=${encodeURIComponent(username)}&password=${
encodeURIComponent(password)}`, {
          method: 'POST',
          credentials: 'include',
        });

        if (!response.ok) {
          throw new Error(`Erreur HTTP ${response.status} :
${response.statusText}`);
        }

        alert(`VM ${vmid} arrêtée.`);
        fetchVMs(); // Rafraîchir la liste des VMs
      } catch (error) {
        console.error(`Erreur lors de l'arrêt de la VM ${vmid}:`, error);
      }
    }

    // Charger initialement les VMs
    fetchVMs();
  </script>
  <script src="./logout.js"></script>
</body>
</html>
```

clone_vm.html

```
<!DOCTYPE html>
<html lang="fr">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Cloner une VM</title>
  <link rel="stylesheet"
href="https://pro.fontawesome.com/releases/v5.12.0/css/all.css">
  <link rel="icon" type="image/x-icon" href="./VMM.ico">
  <style>
    /* Styles généraux */
    body {
```

RT3-FI

```
    font-family: Arial, sans-serif;
    margin: 0;
    padding: 0;
    background-color: #f9f9f9;
  }

  .modern-navbar {
    display: flex;
    justify-content: space-between;
    align-items: center;
    padding: 15px 30px;
    background-color: #2c3e50;
    color: white;
    box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);
    flex-wrap: wrap;
  }

  .modern-navbar .logo {
    font-size: 1.5em;
    font-weight: bold;
    flex: 1 1 auto;
  }

  .modern-navbar .nav-links {
    list-style-type: none;
    margin: 0;
    padding: 0;
    display: flex;
    flex-wrap: wrap;
    gap: 10px;
    justify-content: flex-end;
    flex: 1 1 auto;
  }

  .modern-navbar .nav-links li {
    display: inline;
  }

  .modern-navbar .nav-links a {
    text-decoration: none;
    font-weight: bold;
    color: white;
    padding: 10px 15px;
    border-radius: 15px;
    background-color: #364b61;
    transition: background-color 0.3s ease;
    text-align: center;
  }

  .modern-navbar .nav-links a:hover {
    background-color: #2c3e50;
  }
```

RT3-FI

```
.clonediv {
  display: flex;
  justify-content: center;
  align-items: center;
  height: 500px;
}

.container {
  width: 100%;
  max-width: 500px;
  background-color: white;
  padding: 20px;
  border-radius: 8px;
  box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
}

h1 {
  text-align: center;
  color: #2c3e50;
}

label {
  font-weight: bold;
  margin-top: 10px;
  display: block;
}

input, select {
  width: 95%;
  padding: 10px;
  margin-top: 5px;
  margin-bottom: 15px;
  border: 1px solid #ccc;
  border-radius: 4px;
}

button {
  width: 100%;
  padding: 10px;
  background-color: #328bd4;
  color: white;
  border: none;
  border-radius: 4px;
  font-size: 16px;
  cursor: pointer;
  transition: background-color 0.3s ease;
}

button:hover {
  background-color: #256fa1;
}

.error-message {
```

RT3-FI

```
      color: red;
      text-align: center;
      margin-top: 10px;
    }

    .success-message {
      color: green;
      text-align: center;
      margin-top: 10px;
    }
  </style>
</head>
<body>
    <nav class="modern-navbar">
        <img src="./img/VMM.png" alt="VMM" style="width: 50px; margin-right:
5px;">
        <div class="logo">Virtual Machines Manager</div>
        <ul class="nav-links">
          <li><a href="/dashboard">Accueil</a></li>
          <li><a href="/clone-vm">Cloner VM</a></li>
          <li><a href="/new-vm">Nouvelle VM</a></li>
          <li><a href="/del-vm">Supprimer VM</a></li>
          <li><a href="#" id="logout">Se déconnecter</a></li>
        </ul>
    </nav>

    <div class="clonediv">
        <div class="container">
            <h1>Cloner une VM</h1>
            <form id="cloneVmForm">
            <label for="sourceVmid">ID de la VM source :</label>
            <input type="number" id="sourceVmid" placeholder="Entrez l'ID de
la VM source" required>

            <label for="newVmid">Nouvel ID de la VM clonée :</label>
            <input type="number" id="newVmid" placeholder="Chargement..."
disabled required>

            <label for="newVmName">Nom de la VM clonée :</label>
            <input type="text" id="newVmName" placeholder="Entrez le nom de
la nouvelle VM" required>

            <button type="submit">Cloner</button>
            </form>
            <div id="result" class="error-message"></div>
        </div>
    </div>

  <script>
    document.addEventListener('DOMContentLoaded', async () => {
        function getCookie(name) {
            const value = `; ${document.cookie}`;
            const parts = value.split(`; ${name}=`);
```

RT3-FI

```
            if (parts.length === 2) return parts.pop().split(';').shift();
            return null;
        }

        let username, password;

        try {
            const userCredentials = getCookie('userCredentials');
            if (!userCredentials) throw new Error('Cookie userCredentials
introuvable');

            const decodedUrl = decodeURIComponent(userCredentials);
            const decodedCredentials = atob(decodedUrl);
            [username, password] = decodedCredentials.split(':');

            if (!username || !password) throw new Error('Identifiant ou mot
de passe manquant');
        } catch (error) {
            console.error('Erreur lors du décodage du cookie userCredentials
:', error);
            alert('Erreur avec le cookie d\'authentification. Veuillez vous
reconnecter.');
            window.location.href = '/login';
            return;
        }

        async function fetchVMID() {
            const newVmidInput = document.getElementById('newVmid');
            try {
                const response = await
fetch(`/all-vms?username=${encodeURIComponent(username)}&password=${encodeURI
Component(password)}`);
                if (!response.ok) throw new Error('Failed to fetch VMs');

                const vms = await response.json();
                const maxVMID = Math.max(...vms.map(vm => vm.vmid), 0);
                newVmidInput.value = maxVMID + 1;
                newVmidInput.disabled = false;
            } catch (error) {
                console.error('Error fetching VMID:', error);
                newVmidInput.placeholder = 'Erreur lors du chargement';
            }
        }

        fetchVMID();

        document.getElementById('cloneVmForm').addEventListener('submit',
async (e) => {
            e.preventDefault();

            const sourceVmid = document.getElementById('sourceVmid').value;
            const newVmid = document.getElementById('newVmid').value;
            const newVmName = document.getElementById('newVmName').value;
```

```
            try {
                const response = await
fetch(`/clone-vm?username=${encodeURIComponent(username)}&password=${encodeUR
IComponent(password)}`, {
                    method: 'POST',
                    headers: { 'Content-Type': 'application/json' },
                    body: JSON.stringify({ sourceVmid, newVmid, newVmName })
                });

                const resultElement = document.getElementById('result');

                if (response.ok) {
                    const data = await response.json();
                    resultElement.className = 'success-message';
                    resultElement.textContent = `VM clonée avec succès :
${data.message}`;
                    window.location.href = '/dashboard';
                } else {
                    const error = await response.json();
                    resultElement.className = 'error-message';
                    resultElement.textContent = `Erreur : ${error.message}`;
                }
            } catch (error) {
                console.error('Erreur lors du clonage de la VM :', error);
                document.getElementById('result').className =
'error-message';
                document.getElementById('result').textContent = 'Une erreur
est survenue lors du clonage de la VM.';
            }
        });
    });
  </script>
  <script src="./logout.js"></script>
</body>
</html>
```

new_vm.html

```
<!DOCTYPE html>
<html lang="fr">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width,
initial-scale=1.0">
  <title>Créer une nouvelle machine virtuelle</title>
  <link rel="stylesheet"
href="https://pro.fontawesome.com/releases/v5.12.0/css/all.css">
  <link rel="icon" type="image/x-icon" href="./VMM.ico">
  <style>
    body {
      font-family: Arial, sans-serif;
```

```
      margin: 0;
      padding: 0;
      background-color: #f9f9f9;
    }

    /* Navbar Styles */
    .modern-navbar {
      display: flex;
      justify-content: space-between;
      align-items: center;
      padding: 15px 30px;
      background-color: #2c3e50;
      color: white;
      box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);
      flex-wrap: wrap; /* Permettre le passage à la ligne pour les
petits écrans */
    }

    .modern-navbar .logo {
      font-size: 1.5em;
      font-weight: bold;
      flex: 1 1 auto;
    }

    .modern-navbar .nav-links {
      list-style-type: none;
      margin: 0;
      padding: 0;
      display: flex;
      flex-wrap: wrap; /* Empiler les liens sur les petits écrans */
      gap: 10px;
      justify-content: flex-end; /* Aligner les liens à droite */
      flex: 1 1 auto;
    }

    .modern-navbar .nav-links li {
      display: inline;
    }

    .modern-navbar .nav-links a {
      text-decoration: none;
      font-weight: bold;
      color: white;
      padding: 10px 15px;
      border-radius: 15px;
      background-color: #364b61;
      transition: background-color 0.3s ease;
      text-align: center;
    }

    .modern-navbar .nav-links a:hover {
```

RT3-FI

```
    background-color: #2c3e50;
  }

  /* Welcome Section */
  .welcome {
    text-align: center;
    padding: 20px;
    background-color: #ecf0f1;
    border-bottom: 1px solid #dcdcdc;
  }

  .welcome h1 {
    margin: 0;
    font-size: 2em;
    color: #2c3e50;
  }

  .container {
    max-width: 1200px;
    margin: 30px auto;
    background: white;
    padding: 20px;
    border-radius: 10px;
    box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
  }

  .tabs {
    display: flex;
    border-bottom: 1px solid #ddd;
    margin-bottom: 20px;
  }

  .tab {
    padding: 10px 20px;
    cursor: pointer;
    border: 1px solid #ddd;
    border-bottom: none;
    border-top-left-radius: 15px;
    border-top-right-radius: 15px;
    background: #f9f9f9;
    margin-right: 5px;
  }

  .tab.active {
    background: #2c3e50;
    color: white;
    border-color: #2c3e50;
  }

  .tab-content {
    display: none;
```

RT3-FI

```
}

.tab-content.active {
  display: block;
}

.form-group {
  margin-bottom: 15px;
}

label {
  display: block;
  margin-bottom: 5px;
  font-weight: bold;
}

input, select {
  width: 100%;
  padding: 8px;
  border: 1px solid #ddd;
  border-radius: 5px;
}

button {
  padding: 10px 15px;
  background-color: #4CAF50;
  border: none;
  color: white;
  border-radius: 15px;
  cursor: pointer;
  font-weight: bold;
  transition: background-color 0.3s;
}

button:hover {
  background-color: #45a049;
}

/* Responsive Styles */
@media (max-width: 885px) {
  .modern-navbar {
    flex-direction: column; /* Empiler les éléments */
    align-items: flex-start;
  }

  .modern-navbar .nav-links {
    justify-content: flex-start;
    margin-top: 15px;
  }

  .modern-navbar .nav-links li {
```

```
        margin-bottom: 15px;
      }

      .modern-navbar .nav-links a {
        width: 100%; /* Chaque lien prend toute la largeur disponible
*/
      }
    }
  </style>
</head>
<body>
    <nav class="modern-navbar">
        <img src="./img/VMM.png" alt="VMM" style="width: 50px;
margin-right: 5px;">
        <div class="logo">Virtual Machines Manager</div>
        <ul class="nav-links">
            <li><a href="/dashboard">Accueil</a></li>
            <li><a href="/new-vm">Nouvelle VM</a></li>
            <li><a href="/del-vm">Supprimer VM</a></li>
            <li><a href="#" id="logout">Se déconnecter</a></li>
        </ul>
    </nav>

    <div class="welcome">
        <h1 id="welcome-user">Créer une nouvelle Machine Virtuelle</h1>
    </div>

  <div class="container">
    <div class="tabs">
      <div class="tab active" data-target="general">Général</div>
      <div class="tab" data-target="os">OS</div>
      <div class="tab" data-target="disks">Disque</div>
      <div class="tab" data-target="cpu1">CPU</div>
      <div class="tab" data-target="memory">RAM</div>
      <div class="tab" data-target="confirm">Confirmer</div>
    </div>

    <form id="createVmForm">
      <!-- General -->
      <div class="tab-content active" id="general">
        <div class="form-group">
          <label for="vmid">VM ID:</label>
          <input type="number" id="vmid" name="vmid" required>
        </div>
        <div class="form-group">
          <label for="name">Nom:</label>
          <input type="text" id="name" name="name" required>
        </div>
        <div class="form-group">
            <label for="tags">Tags:</label>
```

RT3-FI

```
            <input type="text" id="tags" name="tags" placeholder="Enter
tags (comma-separated)">
          </div>
      </div>

      <!-- OS -->
      <div class="tab-content" id="os">
        <div class="form-group">
          <label for="iso-image">ISO:</label>
          <select id="iso-image" name="isoImage" required>
            <!-- Dynamically load ISO options -->
          </select>
        </div>
      </div>

      <!-- Disks -->
    <div class="tab-content" id="disks">
        <div class="form-group">
            <label for="disk-size">Taille du Disque (GiB):</label>
            <input type="number" id="disk-size" name="diskSize"
required>
        </div>
    </div>

      <!-- CPU -->
      <div class="tab-content" id="cpu1">
        <div class="form-group">
          <label for="sockets">Sockets:</label>
          <input type="number" id="sockets" name="sockets" required>
        </div>
        <div class="form-group">
          <label for="cores">Coeurs:</label>
          <input type="number" id="cores" name="cores" required>
        </div>
      </div>

      <!-- Memory -->
      <div class="tab-content" id="memory">
        <div class="form-group">
          <label for="memory">RAM (MiB):</label>
          <input type="number" id="memory" name="memory" required>
        </div>
      </div>

      <!-- Confirm -->
      <div class="tab-content" id="confirm">
        <h3>Créer votre nouvelle Machine Virtuelle</h3>
        <button type="submit" class="button">Créer</button>
      </div>
    </form>
  </div>
```

RT3-FI

```
  <script>
    document.addEventListener('DOMContentLoaded', () => {
        document.querySelectorAll('.tab').forEach(tab => {
            tab.addEventListener('click', () => {
                document.querySelectorAll('.tab').forEach(t =>
t.classList.remove('active'));
                document.querySelectorAll('.tab-content').forEach(tc =>
tc.classList.remove('active'));
                tab.classList.add('active');

document.getElementById(tab.dataset.target).classList.add('active');
            });
        });

        // Fonction pour récupérer les cookies
        function getCookie(name) {
            const value = `; ${document.cookie}`;
            const parts = value.split(`; ${name}=`);
            if (parts.length === 2) return
parts.pop().split(';').shift();
            return null;
        }

        let username, password;

        try {
            // Récupérer et décoder le cookie contenant les
identifiants
            const userCredentials = getCookie('userCredentials');
            if (!userCredentials) throw new Error('Cookie
userCredentials introuvable');

            const decodedUrl = decodeURIComponent(userCredentials);
            const decodedCredentials = atob(decodedUrl);
            [username, password] = decodedCredentials.split(':');

            if (!username || !password) throw new Error('Identifiant ou
mot de passe manquant');
        } catch (error) {
            console.error('Erreur lors du décodage du cookie
userCredentials :', error);
            alert('Erreur avec le cookie d\'authentification. Veuillez
vous reconnecter.');
            window.location.href = '/login';
            return;
        }

        async function fetchIso() {
            const isoImageSelect =
document.getElementById('iso-image');
```

RT3-FI

```
            try {
                const response = await
fetch(`/iso-list?username=${encodeURIComponent(username)}&password=${en
codeURIComponent(password)}`);
                if (!response.ok) throw new Error('Failed to fetch ISO
images');

                const isoImages = await response.json();
                isoImageSelect.innerHTML = isoImages
                    .map(iso => `<option
value="${iso.volid}">${iso.volid}</option>`)
                    .join('');
            } catch (error) {
                console.error('Error fetching ISO images:', error);
                isoImageSelect.innerHTML = '<option value="">Erreur
lors du chargement des ISO</option>';
            }
        }

        async function fetchVMID() {
            const vmidInput = document.getElementById('vmid');
            try {
                const response = await
fetch(`/all-vms?username=${encodeURIComponent(username)}&password=${enc
odeURIComponent(password)}`);
                if (!response.ok) throw new Error('Failed to fetch
VMs');

                const vms = await response.json();
                const maxVMID = Math.max(...vms.map(vm => vm.vmid), 0);
                vmidInput.value = maxVMID + 1;
            } catch (error) {
                console.error('Error fetching VMID:', error);
                vmidInput.placeholder = 'Enter VM ID';
            }
        }

        fetchIso();
        fetchVMID();


document.getElementById('createVmForm').addEventListener('submit',
async (event) => {
            event.preventDefault();
            const formData = new FormData(event.target);
            const data = Object.fromEntries(formData);
            data.username = username;
            data.password = password;

            try {
                const response = await fetch('/new-vm', {
```

```
                    method: 'POST',
                    headers: { 'Content-Type': 'application/json' },
                    body: JSON.stringify(data),
                });

                if (response.ok) {
                    alert('VM créée avec succès !');
                    const result = await response.json();
                    window.location.href = '/dashboard';
                } else {
                    const errorText = await response.text();
                    alert(`Erreur lors de la création de la VM :
${errorText}`);
                }
            } catch (error) {
                console.error('Erreur :', error);
                alert('Une erreur est survenue lors de la création de
la VM.');
            }
        });
    });
  </script>
  <script src="./logout.js"></script>
</body>
</html>
```

del_vm.html

```
<!DOCTYPE html>
<html lang="fr">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Supprimer une VM</title>
  <link rel="icon" type="image/x-icon" href="./VMM.ico">
  <style>
    body {
      font-family: Arial, sans-serif;
      background-color: #f9f9f9;
      margin: 0;
      padding: 0;
    }

    /* Navbar Styles */
    .modern-navbar {
      display: flex;
      justify-content: space-between;
      align-items: center;
      padding: 15px 30px;
      background-color: #2c3e50;
      color: white;
```

RT3-FI

```
    box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);
    flex-wrap: wrap;
}

.modern-navbar .logo {
    font-size: 1.5em;
    font-weight: bold;
    flex: 1 1 auto;
}

.modern-navbar .nav-links {
    list-style-type: none;
    margin: 0;
    padding: 0;
    display: flex;
    flex-wrap: wrap;
    gap: 10px;
    justify-content: flex-end;
    flex: 1 1 auto;
}

.modern-navbar .nav-links li {
    display: inline;
}

.modern-navbar .nav-links a {
    text-decoration: none;
    font-weight: bold;
    color: white;
    padding: 10px 15px;
    border-radius: 15px;
    background-color: #364b61;
    transition: background-color 0.3s ease;
    text-align: center;
}

.modern-navbar .nav-links a:hover {
    background-color: #2c3e50;
}

.welcome {
    text-align: center;
    padding: 20px;
    background-color: #ecf0f1;
    border-bottom: 1px solid #dcdcdc;
    width: 100%;
}

.welcome h1 {
    margin: 0;
    font-size: 2em;
    color: #2c3e50;
}
```

RT3-FI

```
.container {
  display: flex;
  flex-wrap: wrap;
  justify-content: center;
  gap: 20px;
  padding: 20px;
}

.delete-vm-container {
  margin-top: 30px;
  width: 600px;
  max-width: 400px;
  background: #fff;
  overflow: hidden;
  padding: 20px;
  box-shadow: 0 4px 8px rgba(0, 0, 0, 0.2);
  border-radius: 10px;
  align-items: center;
}

.delete-vm-container h2 {
  text-align: center;
  margin-bottom: 20px;
}

.delete-vm-container label {
  font-weight: bold;
  display: block;
  margin-bottom: 5px;
}

.delete-vm-container input {
  width: 95%;
  padding: 10px;
  margin-bottom: 15px;
  border: 1px solid #ccc;
  border-radius: 4px;
}

.delete-vm-container button {
  width: 100%;
  padding: 10px;
  background: #d9534f;
  color: #fff;
  border: none;
  border-radius: 4px;
  cursor: pointer;
  font-size: 1em;
}

.delete-vm-container button:hover {
  background: #c9302c;
```

```
    }

    .error-message {
      color: red;
      text-align: center;
      margin-top: 10px;
      font-size: 0.9em;
    }

    .success-message {
      color: green;
      text-align: center;
      margin-top: 10px;
      font-size: 0.9em;
    }

    /* Responsive Styles */
    @media (max-width: 885px) {
      .modern-navbar {
        flex-direction: column;
        align-items: flex-start;
      }

      .modern-navbar .nav-links {
        justify-content: flex-start;
        margin-top: 15px;
      }

      .modern-navbar .nav-links a {
        width: 100%;
      }
    }
  </style>
</head>
<body>
  <nav class="modern-navbar">
    <img src="./img/VMM.png" alt="VMM" style="width: 50px; margin-right:
5px;">
    <div class="logo">Virtual Machines Manager</div>
    <ul class="nav-links">
      <li><a href="/dashboard">Accueil</a></li>
      <li><a href="/new-vm">Nouvelle VM</a></li>
      <li><a href="/del-vm">Supprimer VM</a></li>
      <li><a href="#" id="logout">Se déconnecter</a></li>
    </ul>
  </nav>

  <div class="welcome">
    <h1 id="welcome-user">Supprimer une VM</h1>
  </div>

    <div class="container">
        <div class="delete-vm-container">
```

RT3-FI

```
            <h2>Supprimer une VM</h2>
            <form id="deleteVmForm">
            <label for="vmid">VMID</label>
            <input type="number" id="vmid" placeholder="Entrez le VMID"
required>
            <button type="submit">Supprimer</button>
            </form>
            <div id="result"></div>
        </div>
    </div>

  <script>
    document.getElementById('deleteVmForm').addEventListener('submit', async
(e) => {
      e.preventDefault();

      const vmid = document.getElementById('vmid').value;

      if (!confirm(`Êtes-vous sûr de vouloir supprimer la VM avec VMID
${vmid} ?`)) {
        return; // Annuler la suppression si l'utilisateur clique sur
"Annuler"
      }

      // Fonction pour récupérer les cookies
      function getCookie(name) {
        const value = `; ${document.cookie}`;
        const parts = value.split(`; ${name}=`);
        if (parts.length === 2) return parts.pop().split(';').shift();
        return null;
      }

      let username, password;

      try {
        // Récupérer le cookie contenant les identifiants
        const userCredentials = getCookie('userCredentials');

        if (!userCredentials) {
          throw new Error('Cookie userCredentials introuvable');
        }

        // Décoder le cookie
        const decodedUrl = decodeURIComponent(userCredentials);
        const decodedCredentials = atob(decodedUrl);

        // Extraire le nom d'utilisateur et le mot de passe
        [username, password] = decodedCredentials.split(':');

        if (!username || !password) {
          throw new Error('Identifiant ou mot de passe manquant après
décodage');
        }
```

```
      } catch (error) {
        console.error('Erreur lors du décodage du cookie userCredentials :',
error);
        alert('Erreur avec le cookie d\'authentification. Veuillez vous
reconnecter.');
        window.location.href = '/login';
        return; // Arrêter l'exécution si le cookie est invalide
      }

      try {
        // Envoi de la requête de suppression avec les identifiants
        const response = await
fetch(`/vms/${vmid}?username=${encodeURIComponent(username)}&password=${encod
eURIComponent(password)}`, {
          method: 'DELETE',
          credentials: 'include',
        });

        const resultElement = document.getElementById('result');
        if (response.ok) {
          const message = await response.text();
          resultElement.className = 'success-message';
          resultElement.textContent = `Succès : ${message}`;
          setTimeout(() => (window.location.href = '/dashboard'), 1000);
        } else {
          const errorText = await response.text();
          resultElement.className = 'error-message';
          resultElement.textContent = `Erreur : ${errorText}`;
        }
      } catch (error) {
        console.error('Erreur lors de la suppression de la VM :', error);
        const resultElement = document.getElementById('result');
        resultElement.className = 'error-message';
        resultElement.textContent = 'Une erreur est survenue lors de la
suppression de la VM.';
      }
    });
  </script>
  <script src="./logout.js"></script>
</body>
</html>
```

logout.js

```
document.getElementById('logout').addEventListener('click', async () => {
  try {
    // Appeler la route de déconnexion
    await fetch('https://192.168.0.56:4000/logout', { method: 'POST',
credentials: 'include' });

    // Vider le sessionStorage
    sessionStorage.clear();

    // Rediriger vers la page d'accueil
    window.location.href = '/';
```

```
  } catch (error) {
    console.error('Erreur lors de la déconnexion :', error);
  }
});
```

Sources

wg-easy :

- https://docs.techdox.nz/wgeasy/
- https://github.com/wg-easy/wg-easy
- https://www.youtube.com/watch?v=SogiBS2gRI8&t=712s

proxmox :

RT3-FI

- https://pve.proxmox.com/pve-docs/pve-admin-guide.html
- https://pve.proxmox.com/wiki/Main_Page
- https://pve.proxmox.com/pve-docs/api-viewer/index.html
- https://pve.proxmox.com/wiki/Proxmox_VE_API

ChatGPT :

- https://chatgpt.com

développement durable :

- https://www.agenda-2030.fr/17-objectifs-de-developpement-durable/