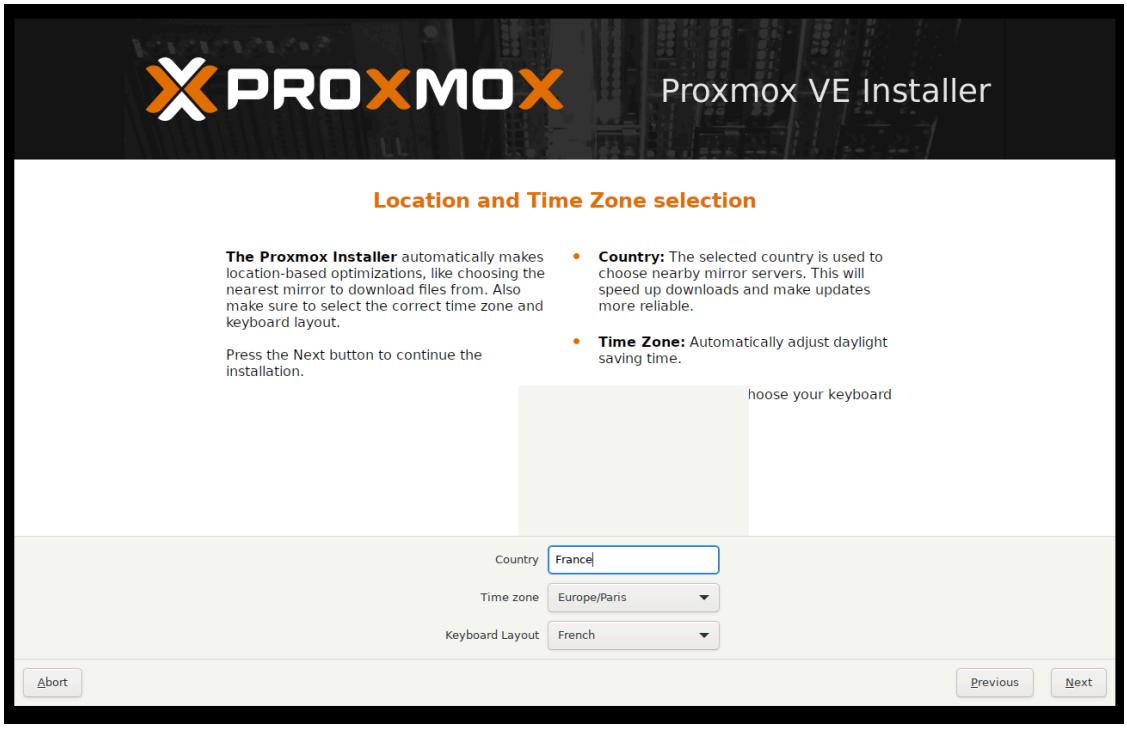
Proxmox in web with user interface

⚠️ **WARNING :** Make you sure you have virtualisation active on the server.

Create a machine with proxmox and select graphical install at boot, agree to the license and select disk for install :

select the disk to install proxmox

Choose your country and your keyboard :



select your country

Fill your password and add your mail :

add password

Configure your network :

IP

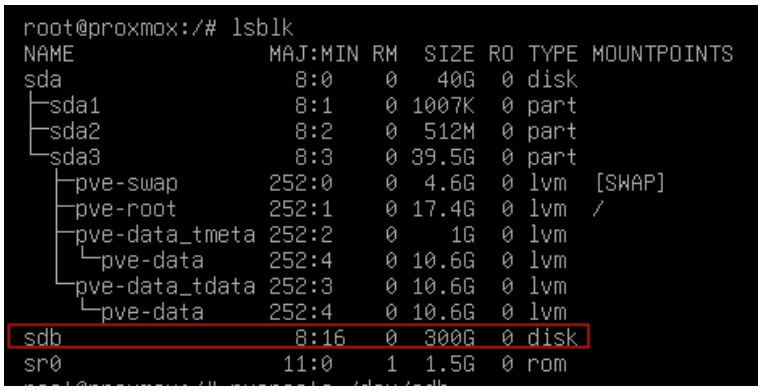
Then click on install.

Once installed, the login is “root” and the password is the one you have put during the installation.

Now, we can add disk to proxmox server :

To add disk to proxmox server, it’s necessary to locate the disk :

`lsblk`



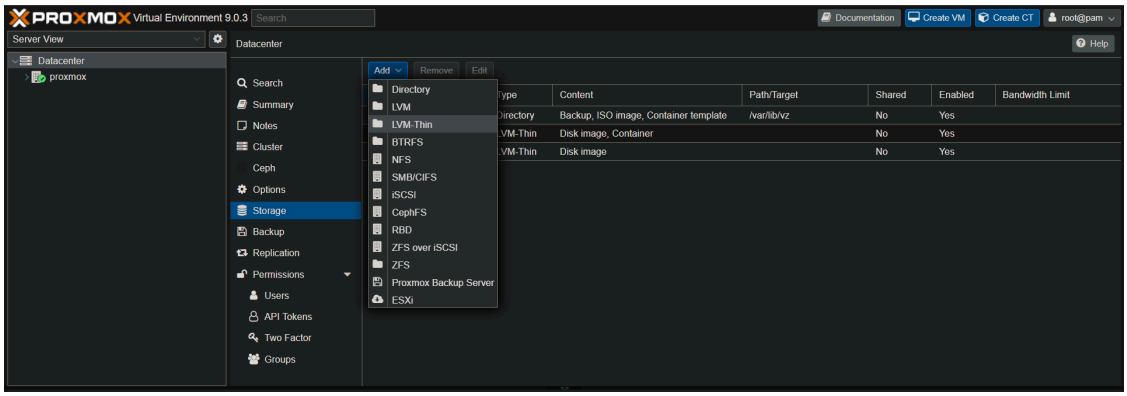
lsblk

Once the disk is located, you can add him to proxmox :

```
vgcreate local-lvm /dev/sdb
lvcreate -l 100%FREE -T local-lvm/local-lvm-2
```

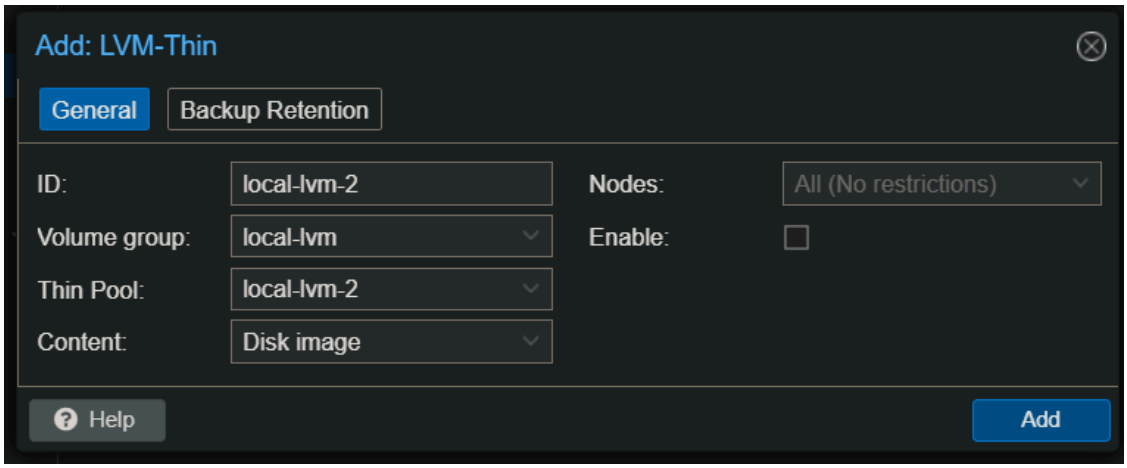
To add the disk, go to the graphical interface (http://YOUR-IP:8006)

Go to Datacenter > Storage > Add > LVM-Thin :



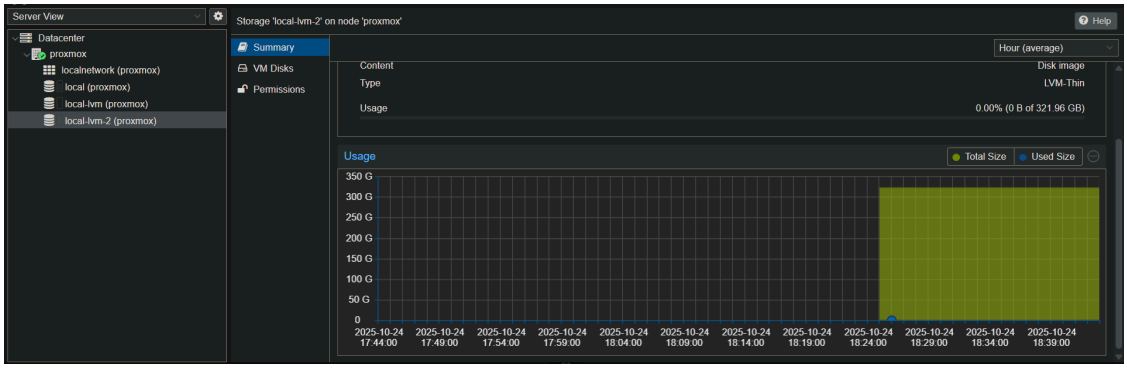
add_disk

Add the disk :



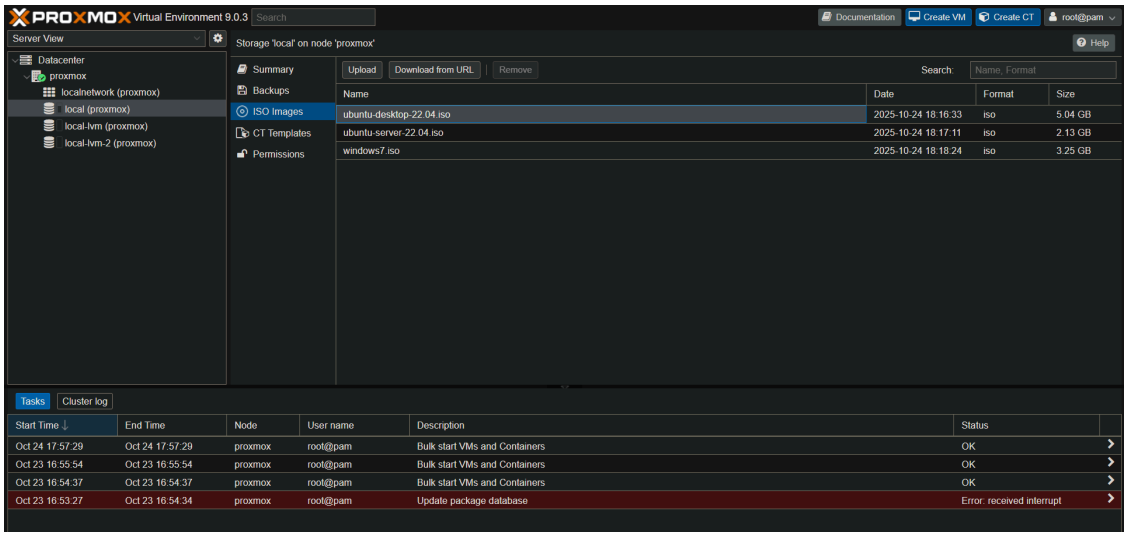
add_lvm_thin

To verify, click on the new disk in the left menu and look at the size :



new_disk

To create VM, you need the .iso file, so you can upload in graphical interface mode into the local disk :



add_iso

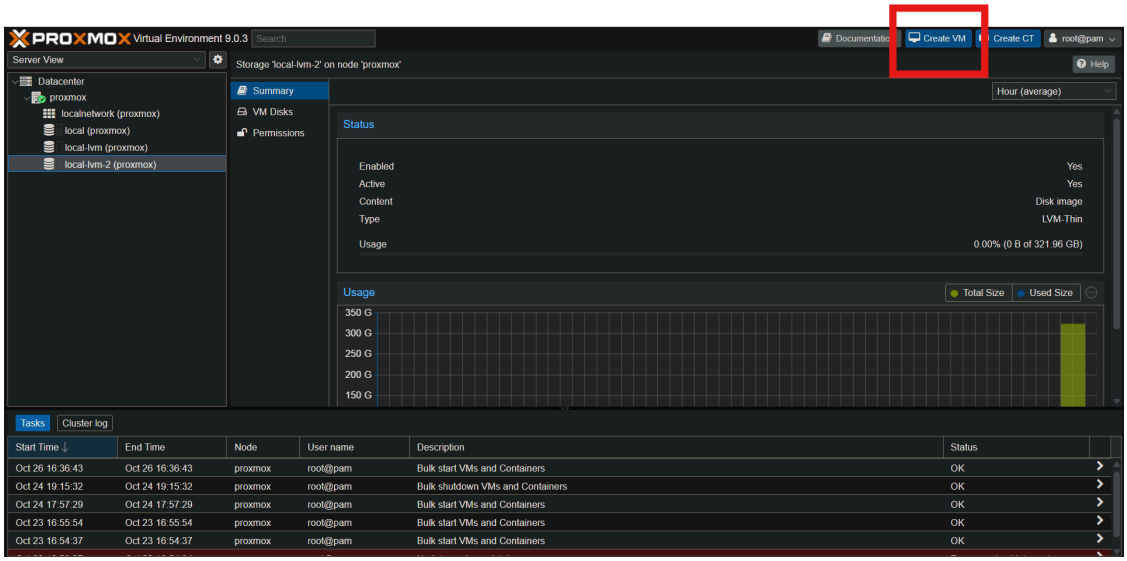
or directly in the server in :

`/var/lib/vz/template/iso`

Create template :

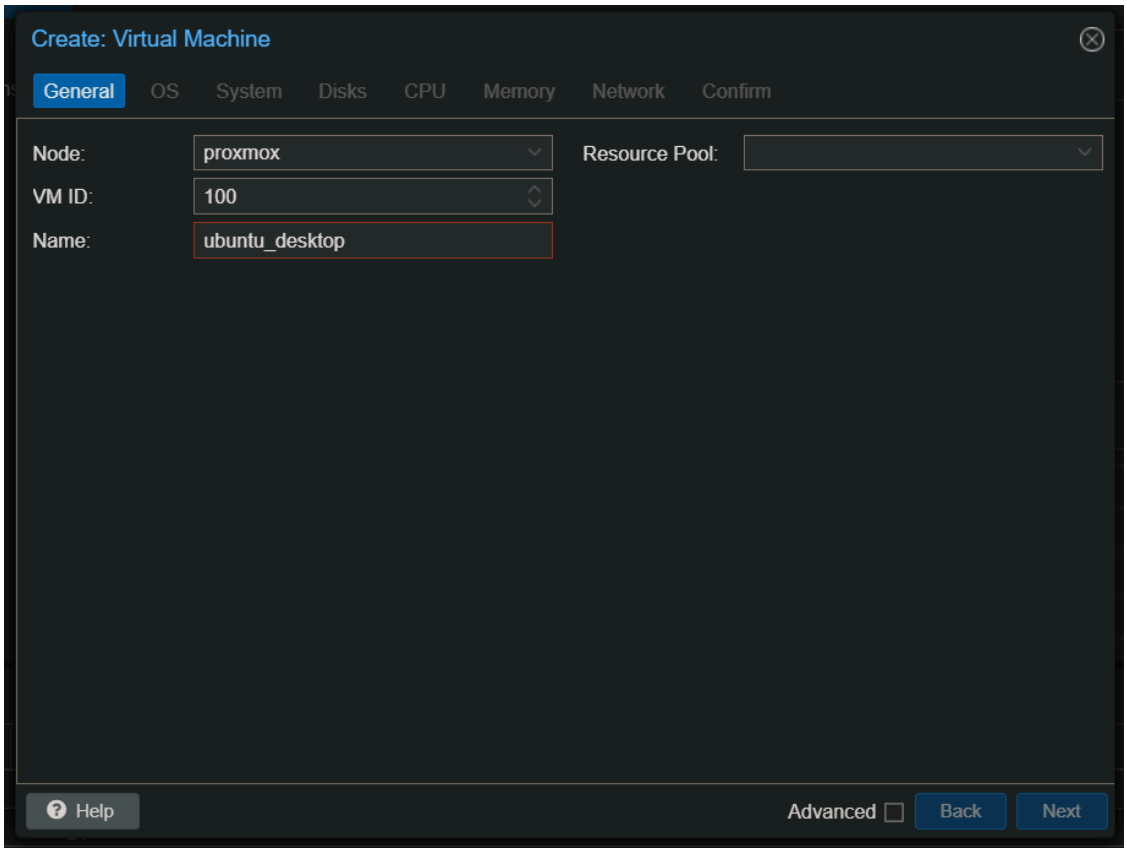
To create a template, you must create a VM and convert this one in a template :

To create a VM, you must log in the proxmox GUI and go to create VM :



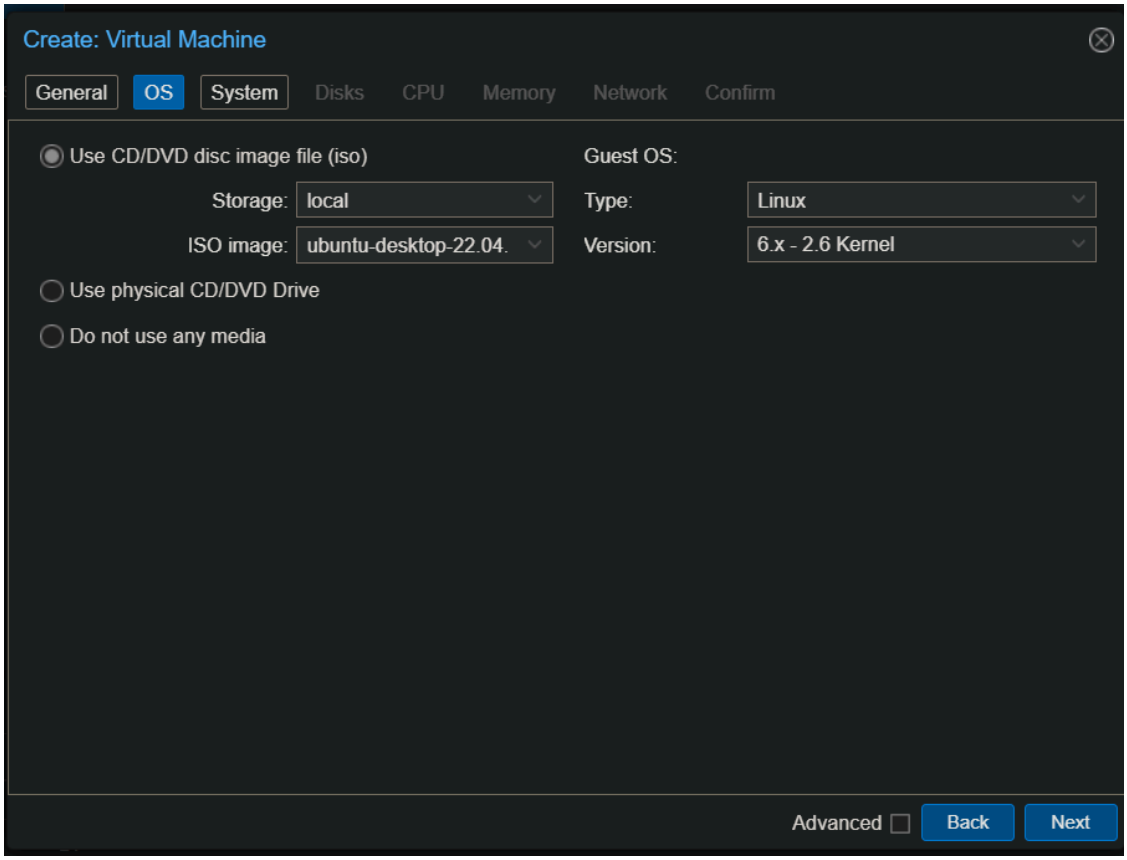
create_vm

Give it a name :



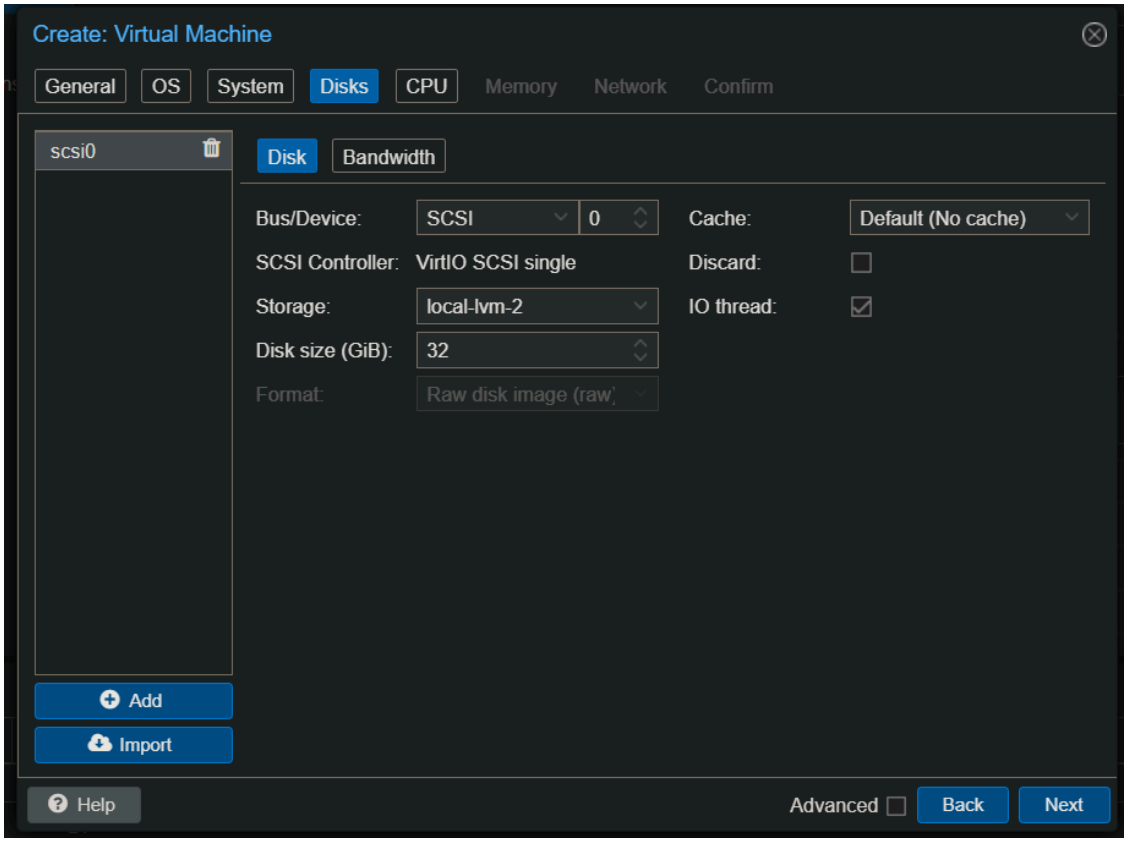
VM_name

Select your ISO :



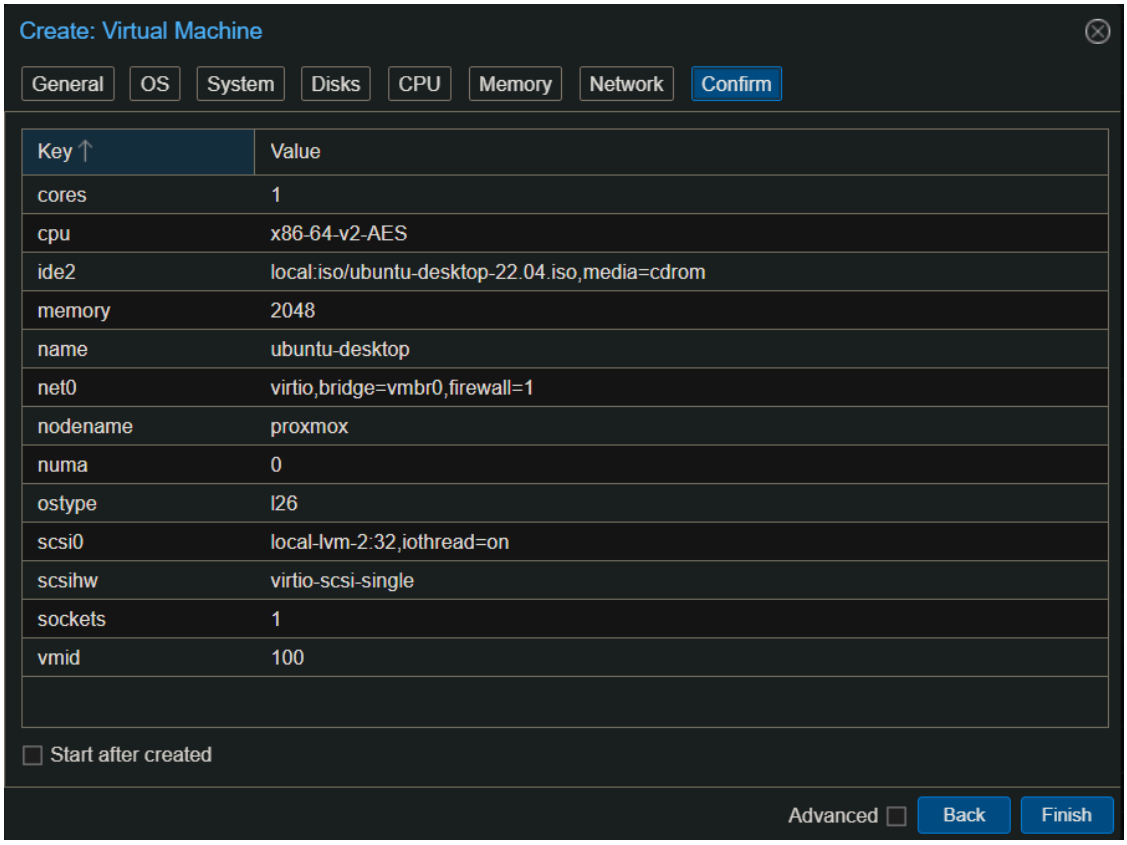
select_ISO

In Disk, select your second disk and choose the capacity of your virtual disk :



choose_capacity

And in another tab, choose your number of CPU or Memory and finish the creation.

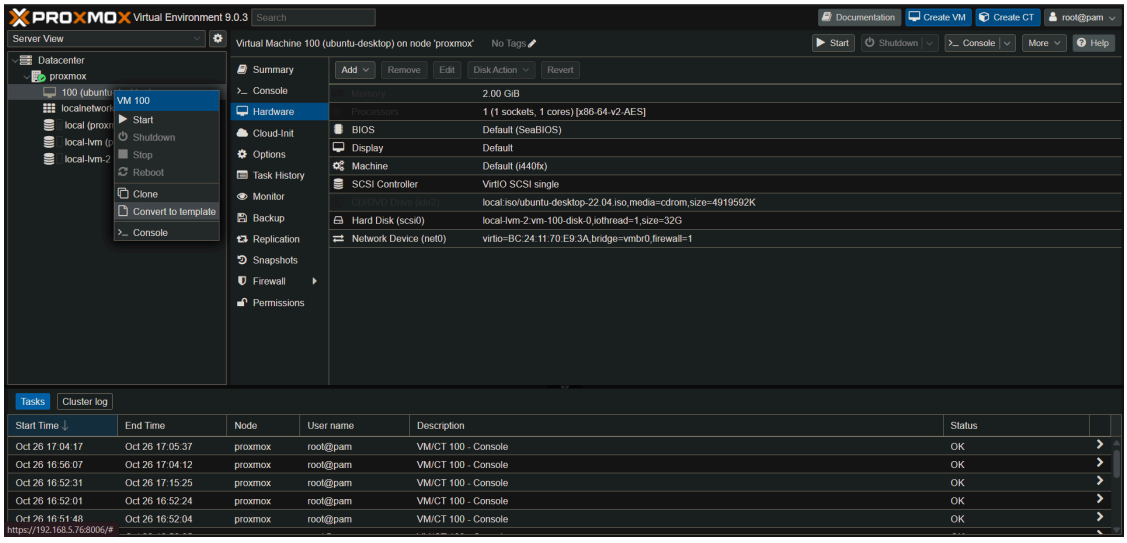


finish_creation

Now you have to run your virtual machine and install the system.

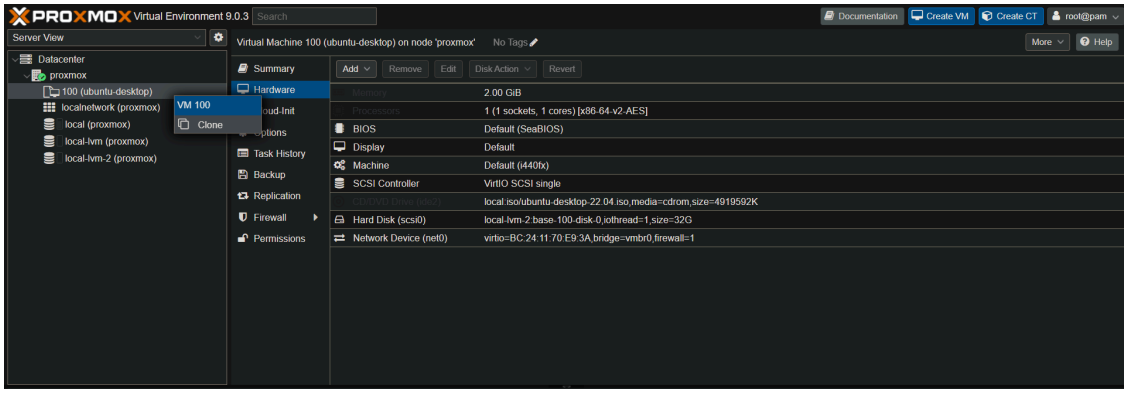
When you have installed your OS, you can shutdown your VM.

Then you can right click on your VM and select convert to template :



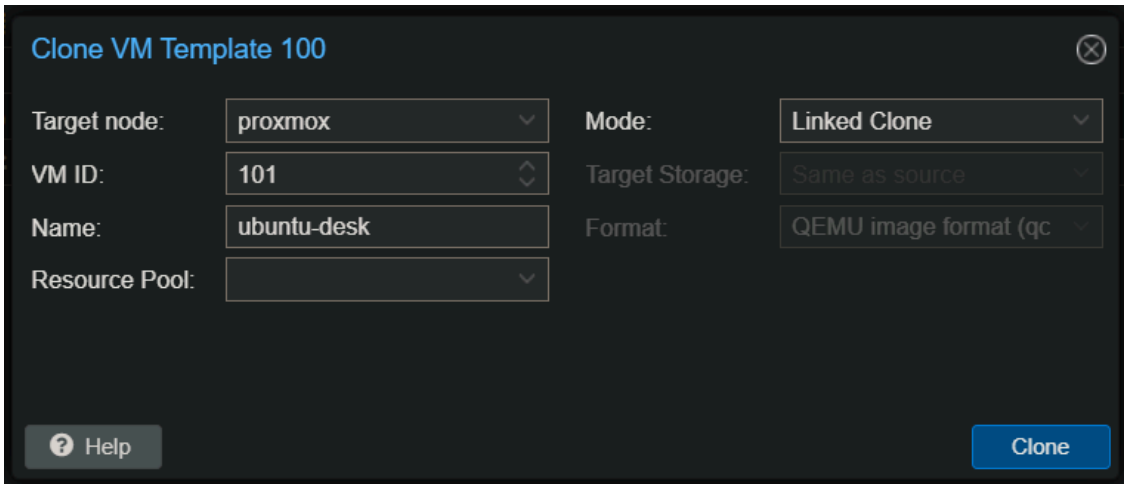
convert_to_template

Now you can create the clone of your template, to do that you have to right click on the template and select clone :



clone_template

Give a name to the new cloned VM and choose the “Linked Clone” mode to have a fast creation (if the template is deleted, the VM can’t run) :



clone_name

Deploy web interface :

First you have to create a Linux server, and then you must install this package :

```
sudo apt update && sudo apt install docker.io docker-compose -y
```

To have encrypted data, you have to generate the .env key, to create, you must execute this code :

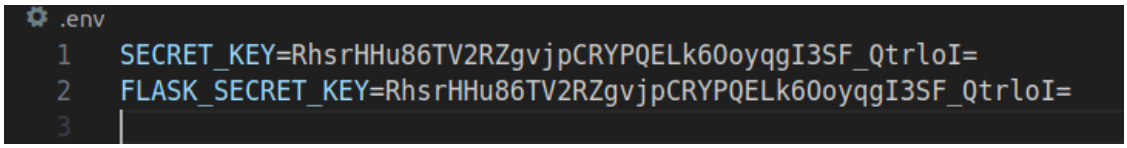
creation_key.py

```
#!/usr/bin/env python3
```

```
import os
from dotenv import load_dotenv
from cryptography.fernet import Fernet

# Nom du fichier .env
ENV_FILE = ".env"

# Vérifier si le fichier .env existe déjà
if not os.path.exists(ENV_FILE):
    # Générer une nouvelle clé secrète
    secret_key = Fernet.generate_key().decode()

    # Écrire la clé dans .env
    with open(ENV_FILE, "w") as f:
        f.write(f"SECRET_KEY={secret_key}\n")

    print("Fichier .env créé avec une nouvelle clé secrète.")

# Charger les variables d'environnement
load_dotenv()

# Récupérer la clé secrète
SECRET_KEY = os.getenv("SECRET_KEY").encode()

# Initialiser Fernet
cipher = Fernet(SECRET_KEY)
```

```
chmod +x creation_key.py
./creation_key.py
```

Now you have to copy the “SECRET_KEY” created in the .env file and paste it to the other variable “FLASK_SECRET_KEY” that you have created manually:



.env file content

After all that, you have to generate your own encrypted data with your key, to generate this, use the following code :

genere_encode.py

```
#!/usr/bin/env python3

import os
from cryptography.fernet import Fernet
from dotenv import load_dotenv
```

```
load_dotenv()

SECRET_KEY = os.getenv("SECRET_KEY")
if not SECRET_KEY:
    raise SystemExit("SECRET_KEY not found in .env")

# si la clé est stockée en base64 déjà encodée, on l'utilise telle quelle
cipher = Fernet(SECRET_KEY.encode() if isinstance(SECRET_KEY, str) else
SECRET_KEY)

def enc(s):
    if s is None:
        s = ""
    return cipher.encrypt(s.encode()).decode()



print()
print('data encrypt : ')

data1 = "project_web_site"

print()
print(enc(data1))

data2 = "progtr00"

print()
print(enc(data2))

print()
```

> ⚠️ **WARNING :** In the app code, you have to change the encrypted data by your own data.

If your server is accessible with NAT then use the following code :

NAT app code

```
from flask import Flask, render_template, request, redirect, url_for, flash,
session, make_response
import radius
from proxmoxer import ProxmoxAPI
import logging
import mysql.connector as MC
import time
from urllib.parse import urlparse
```

```
import paramiko
from dotenv import load_dotenv
from cryptography.fernet import Fernet
import os


load_dotenv() # Charger les variables d'environnement depuis le fichier .env


app = Flask(__name__)
app.config['MAX_CONTENT_LENGTH'] = 50 * 1024 * 1024  # 50 Mo
app.secret_key = os.getenv("FLASK_SECRET_KEY")
logging.basicConfig(level=logging.INFO)


SECRET_KEY = os.getenv("SECRET_KEY").encode() # recuperation de la cle
secrete depuis le fichier .env


cipher = Fernet(SECRET_KEY) # Initialiser l'instance de Fernet pour le
chiffrement/déchiffrement


# Chiffrer une donnée
def encrypt_data(data):
    return cipher.encrypt(data.encode()).decode()

# Déchiffrer une donnée
def decrypt_data(data):
    return cipher.decrypt(data.encode()).decode()

# ======================
# Connexion Proxmox
# ======================
PROXMOX_HOST =
decrypt_data("gAAAAABpJrVxIkv7GH4_jqM4c24oTWQbntqeTiXAeockxigzGNp_RkUwpuyJvHt
AFCdqwP87vY4MLbml_oJLsn4U7mfMmIRTAA==")
PROXMOX_USER =
decrypt_data("gAAAAABpJrVQ-8KWhyXtl9bcZPGs2MUNZG8HFl8WDkiZMGh95Yyx8twN3mWGrcE
yVndsKucmpaS5HBb5t5p74boWAzfqujlh2w==")
PROXMOX_PASS =
decrypt_data("gAAAAABpJq_dzyLrhe0wJGIi70Py93gkgPxumwLFiK5dBZjLZG3CfP3LlMOcpO0
nizgz3V_coRa_tV3NRe3mZsnhIUBLu98L2w==")
NODE_NAME =
decrypt_data("gAAAAABpJrUu29GqWI81n3KJ_hO_emo8aRwp_CKRWCbs6CZzqVCikZoeDZV6_ea
H5WP5-zYrSmend5caIL1BfRzJdfTPObhihA==")
proxmox = ProxmoxAPI(PROXMOX_HOST, user=PROXMOX_USER, password=PROXMOX_PASS,
verify_ssl=False)
```

```
# ======================
# Configuration RADIUS
# ======================
RADIUS_SERVER =
decrypt_data("gAAAAABpJrMwwReuMUq2Tfggci6sVv7RhIe6ekpTEfFtx6_1owPLdTZy_9euzLK
tkDo-6k00W7b_m9_a3RRQutlbDttGCPhwmQ==")
RADIUS_SECRET =
decrypt_data("gAAAAABpJq_dzyLrhe0wJGIi70Py93gkgPxumwLFiK5dBZjLZG3CfP3LlMOcpO0
nizgz3V_coRa_tV3NRe3mZsnhIUBLu98L2w==")
RADIUS_PORT =
int(decrypt_data("gAAAAABpJrTe8WW1Ly8a7liNc6DFAGcDZEvTY5cRodq9riykwhGqxAFbx0_
tL6_OPCGFnt2hZ8oD8PtRdbJXJuA_APGnXm-3Sw=="))
ADMIN_BYPASS_USER =
decrypt_data("gAAAAABpJrSVh7QGU8CevifcBuAnAgCQmhpgljctpWP7wKAgzk_le0O43qQN41d
-Ts1L4bQft2wGy8n-LuUcd10tZd_UtDMfug==")
ADMIN_BYPASS_PASS =
decrypt_data("gAAAAABpJq_dzyLrhe0wJGIi70Py93gkgPxumwLFiK5dBZjLZG3CfP3LlMOcpO0
nizgz3V_coRa_tV3NRe3mZsnhIUBLu98L2w==")
ALLOW_LOCAL_FALLBACK = True

# ======================
# Connexion MySQL
# ======================
def connect_db():
    for i in range(10):
        try:
            connection = MC.connect(

host=decrypt_data("gAAAAABpJrRzOke5bJiulNrqqIQUMKmFp043eMRSRppnGPl7FZ3fEGHZLT
S1i0Tqga9_d378nYOD3PHsdpt4S8-oK875RYwGDA=="),

user=decrypt_data("gAAAAABpJq_dLDBbtp1KEmaQAHGYMB3_duMN_GkzcSHcgsoFAxxR0OHOa1
zG6RsOPuJp2WyM_kafiZ9bGaOpCZgkYvl5Wvg9Hw=="),

password=decrypt_data("gAAAAABpJq_dzyLrhe0wJGIi70Py93gkgPxumwLFiK5dBZjLZG3CfP
3LlMOcpO0nizgz3V_coRa_tV3NRe3mZsnhIUBLu98L2w=="),

database=decrypt_data("gAAAAABpJrWWx9wgBwHCJuW9MY-Ty_9SqBzaNmQTJU-nglK65JoiQ0
zf026LqpbTPMc5ybmZ3qDMg5xVLhlm-96laSUohXRkxYUf8TXD7uXhDsP3tokRlR0=")
            )
            print("Connexion réussie à la base de données !")
            return connection
        except MC.Error as e:
            print(f"Tentative {i+1}/10 : MySQL pas encore prêt ({e}),
attente...")
            time.sleep(3)
    return None

connection = connect_db()
if connection is None:
```

```
    raise RuntimeError("Impossible de se connecter à la base de données après
plusieurs tentatives.")



def get_real_host():
"""Détecte l'hôte réel utilisé par l'utilisateur"""
forwarded_host = request.headers.get("X-Forwarded-Host")
referer = request.headers.get("Referer")
if forwarded_host:
    return forwarded_host.split(":")[0]
elif referer:
    return urlparse(referer).hostname
else:
    return request.host.split(":")[0]



# ======================
# Helpers utilitaires
# ======================
def get_vms_from_proxmox():
    try:
        return proxmox.nodes(NODE_NAME).qemu.get()
    except Exception as e:
        logging.exception("Erreur récupération VMs Proxmox")
        return []

def get_owners_map():
    cursor = connection.cursor(dictionary=True)
    cursor.execute("""
        SELECT vms.vmid, vms.name, vms.owner_id, vms.is_template,
users.username AS owner_name
        FROM vms LEFT JOIN users ON vms.owner_id = users.id
    """)
    rows = cursor.fetchall()
    mapping = {}
    for r in rows:
        try:
            mapping[int(r["vmid"])] = {
                "db_name": r["name"],
                "owner_id": r["owner_id"],
                "owner_name": r["owner_name"],
                "is_template_db": bool(r["is_template"])
            }
        except Exception:
            continue
    cursor.close()
    return mapping
```

```
def find_db_vm_by_vmid(vmid):
    cursor = connection.cursor(dictionary=True)
    cursor.execute("SELECT * FROM vms WHERE vmid = %s", (vmid,))
    row = cursor.fetchone()
    cursor.close()
    return row

def get_or_create_user(username, role="user"):
    cursor = connection.cursor(dictionary=True)
    cursor.execute("SELECT * FROM users WHERE username = %s", (username,))
    user = cursor.fetchone()
    if not user:
        cursor.execute("INSERT INTO users (username, role) VALUES (%s, %s)",
(username, role))
        connection.commit()
        cursor.execute("SELECT * FROM users WHERE username = %s",
(username,))
        user = cursor.fetchone()
    cursor.close()
    return user

def register_vm_in_db(vmid, name, owner_id=None, is_template=False):
    cursor = connection.cursor()
    cursor.execute("SELECT id FROM vms WHERE vmid = %s", (vmid,))
    if not cursor.fetchone():
        cursor.execute(
            "INSERT INTO vms (vmid, name, owner_id, is_template) VALUES (%s,
%s, %s, %s)",
            (vmid, name, owner_id, is_template)
        )
        connection.commit()
    cursor.close()

def find_vmid_by_name(name):
    try:
        vms = proxmox.nodes(NODE_NAME).qemu.get()
        for vm in vms:
            if str(vm.get("name")) == str(name):
                return int(vm["vmid"])
        if vms:
            return max(int(vm["vmid"]) for vm in vms)
    except Exception:
        logging.exception("Erreur récupération VM par nom")
    return None

def user_can_operate_on_vmid(vmid, session_user_id, session_role):
    if session_role == "admin":
        return True
    db_vm = find_db_vm_by_vmid(vmid)
```

```
    if db_vm and db_vm.get("owner_id") == session_user_id:
        return True
    return False



def generate_proxmox_ticket():
    """Génère un ticket d'authentification Proxmox"""
    try:
        auth = proxmox.access.ticket.post(
            username=PROXMOX_USER,
            password=PROXMOX_PASS
        )
        ticket = auth["ticket"]
        csrf = auth["CSRFPreventionToken"]
        logging.info("Ticket Proxmox généré avec succès")
        return ticket, csrf
    except Exception as e:
        logging.exception("Erreur lors de la génération du ticket Proxmox")
        return None, None

# ======================
# ROUTE : Login
# ======================
@app.route("/", methods=["GET", "POST"])
def login():
    if request.method == "GET":
        return render_template("login.html")
    username = request.form.get("username", "").strip()
    password = request.form.get("password", "")
    if not username or not password:
        flash("Nom d'utilisateur ou mot de passe manquant.", "danger")
        return render_template("login.html")
    # Bypass admin
    if username == ADMIN_BYPASS_USER and password == ADMIN_BYPASS_PASS:
        user = get_or_create_user(username, role="admin")
        session["user"] = user["username"]
        session["user_id"] = user["id"]
        session["role"] = user["role"]
        flash("Connexion administrateur réussie", "success")
        return redirect(url_for("machines"))
    # Auth RADIUS
    try:
        client = radius.Radius(RADIUS_SECRET, host=RADIUS_SERVER,
port=RADIUS_PORT)
        if client.authenticate(username, password):
            user = get_or_create_user(username, role="user")
            session["user"] = user["username"]
            session["user_id"] = user["id"]
```

```
            session["role"] = user["role"]
            flash("Connexion réussie via RADIUS", "success")
            return redirect(url_for("machines"))
        else:
            flash("Échec de l'authentification RADIUS", "danger")
            return render_template("login.html")
    except Exception as e:
        logging.exception("Erreur RADIUS")
        if ALLOW_LOCAL_FALLBACK:
            user = get_or_create_user(username, role="user")
            session["user"] = user["username"]
            session["user_id"] = user["id"]
            session["role"] = user["role"]
            flash("Serveur RADIUS injoignable — connexion fallback locale
activée", "warning")
            return redirect(url_for("machines"))
        else:
            flash(f"Erreur de communication avec le serveur RADIUS : {e}",
"danger")
            return render_template("login.html")

# =======================
# ROUTE : Logout
# =======================
@app.route("/logout")
def logout():
    session.clear()
    flash("Déconnecté avec succès", "info")
    return redirect(url_for("login"))

# =======================
# ROUTE : Liste des VMs
# =======================
@app.route("/machines")
def machines():
    if "user" not in session:
        return redirect(url_for("login"))
    user_id = session.get("user_id")
    role = session.get("role", "user")
    proxmox_vms = get_vms_from_proxmox()
    owners_map = get_owners_map()
    all_vms = []
    for vm in proxmox_vms:
        vmid = int(vm["vmid"])
        is_template_flag = bool(vm.get("template", 0))
        db_info = owners_map.get(vmid)
        vm_entry = {
            "vmid": vmid,
            "name": vm.get("name") or (db_info["db_name"] if db_info else
f"vm-{vmid}"),
```

```
            "is_template": is_template_flag or (db_info["is_template_db"] if
db_info else False),
            "owner_id": db_info["owner_id"] if db_info else None,
            "owner_name": db_info["owner_name"] if db_info else None,
            "proxmox_raw": vm
        }
        all_vms.append(vm_entry)
    if role == "admin":
        visible_vms = all_vms
    else:
        visible_vms = [v for v in all_vms if (v["owner_id"] == user_id) or
v["is_template"]]
    vms = [v for v in visible_vms if not v["is_template"]]
    templates = [v for v in visible_vms if v["is_template"]]
    host_used = get_real_host()
    return render_template("index.html", vms=vms, templates=templates,
NODE_NAME=NODE_NAME, HOST_USED=host_used)


# =======================
# ROUTE : Démarrer VM + console NoVNC
# =======================
@app.route("/start_vm", methods=["POST"])
def start_vm():
    if "user" not in session:
        return redirect(url_for("login"))
    try:
        vmid = int(request.form["vmid"])
    except Exception:
        flash("VMID invalide.", "danger")
        return redirect(url_for("machines"))
    user_id = session["user_id"]
    role = session["role"]
    if not user_can_operate_on_vmid(vmid, user_id, role):
        flash("Vous n'avez pas la permission de démarrer cette VM.",
"danger")
        return redirect(url_for("machines"))
    try:
        proxmox.nodes(NODE_NAME).qemu(vmid).status.start.post()
        flash(f"VM {vmid} démarrée.", "success")
    except Exception as e:
        flash(f"Erreur lors du démarrage VM : {e}", "danger")
        return redirect(url_for("machines"))

    # Génération du ticket Proxmox
    ticket, csrf = generate_proxmox_ticket()
    if not ticket:
        flash("Impossible de générer le ticket d'authentification Proxmox",
"danger")
        return redirect(url_for("machines"))
```

```
    # Récupération du nom de la VM
    try:
        vm_info = proxmox.nodes(NODE_NAME).qemu(vmid).config.get()
        vmname = vm_info.get("name", f"vm-{vmid}")
    except Exception:
        vmname = f"vm-{vmid}"

    # Stocker le ticket en session pour la page proxy
    session['console_ticket'] = ticket
    session['console_vmid'] = vmid
    session['console_vmname'] = vmname

    logging.info(f"VM {vmid} démarrée, redirection vers page proxy console")

    # Redirection vers la page proxy qui va définir le cookie
    return redirect(url_for('console_proxy'))

# =======================
# ROUTE : Ouvrir Console
# =======================
@app.route("/open_console/<int:vmid>")
def open_console(vmid):
    if "user" not in session:
        return redirect(url_for("login"))

    user_id = session["user_id"]
    role = session["role"]

    if not user_can_operate_on_vmid(vmid, user_id, role):
        flash("Vous n'avez pas la permission d'accéder à cette console.",
"danger")
        return redirect(url_for("machines"))

    # Génération du ticket Proxmox
    ticket, csrf = generate_proxmox_ticket()
    if not ticket:
        flash("Impossible de générer le ticket d'authentification Proxmox",
"danger")
        return redirect(url_for("machines"))

    # Récupération du nom de la VM
    try:
        vm_info = proxmox.nodes(NODE_NAME).qemu(vmid).config.get()
        vmname = vm_info.get("name", f"vm-{vmid}")
    except Exception:
        vmname = f"vm-{vmid}"

    # Stocker le ticket en session pour la page proxy
    session['console_ticket'] = ticket
```

```
    session['console_vmid'] = vmid
    session['console_vmname'] = vmname

    logging.info(f"Ouverture console NoVNC pour VM {vmid}")

    # Redirection vers la page proxy qui va définir le cookie
    return redirect(url_for('console_proxy'))

# ======================
# ROUTE : Page Proxy Console
# ======================
@app.route("/console_proxy")
def console_proxy():
    """Page proxy qui définit le cookie PVEAuthCookie et redirige vers
NoVNC"""
    if "user" not in session:
        return redirect(url_for("login"))

    # Récupérer les infos depuis la session
    ticket = session.get('console_ticket')
    vmid = session.get('console_vmid')
    vmname = session.get('console_vmname')

    if not ticket or not vmid:
        flash("Session console expirée", "danger")
        return redirect(url_for("machines"))

    # Construction de l'URL NoVNC
    real_host = get_real_host()
    console_url = (
        f"http://{real_host}:8006/?console=kvm&novnc=1"
        f"&vmid={vmid}&vmname={vmname}&node={NODE_NAME}&cmd=connect"
    )

    # Créer une réponse HTML qui définit le cookie et redirige
    html = f"""
    <!DOCTYPE html>
    <html>
    <head>
        <meta charset="UTF-8">
        <title>Ouverture console...</title>
        <style>
            body {{
                font-family: Arial, sans-serif;
                display: flex;
                justify-content: center;
                align-items: center;
                height: 100vh;
                margin: 0;
                background: linear-gradient(135deg, #667eea 0%, #764ba2
```

```
100%);
                color: white;
            }}
            .loader {{
                text-align: center;
            }}
            .spinner {{
                border: 4px solid rgba(255,255,255,0.3);
                border-top: 4px solid white;
                border-radius: 50%;
                width: 50px;
                height: 50px;
                animation: spin 1s linear infinite;
                margin: 0 auto 20px;
            }}
            @keyframes spin {{
                0% {{ transform: rotate(0deg); }}
                100% {{ transform: rotate(360deg); }}
            }}
        </style>
    </head>
    <body>
        <div class="loader">
            <div class="spinner"></div>
            <h2>Ouverture de la console...</h2>
            <p>Redirection en cours</p>
        </div>
        <script>
            // Redirection immédiate vers NoVNC
            window.location.href = "{console_url}";
        </script>
    </body>
    </html>
    """

    # Créer la réponse avec le cookie défini
    response = make_response(html)

    # Définir le cookie PVEAuthCookie pour le domaine Proxmox
    response.set_cookie(
        'PVEAuthCookie',
        value=ticket,
        domain=None,
        path='/',
        secure=False,
        httponly=False,
        samesite='Lax'
    )

    logging.info(f"Cookie PVEAuthCookie défini pour la console VM {vmid}")
```

```
    return response

# ======================
# ROUTE : Stopper VM
# ======================
@app.route("/stop_vm", methods=["POST"])
def stop_vm():
    if "user" not in session:
        return redirect(url_for("login"))
    try:
        vmid = int(request.form["vmid"])
    except Exception:
        flash("VMID invalide.", "danger")
        return redirect(url_for("machines"))
    user_id = session["user_id"]
    role = session["role"]
    if not user_can_operate_on_vmid(vmid, user_id, role):
        flash("Vous n'avez pas la permission d'arrêter cette VM.", "danger")
        return redirect(url_for("machines"))
    try:
        proxmox.nodes(NODE_NAME).qemu(vmid).status.stop.post()
        flash(f"VM {vmid} arrêtée.", "success")
    except Exception as e:
        flash(f"Erreur arrêt VM {vmid}: {e}", "danger")
    return redirect(url_for("machines"))

# ======================
# ROUTE : Supprimer VM (NOUVELLE)
# ======================
@app.route("/delete_vm", methods=["POST"])
def delete_vm():
    if "user" not in session:
        return redirect(url_for("login"))

    try:
        vmid = int(request.form["vmid"])
    except Exception:
        flash("VMID invalide.", "danger")
        return redirect(url_for("machines"))

    user_id = session["user_id"]
    role = session["role"]

    # Vérifier les permissions
    if not user_can_operate_on_vmid(vmid, user_id, role):
        flash("Vous n'avez pas la permission de supprimer cette VM.",
"danger")
        return redirect(url_for("machines"))
```

```
    try:
        # Vérifier que la VM est bien arrêtée
        vm_status = proxmox.nodes(NODE_NAME).qemu(vmid).status.current.get()
        if vm_status.get("status") != "stopped":
            flash(f"La VM {vmid} doit être arrêtée avant d'être supprimée.",
"warning")
            return redirect(url_for("machines"))

        # Supprimer la VM avec purge des volumes (disques)
        # purge=1 supprime tous les volumes associés
        proxmox.nodes(NODE_NAME).qemu(vmid).delete(purge=1)

        # Supprimer de la base de données
        cursor = connection.cursor()
        cursor.execute("DELETE FROM vms WHERE vmid = %s", (vmid,))
        connection.commit()
        cursor.close()

        flash(f"VM {vmid} supprimée avec succès (volumes inclus).",
"success")
        logging.info(f"VM {vmid} supprimée par l'utilisateur
{session['user']}")

    except Exception as e:
        logging.exception(f"Erreur suppression VM {vmid}")
        flash(f"Erreur lors de la suppression de la VM {vmid}: {e}",
"danger")

    return redirect(url_for("machines"))

# =======================
# ROUTE : Cloner Template
# =======================
@app.route("/clone_template", methods=["POST"])
def clone_template():
    if "user" not in session:
        return redirect(url_for("login"))
    try:
        template_id = int(request.form["template_id"])
    except Exception:
        flash("template_id invalide.", "danger")
        return redirect(url_for("machines"))

    new_name = request.form.get("new_name", "").strip()
    if not new_name:
        flash("Nom de la nouvelle VM requis.", "danger")
        return redirect(url_for("machines"))

    user_id = session["user_id"]
    role = session["role"]
```

```
    try:
        tcfg = proxmox.nodes(NODE_NAME).qemu(template_id).config.get()
        if not (tcfg.get("template") == 1 or tcfg.get("template") == "1"):
            if role != "admin":
                flash("Seuls les templates peuvent être clonés par un
utilisateur normal.", "danger")
                return redirect(url_for("machines"))

        # Générer un nouvel ID unique
        newid = proxmox.cluster.nextid.get()

        # Clonage avec paramètres requis
        proxmox.nodes(NODE_NAME).qemu(template_id).clone.post(
            newid=newid,
            name=new_name,
            target=NODE_NAME,
            full=0
        )

        time.sleep(3)

        cursor = connection.cursor()
        cursor.execute(
            "INSERT INTO vms (vmid, name, owner_id, is_template) VALUES (%s,
%s, %s, %s)",
            (newid, new_name, user_id, False)
        )
        connection.commit()
        cursor.close()

        flash(f"Template {template_id} cloné en VM '{new_name}'
(VMID={newid}).", "success")

    except Exception as e:
        logging.exception("Erreur clonage template")
        flash(f"Erreur clonage template : {e}", "danger")

    return redirect(url_for("machines"))

# ======================
# creation et suppression user
# ======================

@app.route("/create_user")
def create_user():
    if 'user' not in session or session["user"] != "administrateur":
        return redirect(url_for('login'))
    return render_template("ad.html")
```

```python
@app.route('/send_info', methods=['POST'])
def send_info():
    user_to_create = request.form.get('nom')
    password_to_set = request.form.get('password')

    host =
decrypt_data("gAAAAABpJrMwwReuMUq2Tfggci6sVv7RhIe6ekpTEfFtx6_1owPLdTZy_9euzLK
tkDo-6k00W7b_m9_a3RRQutlbDttGCPhwmQ==")
    username =
decrypt_data("gAAAAABpJrP-PcI-X19OOAtWu5eI0bE1XYWoroYHfr0Mif9aJc8l39EyMh4aehL
RfJHBYMTX-t8CvzLYurDAYXY3veU3mFMeEw==")
    password =
decrypt_data("gAAAAABpJq_dhjTDI_UG9k28PHc0KE4mTvQIq06nhmcTgxS-eRnSfMRrj6bf_Lz
Ckl7Ffg11O4J3oI-4CXwG7R-IL8dEu04jxQ==")

    command = (
        f'powershell.exe -NoProfile -ExecutionPolicy Bypass '
        f'-File "C:\\Scripts\\create_user.ps1" '
        f'-UserName "{user_to_create}" '
        f'-Password "{password_to_set}"'
    )

    ssh = paramiko.SSHClient()
    ssh.set_missing_host_key_policy(paramiko.AutoAddPolicy())
    ssh.connect(host, username=username, password=password)

    stdin, stdout, stderr = ssh.exec_command(command)

    ssh.close()

    return redirect(url_for("create_user"))


@app.route('/delete_info', methods=['POST'])
def delete_info():
    user_to_create = request.form.get('nom')

    host =
decrypt_data("gAAAAABpJrMwwReuMUq2Tfggci6sVv7RhIe6ekpTEfFtx6_1owPLdTZy_9euzLK
tkDo-6k00W7b_m9_a3RRQutlbDttGCPhwmQ==")
    username =
decrypt_data("gAAAAABpJrP-PcI-X19OOAtWu5eI0bE1XYWoroYHfr0Mif9aJc8l39EyMh4aehL
RfJHBYMTX-t8CvzLYurDAYXY3veU3mFMeEw==")
    password =
decrypt_data("gAAAAABpJq_dhjTDI_UG9k28PHc0KE4mTvQIq06nhmcTgxS-eRnSfMRrj6bf_Lz
Ckl7Ffg11O4J3oI-4CXwG7R-IL8dEu04jxQ==")
```

```
    command = (
        f'powershell.exe -NoProfile -ExecutionPolicy Bypass '
        f'-File "C:\\Scripts\\delete_user.ps1" '
        f'-UserName "{user_to_create}" '
    )

    ssh = paramiko.SSHClient()
    ssh.set_missing_host_key_policy(paramiko.AutoAddPolicy())
    ssh.connect(host, username=username, password=password)

    stdin, stdout, stderr = ssh.exec_command(command)

    ssh.close()

    return redirect(url_for("create_user"))


if __name__ == "__main__":
    app.run(host="0.0.0.0", port=5000, debug=True)
```

But if the server is directly accessible, follow the code below :

Bridge app code

```
from flask import Flask, render_template, request, redirect, url_for, flash,
session, make_response
import radius
from proxmoxer import ProxmoxAPI
import logging
import mysql.connector as MC
import time
from urllib.parse import urlparse
import paramiko
from dotenv import load_dotenv
from cryptography.fernet import Fernet
import os


load_dotenv() # Charger les variables d'environnement depuis le fichier .env


app = Flask(__name__)
app.config['MAX_CONTENT_LENGTH'] = 50 * 1024 * 1024  # 50 Mo
app.secret_key = os.getenv("FLASK_SECRET_KEY")
logging.basicConfig(level=logging.INFO)


SECRET_KEY = os.getenv("SECRET_KEY").encode() # recuperation de la cle
```

```
secrete depuis le fichier .env


cipher = Fernet(SECRET_KEY) # Initialiser l'instance de Fernet pour le
chiffrement/déchiffrement


# Chiffrer une donnée
def encrypt_data(data):
    return cipher.encrypt(data.encode()).decode()

# Déchiffrer une donnée
def decrypt_data(data):
    return cipher.decrypt(data.encode()).decode()

# =======================
# Connexion Proxmox
# =======================
PROXMOX_HOST =
decrypt_data("gAAAAABpJrVxIkv7GH4_jqM4c24oTWQbntqeTiXAeockxigzGNp_RkUwpuyJvHt
AFCdqwP87vY4MLbml_oJLsn4U7mfMmIRTAA==")
PROXMOX_USER =
decrypt_data("gAAAAABpJrVQ-8KWhyXtl9bcZPGs2MUNZG8HFl8WDkiZMGh95Yyx8twN3mWGrcE
yVndsKucmpaS5HBb5t5p74boWAzfqujlh2w==")
PROXMOX_PASS =
decrypt_data("gAAAAABpJq_dzyLrhe0wJGIi70Py93gkgPxumwLFiK5dBZjLZG3CfP3LlMOcpO0
nizgz3V_coRa_tV3NRe3mZsnhIUBLu98L2w==")
NODE_NAME =
decrypt_data("gAAAAABpJrUu29GqWI81n3KJ_hO_emo8aRwp_CKRWCbs6CZzqVCikZoeDZV6_ea
H5WP5-zYrSmend5caIL1BfRzJdfTPObhihA==")
proxmox = ProxmoxAPI(PROXMOX_HOST, user=PROXMOX_USER, password=PROXMOX_PASS,
verify_ssl=False)

# =======================
# Configuration RADIUS
# =======================
RADIUS_SERVER =
decrypt_data("gAAAAABpJrMwwReuMUq2Tfggci6sVv7RhIe6ekpTEfFtx6_1owPLdTZy_9euzLK
tkDo-6k00W7b_m9_a3RRQutlbDttGCPhwmQ==")
RADIUS_SECRET =
decrypt_data("gAAAAABpJq_dzyLrhe0wJGIi70Py93gkgPxumwLFiK5dBZjLZG3CfP3LlMOcpO0
nizgz3V_coRa_tV3NRe3mZsnhIUBLu98L2w==")
RADIUS_PORT =
int(decrypt_data("gAAAAABpJrTe8WW1Ly8a7liNc6DFAGcDZEvTY5cRodq9riykwhGqxAFbx0_
tL6_OPCGFnt2hZ8oD8PtRdbJXJuA_APGnXm-3Sw=="))
ADMIN_BYPASS_USER =
decrypt_data("gAAAAABpJrSVh7QGU8CevifcBuAnAgCQmhpgljctpWP7wKAgzk_le0O43qQN41d
-Ts1L4bQft2wGy8n-LuUcd10tZd_UtDMfug==")
ADMIN_BYPASS_PASS =
```

```
decrypt_data("gAAAAABpJq_dzyLrhe0wJGIi70Py93gkgPxumwLFiK5dBZjLZG3CfP3LlMOcpO0
nizgz3V_coRa_tV3NRe3mZsnhIUBLu98L2w==")
ALLOW_LOCAL_FALLBACK = True

# =======================
# Connexion MySQL
# =======================
def connect_db():
    for i in range(10):
        try:
            connection = MC.connect(

host=decrypt_data("gAAAAABpJrRzOke5bJiulNrqqIQUMKmFp043eMRSRppnGPl7FZ3fEGHZLT
S1i0Tqga9_d378nYOD3PHsdpt4S8-oK875RYwGDA=="),

user=decrypt_data("gAAAAABpJq_dLDBbtp1KEmaQAHGYMB3_duMN_GkzcSHcgsoFAxxR0OHOa1
zG6RsOPuJp2WyM_kafiZ9bGaOpCZgkYvl5Wvg9Hw=="),

password=decrypt_data("gAAAAABpJq_dzyLrhe0wJGIi70Py93gkgPxumwLFiK5dBZjLZG3CfP
3LlMOcpO0nizgz3V_coRa_tV3NRe3mZsnhIUBLu98L2w=="),

database=decrypt_data("gAAAAABpJrWWx9wgBwHCJuW9MY-Ty_9SqBzaNmQTJU-nglK65JoiQ0
zf026LqpbTPMc5ybmZ3qDMg5xVLhlm-96laSUohXRkxYUf8TXD7uXhDsP3tokRlR0=")
            )
            print("Connexion réussie à la base de données !")
            return connection
        except MC.Error as e:
            print(f"Tentative {i+1}/10 : MySQL pas encore prêt ({e}),
attente...")
            time.sleep(3)
    return None

connection = connect_db()
if connection is None:
    raise RuntimeError("Impossible de se connecter à la base de données après
plusieurs tentatives.")

# =======================
# Helpers utilitaires
# =======================
def get_vms_from_proxmox():
    try:
        return proxmox.nodes(NODE_NAME).qemu.get()
    except Exception as e:
        logging.exception("Erreur récupération VMs Proxmox")
        return []

def get_owners_map():
    cursor = connection.cursor(dictionary=True)
```

```
    cursor.execute("""
        SELECT vms.vmid, vms.name, vms.owner_id, vms.is_template,
users.username AS owner_name
        FROM vms LEFT JOIN users ON vms.owner_id = users.id
    """)
    rows = cursor.fetchall()
    mapping = {}
    for r in rows:
        try:
            mapping[int(r["vmid"])] = {
                "db_name": r["name"],
                "owner_id": r["owner_id"],
                "owner_name": r["owner_name"],
                "is_template_db": bool(r["is_template"])
            }
        except Exception:
            continue
    cursor.close()
    return mapping

def find_db_vm_by_vmid(vmid):
    cursor = connection.cursor(dictionary=True)
    cursor.execute("SELECT * FROM vms WHERE vmid = %s", (vmid,))
    row = cursor.fetchone()
    cursor.close()
    return row

def get_or_create_user(username, role="user"):
    cursor = connection.cursor(dictionary=True)
    cursor.execute("SELECT * FROM users WHERE username = %s", (username,))
    user = cursor.fetchone()
    if not user:
        cursor.execute("INSERT INTO users (username, role) VALUES (%s, %s)",
(username, role))
        connection.commit()
        cursor.execute("SELECT * FROM users WHERE username = %s",
(username,))
        user = cursor.fetchone()
    cursor.close()
    return user

def register_vm_in_db(vmid, name, owner_id=None, is_template=False):
    cursor = connection.cursor()
    cursor.execute("SELECT id FROM vms WHERE vmid = %s", (vmid,))
    if not cursor.fetchone():
        cursor.execute(
            "INSERT INTO vms (vmid, name, owner_id, is_template) VALUES (%s,
%s, %s, %s)",
            (vmid, name, owner_id, is_template)
        )
```

```
        connection.commit()
    cursor.close()

def find_vmid_by_name(name):
    try:
        vms = proxmox.nodes(NODE_NAME).qemu.get()
        for vm in vms:
            if str(vm.get("name")) == str(name):
                return int(vm["vmid"])
        if vms:
            return max(int(vm["vmid"]) for vm in vms)
    except Exception:
        logging.exception("Erreur récupération VM par nom")
    return None

def user_can_operate_on_vmid(vmid, session_user_id, session_role):
    if session_role == "admin":
        return True
    db_vm = find_db_vm_by_vmid(vmid)
    if db_vm and db_vm.get("owner_id") == session_user_id:
        return True
    return False



def generate_proxmox_ticket():
    """Génère un ticket d'authentification Proxmox"""
    try:
        auth = proxmox.access.ticket.post(
            username=PROXMOX_USER,
            password=PROXMOX_PASS
        )
        ticket = auth["ticket"]
        csrf = auth["CSRFPreventionToken"]
        logging.info("Ticket Proxmox généré avec succès")
        return ticket, csrf
    except Exception as e:
        logging.exception("Erreur lors de la génération du ticket Proxmox")
        return None, None

# =======================
# ROUTE : Login
# =======================
@app.route("/", methods=["GET", "POST"])
def login():
    if request.method == "GET":
        return render_template("login.html")
    username = request.form.get("username", "").strip()
    password = request.form.get("password", "")
```

```
    if not username or not password:
        flash("Nom d'utilisateur ou mot de passe manquant.", "danger")
        return render_template("login.html")
    # Bypass admin
    if username == ADMIN_BYPASS_USER and password == ADMIN_BYPASS_PASS:
        user = get_or_create_user(username, role="admin")
        session["user"] = user["username"]
        session["user_id"] = user["id"]
        session["role"] = user["role"]
        flash("Connexion administrateur réussie", "success")
        return redirect(url_for("machines"))
    # Auth RADIUS
    try:
        client = radius.Radius(RADIUS_SECRET, host=RADIUS_SERVER,
port=RADIUS_PORT)
        if client.authenticate(username, password):
            user = get_or_create_user(username, role="user")
            session["user"] = user["username"]
            session["user_id"] = user["id"]
            session["role"] = user["role"]
            flash("Connexion réussie via RADIUS", "success")
            return redirect(url_for("machines"))
        else:
            flash("Échec de l'authentification RADIUS", "danger")
            return render_template("login.html")
    except Exception as e:
        logging.exception("Erreur RADIUS")
        if ALLOW_LOCAL_FALLBACK:
            user = get_or_create_user(username, role="user")
            session["user"] = user["username"]
            session["user_id"] = user["id"]
            session["role"] = user["role"]
            flash("Serveur RADIUS injoignable — connexion fallback locale
activée", "warning")
            return redirect(url_for("machines"))
        else:
            flash(f"Erreur de communication avec le serveur RADIUS : {e}",
"danger")
            return render_template("login.html")

# =======================
# ROUTE : Logout
# =======================
@app.route("/logout")
def logout():
    session.clear()
    flash("Déconnecté avec succès", "info")
    return redirect(url_for("login"))

# =======================
```

```python
# ROUTE : Liste des VMs
# =======================
@app.route("/machines")
def machines():
    if "user" not in session:
        return redirect(url_for("login"))
    user_id = session.get("user_id")
    role = session.get("role", "user")
    proxmox_vms = get_vms_from_proxmox()
    owners_map = get_owners_map()
    all_vms = []
    for vm in proxmox_vms:
        vmid = int(vm["vmid"])
        is_template_flag = bool(vm.get("template", 0))
        db_info = owners_map.get(vmid)
        vm_entry = {
            "vmid": vmid,
            "name": vm.get("name") or (db_info["db_name"] if db_info else
f"vm-{vmid}"),
            "is_template": is_template_flag or (db_info["is_template_db"] if
db_info else False),
            "owner_id": db_info["owner_id"] if db_info else None,
            "owner_name": db_info["owner_name"] if db_info else None,
            "proxmox_raw": vm
        }
        all_vms.append(vm_entry)
    if role == "admin":
        visible_vms = all_vms
    else:
        visible_vms = [v for v in all_vms if (v["owner_id"] == user_id) or
v["is_template"]]
    vms = [v for v in visible_vms if not v["is_template"]]
    templates = [v for v in visible_vms if v["is_template"]]
    return render_template("index.html", vms=vms, templates=templates,
NODE_NAME=NODE_NAME, HOST_USED=PROXMOX_HOST)

# =======================
# ROUTE : Démarrer VM + console NoVNC
# =======================
@app.route("/start_vm", methods=["POST"])
def start_vm():
    if "user" not in session:
        return redirect(url_for("login"))
    try:
        vmid = int(request.form["vmid"])
    except Exception:
        flash("VMID invalide.", "danger")
        return redirect(url_for("machines"))
    user_id = session["user_id"]
    role = session["role"]
```

```
    if not user_can_operate_on_vmid(vmid, user_id, role):
        flash("Vous n'avez pas la permission de démarrer cette VM.",
"danger")
        return redirect(url_for("machines"))
    try:
        proxmox.nodes(NODE_NAME).qemu(vmid).status.start.post()
        flash(f"VM {vmid} démarrée.", "success")
    except Exception as e:
        flash(f"Erreur lors du démarrage VM : {e}", "danger")
        return redirect(url_for("machines"))

    # Génération du ticket Proxmox
    ticket, csrf = generate_proxmox_ticket()
    if not ticket:
        flash("Impossible de générer le ticket d'authentification Proxmox",
"danger")
        return redirect(url_for("machines"))

    # Récupération du nom de la VM
    try:
        vm_info = proxmox.nodes(NODE_NAME).qemu(vmid).config.get()
        vmname = vm_info.get("name", f"vm-{vmid}")
    except Exception:
        vmname = f"vm-{vmid}"

    # Stocker le ticket en session pour la page proxy
    session['console_ticket'] = ticket
    session['console_vmid'] = vmid
    session['console_vmname'] = vmname

    logging.info(f"VM {vmid} démarrée, redirection vers page proxy console")

    # Redirection vers la page proxy qui va définir le cookie
    return redirect(url_for('console_proxy'))

# =======================
# ROUTE : Ouvrir Console
# =======================
@app.route("/open_console/<int:vmid>")
def open_console(vmid):
    if "user" not in session:
        return redirect(url_for("login"))

    user_id = session["user_id"]
    role = session["role"]

    if not user_can_operate_on_vmid(vmid, user_id, role):
        flash("Vous n'avez pas la permission d'accéder à cette console.",
"danger")
        return redirect(url_for("machines"))
```

```
    # Génération du ticket Proxmox
    ticket, csrf = generate_proxmox_ticket()
    if not ticket:
        flash("Impossible de générer le ticket d'authentification Proxmox",
"danger")
        return redirect(url_for("machines"))

    # Récupération du nom de la VM
    try:
        vm_info = proxmox.nodes(NODE_NAME).qemu(vmid).config.get()
        vmname = vm_info.get("name", f"vm-{vmid}")
    except Exception:
        vmname = f"vm-{vmid}"

    # Stocker le ticket en session pour la page proxy
    session['console_ticket'] = ticket
    session['console_vmid'] = vmid
    session['console_vmname'] = vmname

    logging.info(f"Ouverture console NoVNC pour VM {vmid}")

    # Redirection vers la page proxy qui va définir le cookie
    return redirect(url_for('console_proxy'))

# =======================
# ROUTE : Page Proxy Console
# =======================
@app.route("/console_proxy")
def console_proxy():
    """Page proxy qui définit le cookie PVEAuthCookie et redirige vers
NoVNC"""
    if "user" not in session:
        return redirect(url_for("login"))

    # Récupérer les infos depuis la session
    ticket = session.get('console_ticket')
    vmid = session.get('console_vmid')
    vmname = session.get('console_vmname')

    if not ticket or not vmid:
        flash("Session console expirée", "danger")
        return redirect(url_for("machines"))

    # Construction de l'URL NoVNC
    console_url = (
        f"http://{PROXMOX_HOST}:8006/?console=kvm&novnc=1"
        f"&vmid={vmid}&vmname={vmname}&node={NODE_NAME}&cmd=connect"
    )
```

```
    # Créer une réponse HTML qui définit le cookie et redirige
    html = f"""
    <!DOCTYPE html>
    <html>
    <head>
        <meta charset="UTF-8">
        <title>Ouverture console...</title>
        <style>
            body {{
                font-family: Arial, sans-serif;
                display: flex;
                justify-content: center;
                align-items: center;
                height: 100vh;
                margin: 0;
                background: linear-gradient(135deg, #667eea 0%, #764ba2
100%);
                color: white;
            }}
            .loader {{
                text-align: center;
            }}
            .spinner {{
                border: 4px solid rgba(255,255,255,0.3);
                border-top: 4px solid white;
                border-radius: 50%;
                width: 50px;
                height: 50px;
                animation: spin 1s linear infinite;
                margin: 0 auto 20px;
            }}
            @keyframes spin {{
                0% {{ transform: rotate(0deg); }}
                100% {{ transform: rotate(360deg); }}
            }}
        </style>
    </head>
    <body>
        <div class="loader">
            <div class="spinner"></div>
            <h2>Ouverture de la console...</h2>
            <p>Redirection en cours</p>
        </div>
        <script>
            // Redirection immédiate vers NoVNC
            window.location.href = "{console_url}";
        </script>
    </body>
    </html>
    """
```

```
    # Créer la réponse avec le cookie défini
    response = make_response(html)

    # Définir le cookie PVEAuthCookie pour le domaine Proxmox
    response.set_cookie(
        'PVEAuthCookie',
        value=ticket,
        domain=None,
        path='/',
        secure=False,
        httponly=False,
        samesite='Lax'
    )

    logging.info(f"Cookie PVEAuthCookie défini pour la console VM {vmid}")

    return response

# ======================
# ROUTE : Stopper VM
# ======================
@app.route("/stop_vm", methods=["POST"])
def stop_vm():
    if "user" not in session:
        return redirect(url_for("login"))
    try:
        vmid = int(request.form["vmid"])
    except Exception:
        flash("VMID invalide.", "danger")
        return redirect(url_for("machines"))
    user_id = session["user_id"]
    role = session["role"]
    if not user_can_operate_on_vmid(vmid, user_id, role):
        flash("Vous n'avez pas la permission d'arrêter cette VM.", "danger")
        return redirect(url_for("machines"))
    try:
        proxmox.nodes(NODE_NAME).qemu(vmid).status.stop.post()
        flash(f"VM {vmid} arrêtée.", "success")
    except Exception as e:
        flash(f"Erreur arrêt VM {vmid}: {e}", "danger")
    return redirect(url_for("machines"))

# ======================
# ROUTE : Supprimer VM (NOUVELLE)
# ======================
@app.route("/delete_vm", methods=["POST"])
def delete_vm():
    if "user" not in session:
        return redirect(url_for("login"))
```

```
    try:
        vmid = int(request.form["vmid"])
    except Exception:
        flash("VMID invalide.", "danger")
        return redirect(url_for("machines"))

    user_id = session["user_id"]
    role = session["role"]

    # Vérifier les permissions
    if not user_can_operate_on_vmid(vmid, user_id, role):
        flash("Vous n'avez pas la permission de supprimer cette VM.",
"danger")
        return redirect(url_for("machines"))

    try:
        # Vérifier que la VM est bien arrêtée
        vm_status = proxmox.nodes(NODE_NAME).qemu(vmid).status.current.get()
        if vm_status.get("status") != "stopped":
            flash(f"La VM {vmid} doit être arrêtée avant d'être supprimée.",
"warning")
            return redirect(url_for("machines"))

        # Supprimer la VM avec purge des volumes (disques)
        # purge=1 supprime tous les volumes associés
        proxmox.nodes(NODE_NAME).qemu(vmid).delete(purge=1)

        # Supprimer de la base de données
        cursor = connection.cursor()
        cursor.execute("DELETE FROM vms WHERE vmid = %s", (vmid,))
        connection.commit()
        cursor.close()

        flash(f"VM {vmid} supprimée avec succès (volumes inclus).",
"success")
        logging.info(f"VM {vmid} supprimée par l'utilisateur
{session['user']}")

    except Exception as e:
        logging.exception(f"Erreur suppression VM {vmid}")
        flash(f"Erreur lors de la suppression de la VM {vmid}: {e}",
"danger")

    return redirect(url_for("machines"))

# =======================
# ROUTE : Cloner Template
# =======================
@app.route("/clone_template", methods=["POST"])
```

```
def clone_template():
    if "user" not in session:
        return redirect(url_for("login"))
    try:
        template_id = int(request.form["template_id"])
    except Exception:
        flash("template_id invalide.", "danger")
        return redirect(url_for("machines"))

    new_name = request.form.get("new_name", "").strip()
    if not new_name:
        flash("Nom de la nouvelle VM requis.", "danger")
        return redirect(url_for("machines"))

    user_id = session["user_id"]
    role = session["role"]

    try:
        tcfg = proxmox.nodes(NODE_NAME).qemu(template_id).config.get()
        if not (tcfg.get("template") == 1 or tcfg.get("template") == "1"):
            if role != "admin":
                flash("Seuls les templates peuvent être clonés par un
utilisateur normal.", "danger")
                return redirect(url_for("machines"))

        # Générer un nouvel ID unique
        newid = proxmox.cluster.nextid.get()

        # Clonage avec paramètres requis
        proxmox.nodes(NODE_NAME).qemu(template_id).clone.post(
            newid=newid,
            name=new_name,
            target=NODE_NAME,
            full=0
        )

        time.sleep(3)

        cursor = connection.cursor()
        cursor.execute(
            "INSERT INTO vms (vmid, name, owner_id, is_template) VALUES (%s,
%s, %s, %s)",
            (newid, new_name, user_id, False)
        )
        connection.commit()
        cursor.close()

        flash(f"Template {template_id} cloné en VM '{new_name}'
(VMID={newid}).", "success")
```

```
    except Exception as e:
        logging.exception("Erreur clonage template")
        flash(f"Erreur clonage template : {e}", "danger")

    return redirect(url_for("machines"))

# ======================
# creation et suppression user
# ======================

@app.route("/create_user")
def create_user():
    if 'user' not in session or session["user"] != "administrateur":
        return redirect(url_for('login'))
    return render_template("ad.html")


@app.route('/send_info', methods=['POST'])
def send_info():
    user_to_create = request.form.get('nom')
    password_to_set = request.form.get('password')

    host =
decrypt_data("gAAAAABpJrMwwReuMUq2Tfggci6sVv7RhIe6ekpTEfFtx6_1owPLdTZy_9euzLK
tkDo-6k00W7b_m9_a3RRQutlbDttGCPhwmQ==")
    username =
decrypt_data("gAAAAABpJrP-PcI-X19OOAtWu5eI0bE1XYWoroYHfr0Mif9aJc8l39EyMh4aehL
RfJHBYMTX-t8CvzLYurDAYXY3veU3mFMeEw==")
    password =
decrypt_data("gAAAAABpJq_dhjTDI_UG9k28PHc0KE4mTvQIq06nhmcTgxS-eRnSfMRrj6bf_Lz
Ckl7Ffg11O4J3oI-4CXwG7R-IL8dEu04jxQ==")

    command = (
        f'powershell.exe -NoProfile -ExecutionPolicy Bypass '
        f'-File "C:\\Scripts\\create_user.ps1" '
        f'-UserName "{user_to_create}" '
        f'-Password "{password_to_set}"'
    )

    ssh = paramiko.SSHClient()
    ssh.set_missing_host_key_policy(paramiko.AutoAddPolicy())
    ssh.connect(host, username=username, password=password)

    stdin, stdout, stderr = ssh.exec_command(command)

    ssh.close()

    return redirect(url_for("create_user"))
```

```python
@app.route('/delete_info', methods=['POST'])
def delete_info():
    user_to_create = request.form.get('nom')

    host =
decrypt_data("gAAAAABpJrMwwReuMUq2Tfggci6sVv7RhIe6ekpTEfFtx6_1owPLdTZy_9euzLK
tkDo-6k00W7b_m9_a3RRQutlbDttGCPhwmQ==")
    username =
decrypt_data("gAAAAABpJrP-PcI-X19OOAtWu5eI0bE1XYWoroYHfr0Mif9aJc8l39EyMh4aehL
RfJHBYMTX-t8CvzLYurDAYXY3veU3mFMeEw==")
    password =
decrypt_data("gAAAAABpJq_dhjTDI_UG9k28PHc0KE4mTvQIq06nhmcTgxS-eRnSfMRrj6bf_Lz
Ckl7Ffg11O4J3oI-4CXwG7R-IL8dEu04jxQ==")

    command = (
        f'powershell.exe -NoProfile -ExecutionPolicy Bypass '
        f'-File "C:\\Scripts\\delete_user.ps1" '
        f'-UserName "{user_to_create}" '
    )

    ssh = paramiko.SSHClient()
    ssh.set_missing_host_key_policy(paramiko.AutoAddPolicy())
    ssh.connect(host, username=username, password=password)

    stdin, stdout, stderr = ssh.exec_command(command)

    ssh.close()

    return redirect(url_for("create_user"))


if __name__ == "__main__":
    app.run(host="0.0.0.0", port=5000, debug=True)
```

Then, you have to paste your docker-compose.yml and your Dockerfile at the root of the project with all other codes :

Dockerfile

```
FROM python:3.10-slim

WORKDIR /app


RUN pip install six pexpect paramiko
```

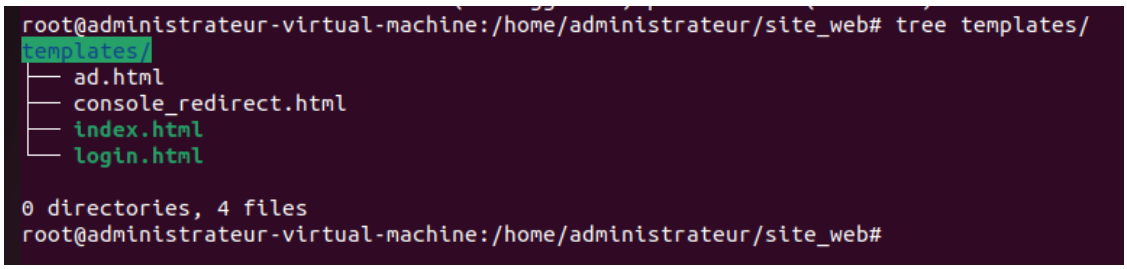
```
RUN pip install requests flask flask-socketio eventlet proxmoxer radius
mysql-connector-python
RUN pip install  dotenv cryptography


COPY . .

CMD ["python", "app.py"]
```

docker-compose.yml

```
version: "3.3"

networks:
  mon_reseau:
    driver: bridge
    ipam:
      config:
        - subnet: 10.16.0.0/24

volumes:
  dbdata:

services:
  db:
    image: mariadb:10.5
    container_name: mysql
    restart: always
    volumes:
      - dbdata:/var/lib/mysql
    environment:
      MYSQL_ROOT_PASSWORD: progtr00
      MYSQL_DATABASE: project_web_site
      MYSQL_USER: root
      MYSQL_PASSWORD: progtr00
    networks:
      - mon_reseau

  phpmyadmin:
    image: phpmyadmin/phpmyadmin
    container_name: phpmyadmin
    restart: always
    depends_on:
      - db
    ports:
      - "8081:80"
    environment:
      PMA_HOST: db
      MYSQL_ROOT_PASSWORD: progtr00
    networks:
```

```
      - mon_reseau

  flask-app:
    build: .
    container_name: flask_app
    restart: always
    depends_on:
      - db
    ports:
      - "5000:5000"
    volumes:
      - .:/app
    networks:
      - mon_reseau
```

To have a web interface, you must have HTML file :



html_file

ad.html

```
<!DOCTYPE html>
<html lang="fr" data-theme="auto">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Connexion</title>

  <style>
    /* ====== THEME (copié depuis ton index, allégé) ====== */
    :root {
      --bg:#0f1115;
      --panel:#12161c;
      --text:#e6e8ee;
      --text-strong:#ffffff;
      --muted:#8892a6;
      --border:#1e2430;
      --primary:#6c8cff;
      --primary-600:#5a78e6;
      --danger:#ff6b6b;
```

```
  --danger-700:#e45a5a;
  --radius:16px;
  --shadow:0 10px 30px rgba(0,0,0,.35);
}
html[data-theme="light"] {
  --bg:#f5f7fb;
  --panel:#ffffff;
  --text:#171a21;
  --text-strong:#0a0c10;
  --muted:#58627a;
  --border:#e8ecf3;
  --primary:#4b6bff;
  --primary-600:#3d59e6;
  --danger:#f25555;
  --danger-700:#d94949;
}

body{
  margin:0;
  height:100vh;
  display:flex;
  align-items:center;
  justify-content:center;
  background:var(--bg);
  color:var(--text);
  font-family: ui-sans-serif, system-ui, -apple-system, Segoe UI;
  position:relative;
}

.back-btn{
  position:absolute;
  top:20px;
  left:20px;
  padding:10px 14px;
  border-radius:12px;
  background:var(--panel);
  border:1px solid var(--border);
  color:var(--text);
  text-decoration:none;
  display:inline-flex;
  align-items:center;
  gap:8px;
  font-weight:600;
  transition:.2s ease;
  box-shadow:var(--shadow);
}
.back-btn:hover{
  transform:translateY(-1px);
  border-color:var(--primary);
  background:color-mix(in oklab, var(--primary) 12%, var(--panel));
```

```
    }

    .login-box{
      width:100%;
      max-width:420px;
      background:var(--panel);
      border:1px solid var(--border);
      box-shadow:var(--shadow);
      padding:32px;
      border-radius:var(--radius);
    }

    h1{
      margin:0 0 22px 0;
      text-align:center;
      color:var(--text-strong);
    }

    input{
      width:93%;
      padding:12px 14px;
      margin-bottom:14px;
      background:var(--bg);
      color:var(--text);
      border:1px solid var(--border);
      border-radius:12px;
      outline:none;
      transition:.2s;
    }
    input:focus{
      border-color:var(--primary);
      box-shadow:0 0 0 3px color-mix(in oklab, var(--primary) 25%,
transparent);
    }

    .btn{
      width:100%;
      padding:12px 14px;
      border-radius:12px;
      border:none;
      cursor:pointer;
      font-weight:700;
      color:#fff;
      background:linear-gradient(180deg, color-mix(in oklab, var(--primary)
18%, transparent), var(--primary));
      border:1px solid color-mix(in oklab, var(--primary) 60%,
var(--border));
      transition:.2s ease;
    }
```

```
    .btn:hover{
      background:var(--primary-600);
      transform:translateY(-1px);
    }

    .theme-toggle{
      display:flex;
      justify-content:center;
      margin-bottom:18px;
      gap:10px;
      align-items:center;
    }
    .switch{
      --w:54px; --h:30px; --pad:3px;
      position:relative; width:var(--w); height:var(--h);
    }
    .switch input{ opacity:0; position:absolute; inset:0; }
    .track{
      border-radius:999px;
      inset:0;
      position:absolute;
      background:var(--panel);
      border:1px solid var(--border);
    }
    .thumb{
      position:absolute;
      width:calc(var(--h) - var(--pad)*2);
      height:calc(var(--h) - var(--pad)*2);
      border-radius:50%;
      background:var(--panel);
      border:1px solid var(--border);
      top:var(--pad);
      left:var(--pad);
      transition:.25s;
      display:grid;
      place-items:center;
      overflow:hidden;
    }
    .switch input:checked ~ .thumb{
      transform:translateX(calc(var(--w) - var(--h)));
    }
    .thumb svg{ width:16px; height:16px; }
  </style>
</head>

<body>
  <!-- Bouton Retour en haut à gauche -->
  <a href="{{ url_for('machines') }}" class="back-btn">
    <svg width="16" height="16" viewBox="0 0 24 24" fill="none"
stroke="currentColor" stroke-width="2" stroke-linecap="round"
```

```
stroke-linejoin="round">
      <path d="M19 12H5M12 19l-7-7 7-7"/>
    </svg>
    Retour aux VMs
  </a>

  <div class="login-box">

    <!-- Switch Thème -->
    <div class="theme-toggle" title="Thème">
      <label id="theme-label">Sombre</label>

      <div class="switch" id="theme-switch-wrapper" aria-checked="false">
        <input id="theme-switch" type="checkbox">
        <div class="track"></div>
        <div class="thumb" id="theme-thumb">
          <svg id="icon-moon" fill="currentColor" viewBox="0 0 24 24"><path
d="M21 12.79A9 9 0 1 1 11.21 3 7 7 0 1 0 21 12.79z"/></svg>
          <svg id="icon-sun" fill="currentColor" viewBox="0 0 24 24"
style="display:none"><path d="M6.76 4.84l-1.8-1.79-1.41 1.41 1.79 1.8
1.42-1.42zM1 13h3v-2H1v2zm10-9h2V1h-2v3zm7.04 1.46l1.79-1.8-1.41-1.41-1.8
1.79 1.42 1.42zM17 13h3v-2h-3v2zM4.96 18.54l-1.79 1.8 1.41 1.41
1.8-1.79-1.42-1.42zM11 23h2v-3h-2v3zm7.24-4.84l1.8 1.79
1.41-1.41-1.79-1.8-1.42 1.42zM12 7a5 5 0 1 0 0 10 5 5 0 0 0 0-10z"/></svg>
        </div>
      </div>
    </div>

    <h1>Gestion AD</h1>

    <!-- Form Ajout -->
    <form method="post" action="{{ url_for('send_info') }}">
      <input type="text" name="nom" placeholder="Nom *" required>
      <input type="text" name="password" placeholder="Mot de passe *"
required>
      <button class="btn">Ajouter utilisateur</button>
    </form>

    <hr style="margin:20px 0; border:1px solid var(--border);">

    <!-- Form Suppression -->
    <form method="post" action="{{ url_for('delete_info') }}">
      <input type="text" name="nom" placeholder="Nom à supprimer *" required>
      <button class="btn" style="background:var(--danger);
border-color:var(--danger-700); color:#250a0a;">
        Supprimer utilisateur
      </button>
    </form>
  </div>
```

```
  <!-- ====== SCRIPT THEME (identique à ton index) ====== -->
  <script>
    (function(){
      const root = document.documentElement;
      const input   = document.getElementById('theme-switch');
      const label   = document.getElementById('theme-label');
      const iconSun = document.getElementById('icon-sun');
      const iconMoon= document.getElementById('icon-moon');
      const switchEl= document.getElementById('theme-switch-wrapper');
      const thumb   = document.getElementById('theme-thumb');

      const saved = localStorage.getItem('theme');
      if (saved === 'light' || saved === 'dark')
root.setAttribute('data-theme', saved);
      else root.setAttribute('data-theme','auto');

      const theme = (() => {
        const mode = root.getAttribute('data-theme');
        if (mode === 'light' || mode === 'dark') return mode;
        return window.matchMedia('(prefers-color-scheme:light)').matches ?
'light' : 'dark';
      })();
      input.checked = (theme === 'light');
      updateUI(theme);

      switchEl.addEventListener('click', ()=>{
        input.checked = !input.checked;
        applyTheme();
      });

      function applyTheme(){
        const next = input.checked ? 'light' : 'dark';
        root.setAttribute('data-theme', next);
        localStorage.setItem('theme', next);
        updateUI(next);
      }
      function updateUI(mode){
        label.textContent = mode === 'light' ? 'Clair' : 'Sombre';
        iconSun.style.display = mode === 'light' ? 'block' : 'none';
        iconMoon.style.display = mode === 'light' ? 'none' : 'block';
      }
    })();
  </script>
</body>
</html>
```

console_redirect.html

```html
<!DOCTYPE html>
<html lang="fr">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Ouverture de la console...</title>
    <style>
        * { box-sizing: border-box; margin: 0; padding: 0; }
        body {
            font-family: -apple-system, BlinkMacSystemFont, "Segoe UI",
Roboto, "Helvetica Neue", Arial, sans-serif;
            background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);
            display: flex;
            align-items: center;
            justify-content: center;
            min-height: 100vh;
            color: #fff;
        }
        .loader-container {
            text-align: center;
            padding: 40px;
            background: rgba(255, 255, 255, 0.1);
            backdrop-filter: blur(10px);
            border-radius: 20px;
            box-shadow: 0 8px 32px rgba(0, 0, 0, 0.3);
            border: 1px solid rgba(255, 255, 255, 0.18);
        }
        .spinner {
            width: 60px;
            height: 60px;
            margin: 0 auto 20px;
            border: 4px solid rgba(255, 255, 255, 0.3);
            border-top-color: #fff;
            border-radius: 50%;
            animation: spin 1s linear infinite;
        }
        @keyframes spin {
            to { transform: rotate(360deg); }
        }
        h1 {
            font-size: 24px;
            margin-bottom: 10px;
            font-weight: 600;
        }
        p {
            font-size: 16px;
            opacity: 0.9;
            margin-bottom: 5px;
        }
        .status {
```

```
            margin-top: 20px;
            padding: 10px 20px;
            background: rgba(46, 213, 115, 0.2);
            border-radius: 8px;
            font-size: 14px;
            font-weight: 500;
        }
        .error {
            background: rgba(255, 107, 107, 0.2);
        }
    </style>
</head>
<body>
    <div class="loader-container">
        <div class="spinner"></div>
        <h1>Ouverture de la console</h1>
        <p>Configuration de l'authentification...</p>
        <div id="status" class="status">Injection du cookie
d'authentification</div>
    </div>

    <script>
        // Fonction pour définir un cookie
        function setCookie(name, value, domain, path = '/', secure = true) {
            let cookieString = `${name}=${encodeURIComponent(value)};
path=${path}`;

            if (domain) {
                cookieString += `; domain=${domain}`;
            }

            if (secure) {
                cookieString += '; secure';
            }

            // SameSite=None est nécessaire pour les cookies cross-site avec
secure
            cookieString += '; SameSite=None';

            document.cookie = cookieString;
            console.log('Cookie défini:', cookieString);
        }

        // Récupération des données depuis le template Flask
        const ticket = "{{ ticket }}";
        const consoleUrl = "{{ console_url }}";
        const proxmoxHost = "{{ proxmox_host }}";

        const statusEl = document.getElementById('status');
```

```
        try {
            // Injection du cookie PVEAuthCookie
            // On essaie plusieurs variantes pour maximiser les chances de
succès

            // 1. Cookie avec le domaine Proxmox exact
            setCookie('PVEAuthCookie', ticket, proxmoxHost, '/', true);

            // 2. Cookie avec le domaine parent (au cas où)
            const domainParts = proxmoxHost.split('.');
            if (domainParts.length > 2) {
                const parentDomain = '.' + domainParts.slice(-2).join('.');
                setCookie('PVEAuthCookie', ticket, parentDomain, '/', true);
            }

            // 3. Cookie sans domaine spécifique (pour le domaine actuel)
            setCookie('PVEAuthCookie', ticket, null, '/', true);

            console.log('Ticket Proxmox injecté avec succès');
            console.log('Ticket:', ticket.substring(0, 50) + '...');
            console.log('URL de la console:', consoleUrl);

            statusEl.textContent = '✓ Cookie configuré avec succès';
            statusEl.classList.remove('error');

            // Petite pause pour s'assurer que le cookie est bien enregistré
            setTimeout(() => {
                statusEl.textContent = 'Redirection vers la console...';

                // Redirection vers NoVNC dans une nouvelle fenêtre
                // On utilise window.open pour éviter les problèmes de
cookies cross-site
                const consoleWindow = window.open(consoleUrl, '_blank',
'noopener,noreferrer');

                if (consoleWindow) {
                    // Attendre un peu puis fermer la page actuelle ou
rediriger
                    setTimeout(() => {
                        window.close(); // Fermer l'onglet actuel si possible
                        // Si window.close() ne fonctionne pas (navigateur
bloque), on redirige
                        setTimeout(() => {
                            window.location.href = '/machines';
                        }, 500);
                    }, 1500);
                } else {
                    // Si le popup a été bloqué, on fait une redirection
normale
                    statusEl.textContent = '⚠ Popup bloqué, redirection en
```

```
cours...';
                    setTimeout(() => {
                        window.location.href = consoleUrl;
                    }, 1000);
                }
            }, 800);

        } catch (error) {
            console.error('Erreur lors de l\'injection du cookie:', error);
            statusEl.textContent = '✗ Erreur: ' + error.message;
            statusEl.classList.add('error');

            // Redirection de secours après 3 secondes
            setTimeout(() => {
                window.location.href = consoleUrl;
            }, 3000);
        }
    </script>
</body>
</html>
```

index.html

```
<!DOCTYPE html>
<html lang="fr" data-theme="auto">
<head>
  <meta charset="utf-8" />
  <title>Proxmox VMs</title>
  <meta name="viewport" content="width=device-width, initial-scale=1" />
  <style>
    /* ========= THEMES ========= */
    :root {
      /* sombre par défaut */
      --bg: #0f1115;
      --panel: #12161c;
      --muted: #8892a6;
      --text: #e6e8ee;
      --text-strong:#ffffff;
      --border:#1e2430;
      --primary:#6c8cff;
      --primary-600:#5a78e6;
      --success:#2ecc71;
      --success-700:#29b765;
      --warning:#f0c93d;
      --danger:#ff6b6b;
      --danger-700:#e45a5a;
      --badge:#1a2030;
      --row: #121826;
      --row-alt:#0f1420;
      --shadow: 0 10px 30px rgba(0,0,0,.35);
```

```
  --radius:16px;
  --gap: 22px;
}
html[data-theme="light"] {
  --bg:#f5f7fb;
  --panel:#ffffff;
  --text:#171a21;
  --text-strong:#0a0c10;
  --muted:#58627a;
  --border:#e8ecf3;
  --primary:#4b6bff;
  --primary-600:#3d59e6;
  --success:#19b56b;
  --success-700:#15a260;
  --warning:#e9bf28;
  --danger:#f25555;
  --danger-700:#d94949;
  --badge:#f2f5ff;
  --row:#ffffff;
  --row-alt:#fafbfe;
  --shadow: 0 10px 30px rgba(14,23,38,.08);
}
@media (prefers-color-scheme: light) {
  html[data-theme="auto"] {
    --bg:#f5f7fb;
    --panel:#ffffff;
    --text:#171a21;
    --text-strong:#0a0c10;
    --muted:#58627a;
    --border:#e8ecf3;
    --primary:#4b6bff;
    --primary-600:#3d59e6;
    --success:#19b56b;
    --success-700:#15a260;
    --warning:#e9bf28;
    --danger:#f25555;
    --danger-700:#d94949;
    --badge:#f2f5ff;
    --row:#ffffff;
    --row-alt:#fafbfe;
    --shadow: 0 10px 30px rgba(14,23,38,.08);
  }
}

/* ========= BASE ========= */
*{box-sizing:border-box}
html,body{height:100%}
body{
  margin:0;
  font-family: ui-sans-serif, system-ui, -apple-system, Segoe UI, Roboto,
```

```
"Helvetica Neue", Arial, "Noto Sans", "Apple Color Emoji","Segoe UI Emoji";
      background:
        radial-gradient(1200px 800px at 10% -10%, color-mix(in oklab,
var(--primary) 12%, transparent), transparent 60%),
        radial-gradient(900px 700px at 110% 0%, color-mix(in oklab,
var(--success) 10%, transparent), transparent 60%),
        var(--bg);
      color:var(--text);
      line-height:1.35;
      transition: background .25s ease, color .25s ease;
    }
    .container{ max-width:1200px; margin:48px auto; padding:0 22px; }

    header{
      display:flex; align-items:center; justify-content:space-between;
      margin-bottom:28px; gap:16px;
    }
    h1{
      margin:0; font-size: clamp(22px, 2.2vw, 32px);
      color:var(--text-strong); letter-spacing:.3px;
    }
    .actions{ display:flex; align-items:center; gap:14px; }

    a.link{
      color:var(--primary); text-decoration:none; font-weight:600;
      padding:10px 14px; background: color-mix(in oklab, var(--primary) 12%,
transparent);
      border:1px solid color-mix(in oklab, var(--primary) 40%,
var(--border));
      border-radius:12px; transition:.2s ease;
    }
    a.link:hover{ transform: translateY(-1px); }

    /* ========= SWITCH THEME ========= */
    .theme-toggle{
      display:flex; align-items:center; gap:10px;
      padding:6px 10px; border-radius:12px;
      background:var(--panel); border:1px solid var(--border);
      box-shadow: var(--shadow);
      user-select:none;
    }
    .theme-toggle label{ font-size:13px; color:var(--muted);
user-select:none; }

    .switch{
      --w: 54px; --h: 30px; --pad: 3px;
      position:relative; width:var(--w); height:var(--h);
      touch-action: pan-x;
      cursor:pointer;
```

```
    }
    .switch input{ position:absolute; opacity:0; inset:0; }
    .track{
      position:absolute; inset:0;
      background: color-mix(in oklab, var(--primary) 14%, var(--badge));
      border:1px solid var(--border);
      border-radius:999px; transition: background .25s ease, border .25s
ease;
    }
    .thumb{
      position:absolute; top:var(--pad); left:var(--pad);
      width: calc(var(--h) - var(--pad)*2); height: calc(var(--h) -
var(--pad)*2);
      border-radius:50%; background: var(--panel);
      border:1px solid var(--border);
      box-shadow: 0 4px 14px rgba(0,0,0,.18);
      transition: transform .25s ease, background .25s ease, border .25s
ease;
      display:grid; place-items:center; overflow:hidden;
      will-change: transform;
    }
    .thumb.dragging{ transition:none; }
    .thumb svg{ width:16px; height:16px; }
    .switch input:checked ~ .thumb{ transform: translateX(calc(var(--w) -
var(--h))); }

    /* ========= FLASH ========= */
    .flash-wrap{ margin: 0 0 18px 0; display:grid; gap:10px;}
    .flash{
      padding:12px 14px; border-radius:12px; border:1px solid var(--border);
      background:var(--panel); box-shadow: var(--shadow);
      display:flex; align-items:center; gap:10px; font-weight:600;
    }
    .flash.success{ border-color: color-mix(in oklab, var(--success) 40%,
var(--border)); background: color-mix(in oklab, var(--success) 12%,
var(--panel));}
    .flash.danger { border-color: color-mix(in oklab, var(--danger)  40%,
var(--border)); background: color-mix(in oklab, var(--danger)  12%,
var(--panel));}
    .flash.warning{ border-color: color-mix(in oklab, var(--warning) 40%,
var(--border)); background: color-mix(in oklab, var(--warning) 12%,
var(--panel));}

    /* ========= PANELS & TABLES ========= */
    .panel{
      background:var(--panel); border:1px solid var(--border);
      border-radius: var(--radius); box-shadow: var(--shadow);
overflow:hidden;
    }
    .panel-header{
```

```
      padding:16px 18px; border-bottom:1px solid var(--border);
      display:flex; align-items:center; justify-content:space-between;
gap:12px;
      background: linear-gradient(180deg, color-mix(in oklab, var(--primary)
12%, transparent), transparent 60%);
    }
    .panel-title{ margin:0; font-size: clamp(16px, 1.6vw, 20px);
color:var(--text-strong); }
    .table-wrap{ overflow:auto; }
    table{
      border-collapse:separate; border-spacing:0;
      width:100%; min-width: 760px; background:transparent;
    }
    th, td{ padding:14px 16px; text-align:left; border-bottom:1px solid
var(--border); vertical-align: middle; }
    th{
      position:sticky; top:0; z-index:1;
      background: linear-gradient(180deg, color-mix(in oklab, var(--panel)
85%, #000), var(--panel));
      color:var(--muted); font-size:13px; letter-spacing:.4px;
text-transform: uppercase;
    }
    tbody tr{ background:var(--row); transition: background .15s ease,
transform .15s ease; }
    tbody tr:nth-child(even){ background:var(--row-alt); }
    tbody tr:hover{ background: color-mix(in oklab, var(--primary) 8%,
var(--row)); }

    .badge{
      display:inline-flex; align-items:center; gap:8px; padding:6px 10px;
      font-size:12px; font-weight:700; border-radius:999px;
background:var(--badge);
      border:1px solid var(--border); letter-spacing:.2px;
    }
    .dot{ width:8px; height:8px; border-radius:50%; background:var(--muted);
}
    .is-running .dot{ background: var(--success); }
    .is-stopped .dot{ background: var(--danger); }
    .is-unknown .dot{ background: var(--warning); }

    .btn{
      appearance:none; border:none; cursor:pointer; padding:10px 14px;
font-weight:700;
      border-radius:12px; background:var(--badge); color:var(--text);
      border:1px solid var(--border);
      transition: transform .15s ease, box-shadow .15s ease, background .15s
ease;
      text-decoration: none;
      display: inline-block;
      text-align: center;
```

```
    }
    .btn:hover{ transform: translateY(-1px); box-shadow: 0 6px 20px
rgba(0,0,0,.18); }
    .btn:active{ transform: translateY(0); }
    .btn-primary{
      background: linear-gradient(180deg, color-mix(in oklab, var(--primary)
18%, transparent), var(--primary));
      color:#fff; border-color: color-mix(in oklab, var(--primary) 65%,
var(--border));
    }
    .btn-primary:hover{ background: var(--primary-600); }
    .btn-success{
      background: linear-gradient(180deg, color-mix(in oklab, var(--success)
18%, transparent), var(--success));
      color:#05240f; border-color: color-mix(in oklab, var(--success) 65%,
var(--border));
    }
    .btn-success:hover{ background: var(--success-700); color:#041e0d; }
    .btn-danger{
      background: linear-gradient(180deg, color-mix(in oklab, var(--danger)
18%, transparent), var(--danger));
      color:#2a0b0b; border-color: color-mix(in oklab, var(--danger) 65%,
var(--border));
    }
    .btn-danger:hover{ background: var(--danger-700); color:#210808; }

    .stack{ display:flex; align-items:center; gap:10px; flex-wrap:wrap; }

    form{ display:inline; }
    input[type="text"]{
      background:var(--row); border:1px solid var(--border);
color:var(--text);
      padding:10px 12px; border-radius:10px; outline:none;
      transition: border .15s ease, box-shadow .15s ease, background .15s
ease; min-width: 200px;
    }
    input[type="text"]::placeholder{ color:color-mix(in oklab, var(--muted)
70%, var(--text)); }
    input[type="text"]:focus{
      border-color: color-mix(in oklab, var(--primary) 60%, var(--border));
      box-shadow: 0 4px 14px color-mix(in oklab, var(--primary) 12%,
transparent);
      background: color-mix(in oklab, var(--primary) 6%, var(--row));
    }

    .section{ margin-bottom: var(--gap); }
    .table-footer{ padding:14px 18px; color:var(--muted); font-size:13px; }

    /* ========= MODAL DE CONFIRMATION ========= */
```

```css
.modal-overlay {
  display: none;
  position: fixed;
  top: 0;
  left: 0;
  right: 0;
  bottom: 0;
  background: rgba(0, 0, 0, 0.75);
  z-index: 1000;
  animation: fadeIn 0.2s ease;
}
.modal-overlay.active {
  display: flex;
  align-items: center;
  justify-content: center;
}
.modal {
  background: var(--panel);
  border: 1px solid var(--border);
  border-radius: var(--radius);
  box-shadow: 0 20px 60px rgba(0,0,0,.5);
  padding: 24px;
  max-width: 450px;
  width: 90%;
  animation: slideUp 0.3s ease;
}
.modal-title {
  margin: 0 0 12px 0;
  color: var(--text-strong);
  font-size: 20px;
}
.modal-content {
  color: var(--text);
  margin-bottom: 20px;
  line-height: 1.5;
}
.modal-actions {
  display: flex;
  gap: 10px;
  justify-content: flex-end;
}
@keyframes fadeIn {
  from { opacity: 0; }
  to { opacity: 1; }
}
@keyframes slideUp {
  from { transform: translateY(20px); opacity: 0; }
  to { transform: translateY(0); opacity: 1; }
}
```

```
    @media (max-width: 720px){
      .actions{ width:100%; justify-content:flex-end; }
      input[type="text"]{ width:100%; min-width: 0; }
      .btn{ width:100%; }
      .theme-toggle{ width:100%; justify-content:space-between; }
    }
  </style>
</head>
<body>
  <div class="container">
    <header>
      <h1>Machines Virtuelles</h1>

      <div class="actions">
        <!-- SWITCH -->
        <div class="theme-toggle" title="Basculer clair/sombre">
          <label id="theme-label" for="theme-switch">Sombre</label>
          <div class="switch" id="theme-switch-wrapper" role="switch"
aria-checked="false" aria-labelledby="theme-label">
            <input id="theme-switch" type="checkbox" aria-label="Basculer le
thème">
            <div class="track"></div>
            <div class="thumb" id="theme-thumb" aria-hidden="true">
              <svg id="icon-moon" viewBox="0 0 24 24"
fill="currentColor"><path d="M21 12.79A9 9 0 1 1 11.21 3 7 7 0 1 0 21
12.79z"/></svg>
              <svg id="icon-sun" viewBox="0 0 24 24" fill="currentColor"
style="display:none"><path d="M6.76 4.84l-1.8-1.79-1.41 1.41 1.79 1.8
1.42-1.42zM1 13h3v-2H1v2zm10-9h2V1h-2v3zm7.04 1.46l1.79-1.8-1.41-1.41-1.8
1.79 1.42 1.42zM17 13h3v-2h-3v2zM4.96 18.54l-1.79 1.8 1.41 1.41
1.8-1.79-1.42-1.42zM11 23h2v-3h-2v3zm7.24-4.84l1.8 1.79
1.41-1.41-1.79-1.8-1.42 1.42zM12 7a5 5 0 1 0 0 10 5 5 0 0 0 0-10z"/></svg>
            </div>
          </div>
        </div>

        <!-- Déconnexion -->
        <a class="link" href="/logout">Déconnexion</a>
      </div>

      {% if session['user'] == "administrateur" %}
      <div class="mb-4">
          <a href="{{ url_for('create_user') }}" class="btn btn-primary">
              Gestion User Active Directory
          </a>
      </div>
      {% endif %}

    </header>
```

```
    <!-- Messages Flash -->
    <div class="flash-wrap">
      {% with messages = get_flashed_messages(with_categories=true) %}
        {% if messages %}
          {% for category, message in messages %}
            <div class="flash {{ category }}">{{ message }}</div>
          {% endfor %}
        {% endif %}
      {% endwith %}
    </div>

    <!-- Tableau des VMs -->
    <section class="section panel">
      <div class="panel-header">
        <h2 class="panel-title">Instances</h2>
        <div class="stack" aria-hidden="true">
          <span class="badge"><span class="dot"
style="background:var(--success)"></span> Running</span>
          <span class="badge"><span class="dot"
style="background:var(--danger)"></span> Stopped</span>
          <span class="badge"><span class="dot"
style="background:var(--warning)"></span> Inconnu</span>
        </div>
      </div>
      <div class="table-wrap">
        <table role="table" aria-label="Machines virtuelles">
          <thead>
            <tr>
              <th>ID</th>
              <th>Nom</th>
              <th>Propriétaire</th>
              <th>Status</th>
              <th>Action</th>
            </tr>
          </thead>
          <tbody>
          {% for vm in vms %}
            {% set status_txt = vm.proxmox_raw.status if
vm.proxmox_raw.status else 'Inconnu' %}
            {% set status_class = 'is-running' if status_txt == 'running'
else ('is-stopped' if status_txt == 'stopped' else 'is-unknown') %}
            <tr>
              <td>{{ vm.vmid }}</td>
              <td>{{ vm.name }}</td>
              <td>{{ vm.owner_name or 'Inconnu' }}</td>
              <td>
                <span class="badge {{ status_class }}">
                  <span class="dot"></span>{{ status_txt|capitalize }}
                </span>
```

```
              </td>
              <td>
                <div class="stack">
                {% if vm.proxmox_raw.status != 'running' %}
                  <form action="/start_vm" method="POST">
                    <input type="hidden" name="vmid" value="{{ vm.vmid }}">
                    <button type="submit" class="btn
btn-success">Démarrer</button>
                  </form>
                  <!-- NOUVEAU BOUTON SUPPRIMER (uniquement si VM arrêtée)
-->
                  <button type="button" class="btn btn-danger"
onclick="confirmDelete({{ vm.vmid }}, '{{ vm.name }}')">
                    Supprimer
                  </button>
                {% else %}
                  <form action="/stop_vm" method="POST">
                    <input type="hidden" name="vmid" value="{{ vm.vmid }}">
                    <button type="submit" class="btn
btn-danger">Arrêter</button>
                  </form>
                  <a class="btn btn-primary" href="{{ url_for('open_console',
vmid=vm.vmid) }}" target="_blank" rel="noopener">
                    Ouvrir Console
                  </a>
                {% endif %}
                </div>
              </td>
            </tr>
          {% endfor %}
          </tbody>
        </table>
      </div>
      <div class="table-footer">Total: {{ vms|length }} VM(s)</div>
    </section>

    <!-- Tableau des Templates -->
    <section class="section panel">
      <div class="panel-header">
        <h2 class="panel-title">Templates</h2>
      </div>
      <div class="table-wrap">
        <table role="table" aria-label="Templates">
          <thead>
            <tr>
              <th>ID</th>
              <th>Nom</th>
              <th>Propriétaire</th>
              <th>Cloner en VM</th>
            </tr>
```

```
          </thead>
          <tbody>
          {% for tmpl in templates %}
            <tr>
              <td>{{ tmpl.vmid }}</td>
              <td>{{ tmpl.name }}</td>
              <td>{{ tmpl.owner_name or 'Inconnu' }}</td>
              <td>
                <div class="stack" style="flex-wrap:nowrap;">
                  <form action="/clone_template" method="POST"
style="display:flex; gap:10px; align-items:center; width:100%;">
                    <input type="hidden" name="template_id" value="{{
tmpl.vmid }}">
                    <input type="text" name="new_name" placeholder="Nom de la
VM" required>
                    <button type="submit" class="btn
btn-primary">Cloner</button>
                  </form>
                </div>
              </td>
            </tr>
          {% endfor %}
          </tbody>
        </table>
      </div>
      <div class="table-footer">Total: {{ templates|length }}
template(s)</div>
    </section>
  </div>

  <!-- MODAL DE CONFIRMATION DE SUPPRESSION -->
  <div class="modal-overlay" id="deleteModal">
    <div class="modal">
      <h3 class="modal-title">⚠️ Confirmer la suppression</h3>
      <div class="modal-content">
        <p>Êtes-vous sûr de vouloir supprimer la VM <strong
id="vmName"></strong> (ID: <strong id="vmId"></strong>) ?</p>
        <p style="color: var(--danger);">Cette action est irréversible et
supprimera tous les volumes associés.</p>
      </div>
      <div class="modal-actions">
        <button class="btn" onclick="closeModal()">Annuler</button>
        <form id="deleteForm" action="/delete_vm" method="POST"
style="display:inline;">
          <input type="hidden" name="vmid" id="deleteVmId">
          <button type="submit" class="btn btn-danger">Supprimer
définitivement</button>
        </form>
      </div>
    </div>
```

```
  </div>

  <script>
    // ========= GESTION DU THEME =========
    (function(){
      const root = document.documentElement;
      const input   = document.getElementById('theme-switch');
      const label   = document.getElementById('theme-label');
      const iconSun = document.getElementById('icon-sun');
      const iconMoon= document.getElementById('icon-moon');
      const switchEl= document.getElementById('theme-switch-wrapper');
      const thumb   = document.getElementById('theme-thumb');

      const saved = localStorage.getItem('theme');
      if (saved === 'light' || saved === 'dark') {
        root.setAttribute('data-theme', saved);
      } else {
        root.setAttribute('data-theme', 'auto');
      }

      const effectiveTheme = (() => {
        const mode = root.getAttribute('data-theme');
        if (mode === 'light' || mode === 'dark') return mode;
        return window.matchMedia('(prefers-color-scheme: light)').matches ?
'light' : 'dark';
      })();
      input.checked = (effectiveTheme === 'light');
      updateUI(effectiveTheme);
      syncThumbToChecked();

      switchEl.addEventListener('click', (e) => {
        if (dragState.wasDragging) {
          dragState.wasDragging = false;
          return;
        }
        input.checked = !input.checked;
        applyThemeFromInput();
      });

      input.addEventListener('change', applyThemeFromInput);

      const mql = window.matchMedia('(prefers-color-scheme: light)');
      mql.addEventListener?.('change', e => {
        const savedNow = localStorage.getItem('theme');
        if (!savedNow) {
          const now = e.matches ? 'light' : 'dark';
          root.setAttribute('data-theme', 'auto');
          input.checked = e.matches;
          updateUI(now);
```

```
        syncThumbToChecked();
      }
    });

    function applyThemeFromInput(){
      const next = input.checked ? 'light' : 'dark';
      root.setAttribute('data-theme', next);
      localStorage.setItem('theme', next);
      updateUI(next);
      syncThumbToChecked();
    }
    function updateUI(mode){
      label.textContent = mode === 'light' ? 'Clair' : 'Sombre';
      const isLight = (mode === 'light');
      iconSun.style.display = isLight ? 'block' : 'none';
      iconMoon.style.display = isLight ? 'none'  : 'block';
      switchEl.setAttribute('aria-checked', String(isLight));
    }
    function syncThumbToChecked(){
      const rect = switchEl.getBoundingClientRect();
      const max = rect.width - rect.height;
      thumb.style.transition = '';
      thumb.style.transform = input.checked ? `translateX(${max}px)` :
'translateX(0)';
    }

    const dragState = {
      dragging:false,
      startX:0,
      startChecked:false,
      max:0,
      base:0,
      wasDragging:false
    };

    const startDrag = (clientX) => {
      const rect = switchEl.getBoundingClientRect();
      dragState.dragging = true;
      dragState.startX = clientX;
      dragState.startChecked = input.checked;
      dragState.max = rect.width - rect.height;
      dragState.base = input.checked ? dragState.max : 0;
      dragState.wasDragging = false;
      thumb.classList.add('dragging');
    };
    const moveDrag = (clientX) => {
      if (!dragState.dragging) return;
      const dx = clientX - dragState.startX;
      let next = Math.max(0, Math.min(dragState.max, dragState.base + dx));
      thumb.style.transform = `translateX(${next}px)`;
```

```
    dragState.wasDragging = true;
  };
  const endDrag = () => {
    if (!dragState.dragging) return;
    const style = getComputedStyle(thumb);
    const matrix = new DOMMatrixReadOnly(style.transform);
    const current = matrix.m41 || 0;
    const nextChecked = current > dragState.max / 2;
    dragState.dragging = false;
    thumb.classList.remove('dragging');
    input.checked = nextChecked;
    applyThemeFromInput();
    setTimeout(() => { dragState.wasDragging = false; }, 50);
  };

  thumb.addEventListener('mousedown', (e)=>{
    e.preventDefault();
    startDrag(e.clientX);
    const onMove = (ev)=> moveDrag(ev.clientX);
    const onUp   = ()=>{
      window.removeEventListener('mousemove', onMove);
      window.removeEventListener('mouseup', onUp);
      endDrag();
    };
    window.addEventListener('mousemove', onMove, {passive:true});
    window.addEventListener('mouseup', onUp, {passive:true});
  });

  thumb.addEventListener('touchstart', (e)=>{
    const t = e.changedTouches[0];
    startDrag(t.clientX);
  }, {passive:true});
  thumb.addEventListener('touchmove', (e)=>{
    const t = e.changedTouches[0];
    moveDrag(t.clientX);
  }, {passive:true});
  thumb.addEventListener('touchend', (e)=>{
    endDrag();
  }, {passive:true});
  thumb.addEventListener('touchcancel', ()=>{
    endDrag();
  }, {passive:true});
})();

// ========= GESTION DE LA MODAL DE SUPPRESSION =========
function confirmDelete(vmid, vmname) {
  document.getElementById('vmId').textContent = vmid;
  document.getElementById('vmName').textContent = vmname;
  document.getElementById('deleteVmId').value = vmid;
  document.getElementById('deleteModal').classList.add('active');
```

```
    }

    function closeModal() {
      document.getElementById('deleteModal').classList.remove('active');
    }

    // Fermer la modal en cliquant en dehors
    document.getElementById('deleteModal').addEventListener('click',
function(e) {
      if (e.target === this) {
        closeModal();
      }
    });

    // Fermer avec la touche Échap
    document.addEventListener('keydown', function(e) {
      if (e.key === 'Escape') {
        closeModal();
      }
    });
  </script>
</body>
</html>
```

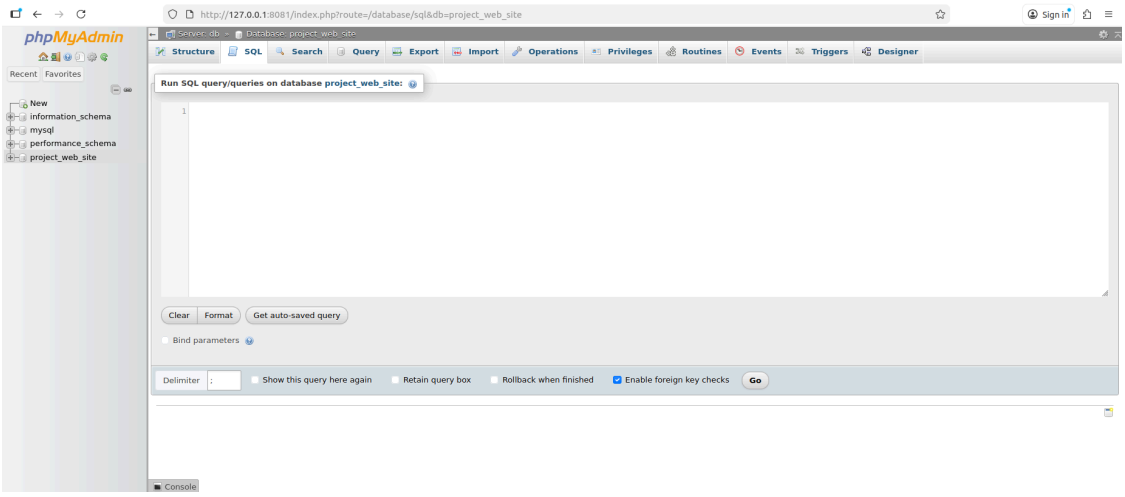
login.html

```
<!DOCTYPE html>
<html lang="fr" data-theme="auto">
<head>
  <meta charset="UTF-8">
  <title>Login RADIUS</title>
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
```

Start docker compose :

```
docker compose build
docker compose up
```

At the first start, you have to create table in the database, you log in the web interface of the phpmyadmin (http://YOUR_IP:8081), and create the table in project_web_site database :



ad_interface

add the following code :

SQL

```sql
CREATE TABLE `users` (
  `id` int(11) NOT NULL,
  `username` varchar(100) NOT NULL,
  `role` enum('user','admin') DEFAULT 'user'
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;

--
-- Dumping data for table `users`
--

INSERT INTO `users` (`id`, `username`, `role`) VALUES
(1, 'administrateur', 'admin');

-- --------------------------------------------------------

--
-- Table structure for table `vms`
--

CREATE TABLE `vms` (
  `id` int(11) NOT NULL,
  `vmid` int(11) NOT NULL,
  `name` varchar(255) DEFAULT NULL,
  `owner_id` int(11) DEFAULT NULL,
  `is_template` tinyint(1) DEFAULT 0
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;

--
-- Indexes for dumped tables
```

```
--


--
-- Indexes for table `users`
--
ALTER TABLE `users`
  ADD PRIMARY KEY (`id`),
  ADD UNIQUE KEY `username` (`username`);

--
-- Indexes for table `vms`
--
ALTER TABLE `vms`
  ADD PRIMARY KEY (`id`),
  ADD UNIQUE KEY `vmid` (`vmid`),
  ADD KEY `owner_id` (`owner_id`);

--
-- AUTO_INCREMENT for dumped tables
--


--
-- AUTO_INCREMENT for table `users`
--
ALTER TABLE `users`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=2;

--
-- AUTO_INCREMENT for table `vms`
--
ALTER TABLE `vms`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT;

--
-- Constraints for dumped tables
--


--
-- Constraints for table `vms`
--
ALTER TABLE `vms`
  ADD CONSTRAINT `vms_ibfk_1` FOREIGN KEY (`owner_id`) REFERENCES `users`
(`id`) ON DELETE SET NULL;
COMMIT;
```

When the tables are created, restart docker :

```
docker compose down
docker compose up
```

The interface is ready to use in http://YOUR_IP:5000