SAE 501 :
Mise en place d'une Infrastructure de TP Basée sur des Conteneurs

Groupe :
Florentin Czurka, Baptiste Duval, Michel Bauchart, Roman Negrila

Sommaire:

1.Introduction

2. Objectifs du projet

3. Choix techniques et justifications

4. Architecture générale

5. Fonctionnement détaillé

6. Sécurité

7. Maintenance et documentation

8. Interface utilisateur

9.Base de données

10. Conclusion

1. Introduction

Le projet consiste à créer une plateforme permettant de déployer et gérer des environnements de TP accessibles aux utilisateurs depuis un navigateur web. Ces environnements sont des machines virtuelles conteneurisées, offrant un isolement complet et une sécurité adaptée pour des travaux pratiques en développement web, bases de données et cybersécurité.

L’objectif principal est de proposer une infrastructure simple à utiliser, sécurisée et facilement maintenable, tout en permettant aux utilisateurs de créer, démarrer, arrêter ou supprimer leurs machines virtuelles sans intervention directe des administrateurs.

2. Objectifs du projet

Le projet consiste à mettre en place une infrastructure complète permettant d'héberger des Travaux Pratiques au sein d'environnements conteneurisés. Le premier objectif est d'installer et configurer un orchestrateur de conteneurs, tel que Docker Swarm ou Kubernetes, afin de pouvoir déployer, gérer et superviser plusieurs environnements de TP de manière automatisée et scalable.

Il s'agit ensuite de concevoir des images de conteneurs spécifiques aux besoins pédagogiques : environnements de développement web, bases de données pour l'apprentissage du SQL, ou encore machines dédiées aux exercices de cybersécurité. Chaque image doit être adaptée, optimisée et sécurisée.

Un autre objectif important est de mettre en place une interface web simple d'utilisation permettant aux étudiants et enseignants d'accéder facilement à leurs environnements de TP. Cette interface doit permettre la création, l'accès, l'arrêt, voire la réinitialisation des conteneurs de formation.

La sécurité constitue également une priorité : contrôle des accès utilisateurs, isolation des conteneurs, gestion des permissions, surveillance et monitoring (logs, métriques, alertes). L'idée est de garantir un environnement fiable, stable et sécurisé.

Enfin, le projet inclut la rédaction d'une documentation complète. Celle-ci doit couvrir les procédures d'installation, d'exploitation, de maintenance (backups, mises à jour) et de supervision afin de permettre une prise en main durable de la plateforme.

L'ensemble doit aboutir à une plateforme fonctionnelle, stable et bien documentée, présentant une réelle valeur pédagogique.

3. Choix techniques et justification

3.1 Conteneurisation : Docker

Docker permet de créer des conteneurs légers et isolés, contenant tout ce qui est nécessaire pour exécuter une machine virtuelle.

Avantages :

- **Rapidité** : les conteneurs se créent, se démarrent et se détruisent en quelques secondes, ce qui permet aux étudiants d'obtenir rapidement un environnement opérationnel.

- **Isolation** : chaque conteneur fonctionne de manière indépendante, ce qui limite les risques d'interférences entre TP et renforce la sécurité.

- **Portabilité** : un conteneur Docker fonctionne de la même manière sur n'importe quel serveur, ce qui assure une uniformité des environnements pour tous les utilisateurs.

- **Simplicité de versioning** : les images peuvent être versionnées, facilitant la mise à jour et la réplication des TP.

Inconvénients :

- **Gestion complexe à grande échelle** : lorsqu'un grand nombre de conteneurs doit être orchestré, Docker seul devient insuffisant.

- **Dépendance à un orchestrateur** : pour automatiser le déploiement, la répartition de charge ou la résilience, il est nécessaire d'ajouter une couche comme Kubernetes ou Docker Swarm.

3.2 Gestion graphique : Portainer

Portainer est utilisé pour superviser et gérer les conteneurs Docker via une interface web simple.

Avantages :

- Interface intuitive pour administrateurs et utilisateurs.

- Permet de visualiser l'état des conteneurs, logs et ressources consommées.

- Facile à installer et configurer.

3.3 Orchestration : écart de Kubernetes

Kubernetes a été écarté car il est plus adapté à des infrastructures massives. Sa complexité et ses besoins en ressources dépassent largement le cadre de notre SAE.

4. Architecture générale

L'architecture de la plateforme se compose de plusieurs couches :

1. **Backend :** Application Flask en Python gérant la création et le contrôle des conteneurs, et l'authentification des utilisateurs.

2. **Base de données :** MySQL pour stocker les informations utilisateurs, mots de passe hachés et rôles (admin/utilisateur).

3. **Front-end :** Pages HTML/CSS/JS pour l'interface utilisateur et l'interaction avec les conteneurs via l'API Flask.

4. **Conteneurs :** Ubuntu ou Kali Linux avec XFCE + noVNC pour accéder au bureau via navigateur.

4.1 Diagramme simplifié de l'architecture

Utilisateur
|
Navigateur
|
Flask App (Backend)
|
MySQL (authentification)
|
Docker Engine (contêneurs)
|
Portainer (supervision)

5. Fonctionnement détaillé

5.1 Authentification

Quand un utilisateur se connecte avec les identifiants qui sont dans la base de données, le script python regarde le rôle associé à l'utilisateur, si la personne qui se connecte a un rôle administrateur il aura accès à beaucoup plus de choses qu'un utilisateur normal.

La méthode d'authentification est très simple, avant que l'utilisateur ait accès à la page de base de notre site internet il est obligé de se connecter, c'est une question de sécurité pour ne pas qu'il y ait n'importe qui qui se connecte sur notre page.

Nous définissons ici "require_login" que nous allons ajouter dans la partie de l'index dans le flask, c'est donc grâce à cela que nous sommes obligés de nous connecter pour avoir accès à la page index.html.

```
def require_login(f):
    @wraps(f)
    def wrapper(*args, **kwargs):
        if "username" not in session:
            return redirect(url_for("login"))
        return f(*args, **kwargs)
    return wrapper
```

Dans la base de données les mots de passe sont stockés hachés avec bcrypt pour une sécurité totale à ce niveau, quand nous faisons une requête pour se connecter flask récupére le mot de passe normal et regarde s'ils sont bien identique.

5.2 Création des conteneurs

Ensuite quand nous sommes connectés nous avons le choix de déployer plusieurs machines différentes. Nous pouvons déployer une machine Kali, Ubunutu ou Windows, ce qui nous offre un large choix de systèmes d'exploitations et donc de TP à effectuer.

Quand un utilisateur veut déployer sa machine, Flask vérifie beaucoup de choses :

1. Il vérifie s' il trouve l'image de la machine que nous voulons créer.
2. Vérifie les ports disponibles pour éviter les conflits.
3. Génère un nom unique pour le conteneur.

4. Crée le conteneur Docker avec les paramètres spécifiés.

5. Démarre le conteneur et expose le port VNC pour l'accès via navigateur.

```
subprocess.run([
    "docker", "create",
    "--name", container_name,
    "-p", f"{port}:6080",
    "--shm-size=2g",
    "--restart", "unless-stopped",
    "ubuntu-novnc-clean:latest"
], check=True)
subprocess.run(["docker", "start", container_name])
```

Voici la partie du code qui nous sert à déployer notre machine virtuelle, donc on crée le container, on lui donne un nom, un port, une taille de mémoire maximale et des directive de redémarrage.

5.3 Accès au bureau via noVNC

Ici nous avons plusieurs choix pour pouvoir accéder au bureau noVNC, nous pouvions soit rester sur notre page web et avec accès en direct au bureau ou récupérer l'adresse IP et le port de la machine pour pouvoir se connecter depuis un autre navigateur ou page web si nous voulons utiliser plusieurs machines en même temps.

Avantages :

- Accès direct depuis un navigateur.
- Pas besoin d'installer un client VNC.
- Multi-utilisateurs possible grâce à l'isolation des conteneurs.

5.4 Gestion des conteneurs depuis l'interface

Grâce à notre interface web nous pouvons créer des machines virtuelles, démarrer, arrêter, supprimer ou redémarrer les machines virtuelles. Ce qui nous permet de faire ce que nous voulons avec nos machines sans contrainte.

Exemple :

```
function ctrl(id, action) {
```

```
    fetch(`/api/control/${id}/${action}`);
    setTimeout(refresh, 3000);
}
```

Voici par exemple un morceau de code qui me permet d'effectuer l'une de ces actions.

5.5 Rafraîchissement et affichage dynamique

En fond, l'interface web utilise du JavaScript pour la bonne fonctionnalité de l'interface.

Par exemple JavaScript est utilisé pour rafraîchir régulièrement l'état des conteneurs, ce qui permet à l'utilisateur de voir si sa machine est éteinte ou allumée. Nous utilisons aussi JavaScript aussi pour récupérer automatiquement le statut et c'est de là que nous voyons si la machine est en ligne et arrêtée. L'utilisation de ces deux codes JavaScript est complémentaire car l'un écrit le statut en ligne ou arrêtée et l'autre actualise la page automatiquement et c'est là que nous savons si oui ou non la machine est en ligne.

6. Sécurité

Au niveau de la sécurité, nous avons bien fait attention de bien stocker des mots de passe hashés et non des mots de passe en brut, ce qui permet une totale ignorance des mots de passe par quiconque lit la base de données.

Nous avons aussi une totale isolation des conteneurs, cela nous permet de séparer chaque machine des autres, que les données ne soient pas liées, qu'il n'y ai pas de possibilité d'atteindre les données d'une machine depuis une autre.

Les utilisateurs ont uniquement accès à leurs machines et non aux machines des autres ce qui permet une totale confidentialité sur l'utilisation des machines.

Il y a une gestion des logs depuis l'interface graphique de l'administrateur, il peut donc voir ce qu'il se passe et s'il y a des choses qui ne sont pas normales, gérer la machine d'un utilisateur ou la bloquer directement pour éviter les problèmes.

7. Maintenance et documentation

7.1 Sauvegardes

Nous pouvons effectuer une sauvegarde de la base de données pour garder dans un fichier une base de données à jour avec les logs et les utilisateurs, donc si nous perdons la base de données nous pouvons la reconstruire directement grâce à ce fichier.

```
mysqldump dockerlab > backup.sql #Commande qui permet d'importer une base de données dans un fichier.
```

Nous pouvons aussi faire une sauvegarde des images Docker, ce qui permet d'avoir une backup des machines qui ont été créées par les utilisateurs.

```
docker save -o ubuntu-novnc.tar ubuntu-novnc-clean:latest #Commande qui permet de sauvegarder les conteneurs Docker
```

7.2 Mise à jour

S'il y a un problème sur les images des machines ou s'il y a des versions nous n'avons qu'à effectuer un docker-compose, et grâce à un DockerFile des mises à jours ou des nouvelles versions des systèmes d'exploitations seront déployées dans nos images, les utilisateurs n'auront qu'à les déployer pour les utiliser.

7.3 Monitoring

Grâce à l'outil Portainer, nous avons une vision totale sur l'état des machines, sur les logs ce qui nous permet d'avoir une vue constante sur la santé du serveur pour qu'il n'y ai pas de problèmes. Nous avons aussi ajouté dans l'interface de l'administrateur un système de vision des logs en direct.

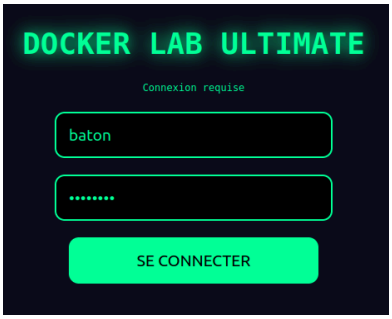
Grâce à cela l'administrateur voit en instantané ce qu'il ce passe et il a un relevé temporel de quand ça c'est passé, pour s'il y a un gros problème pouvoir utiliser une backup du serveur et ne rien perdre.

L'administrateur peut aussi voir les statistiques de l'ordinateur, il peut visualiser la RAM utilisée, le pourcentage de CPU utilisé, le nombre de container créé, il a vraiment une vision totale sur ce qu'il se passe pour éviter les problèmes de surconsommation de capacité du serveur.

8. Interface utilisateur

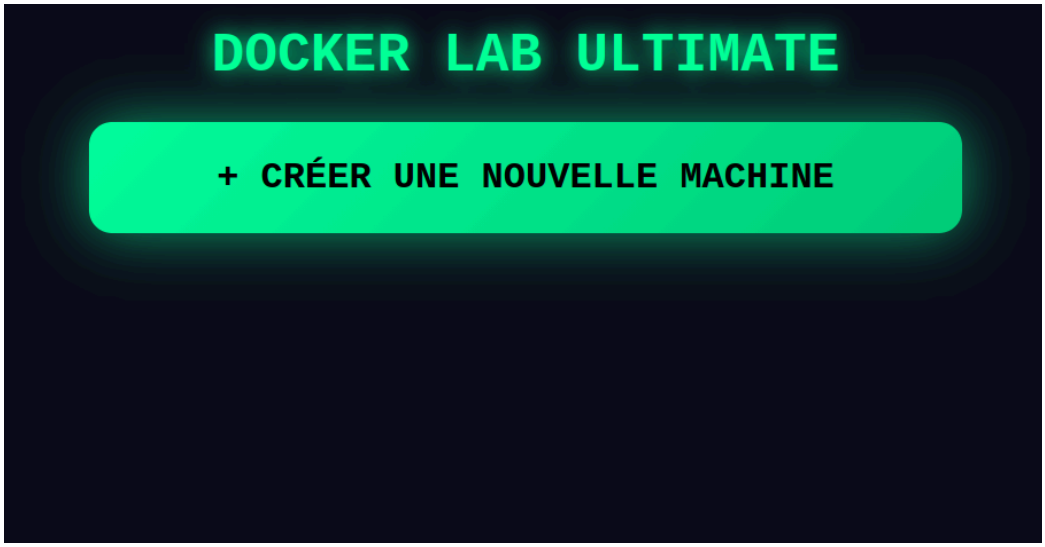
8.1 Utilisateur classique

Quand nous allons sur l'adresse IP du serveur nous arrivons directement sur cette page, il s'agit donc d'une page de connexion avec un stockage des identifiants dans une base de données.



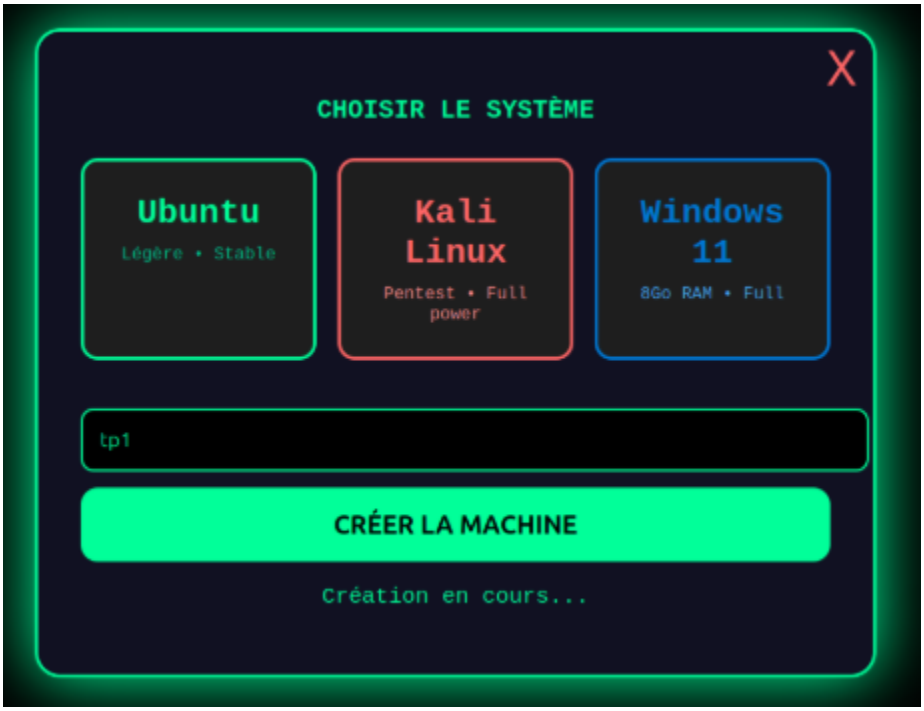
Page de connexion au site

Quand il se connecte il arrive maintenant dans une page avec un bouton pour créer une nouvelle machine et s'il en a déjà créé les machines qu'il avait déjà créées.



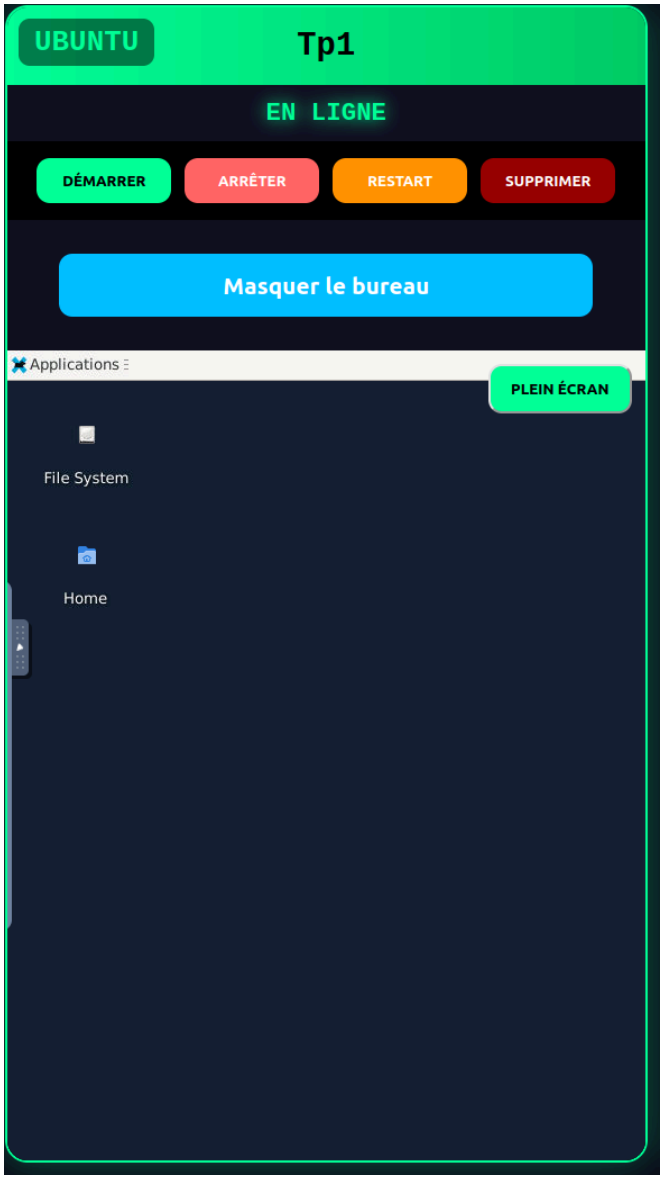
Bouton de création de VM

Quand l'utilisateur clique sur le bouton il a la possibilité de choisir le système qu'il aimerait utiliser sur la machine qu'il veut créer. Il a donc le choix entre Ubuntu, Kali et Windows, quand il clique sur l'un des trois systèmes qu'il veut créer il devra ensuite choisir le nom qu'il veut utiliser et une vérification est faite sur le nom pour vérifier qu'il n'y ai pas de doublons.



Création d'une machine virtuelle

Quand la machine est créée il n'a plus qu'à la démarrer pour pouvoir l'utiliser, quand il démarre la machine il n'a plus qu'à appuyer sur le bouton afficher le bureau et il pourra se connecter à l'interface graphique de la machine.

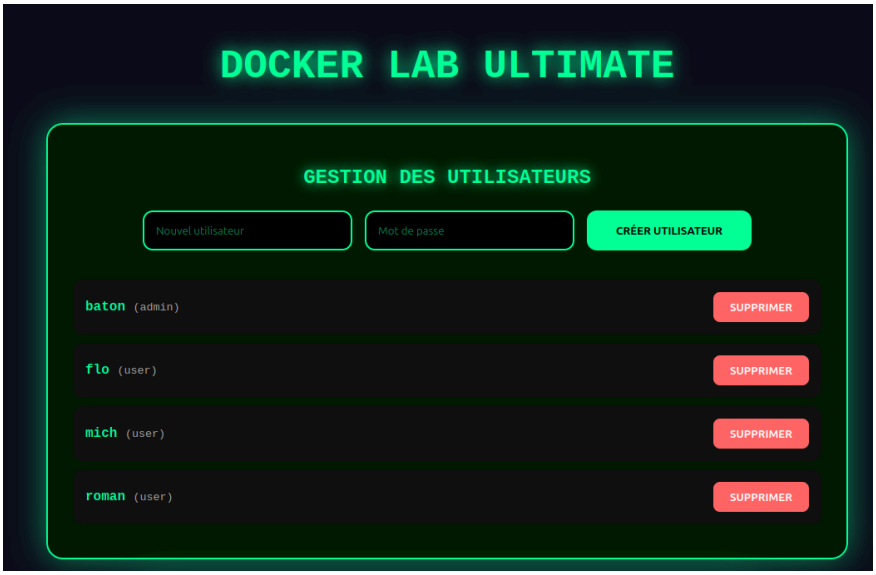


Exemple de bureau d'une machine virtuelle.

Donc voici l'interface graphique d'un utilisateur normal d'une salle de TP, il peut donc faire ce qu'il veut, créer des machines virtuelles, faire ce qu'il veut avec sans restreinte. Mais un utilisateur normal a uniquement accès à ses propres VM pour des questions de sécurité.

8.2 Utilisateur administrateur

Le principe de connexion est exactement le même que pour un utilisateur classique, mais quand il se connecte il a une interface différente. L'utilisateur administrateur à lui accès à toutes les machines des utilisateurs classiques, il a aussi en plus un tableau avec les utilisateurs présents dans la base de données, Il peut donc supprimer des utilisateurs et en créer autant qu'il le veut.



Gestion des utilisateurs

L'administrateur a aussi accès à une page où des logs sont présentes, il voit qui se connecte, qui se déconnecte; quand quelqu'un crée, supprime, démarre et redémarre ses machines.

Gestion des logs de l'administrateur

Il peut aussi avoir accès aux statistiques du serveur, le pourcentage de CPU utilisé, le pourcentage de la RAM utilisée et aussi le nombre de machines existantes.

L'administrateur peut aussi voir et avoir accès à toutes les machines virtuelles présentes dans le serveur, il peut donc voir ce que les utilisateurs font.

Interface de l'administrateur sur la gestion des machines

9. Base de données

Nous avons une petite base de données nommée “dockerlab” qui nous sert à stocker les utilisateurs et les logs qui sont écrit sur le site.

Structure :

dockerlab
│
├── users
│ │
│ ├── id (INT, PRIMARY KEY, AUTO_INCREMENT)
│ ├── username (VARCHAR) – identifiant unique
│ ├── password_hash (VARCHAR) – mot de passe hashé avec bcrypt
│ ├── role (ENUM ou VARCHAR) – "admin" ou "user"
│
└── logs
│
├── id (INT, PRIMARY KEY, AUTO_INCREMENT)
├── contenu (TEXT ou VARCHAR) – description du log / action
├── date (DATETIME) – date et heure de l’événement

Voici le contenu de l’une des tables présente dans la base de donnée:

```
mysql> select * from users;
+----+-----------+--------------------------------------------------------------+-------+---------------------+
| id | username  | password_hash                                                | role  | created_at          |
+----+-----------+--------------------------------------------------------------+-------+---------------------+
|  6 | baton     | $2b$12$ZeoZ/oSelMp9eb/jUZrHL.qFVuJ7se7NWOhIp/5ZnICgyvkCjO7uq | admin | 2025-11-19 14:56:40 |
|  7 | mich      | $2b$12$kD18VHKG/aShRpRhKvR91ehKOVN8xpf4JiVUhBF9UkmcwFEwPFsOe | user  | 2025-11-19 14:57:37 |
|  9 | roman     | $2b$12$3OmJsNCe1EpaAPyUCKzZoOciKw/3jD08ZzdXpikXy3VIbjmbpwMZS | user  | 2025-11-19 15:54:58 |
| 10 | flo       | $2b$12$oTE76oiztpOWTIWjq5q7I.Z7OzpmWXbDRUKrZqlfA7/nDXRcqj1ty | user  | 2025-11-20 14:12:33 |
| 13 | mrmerchez | $2b$12$AGTU5/XCE47Yxx81LFZ6ruw1HODt2QMBPIUAAfmT16Btc2I3Xic3C | user  | 2025-11-20 15:19:41 |
+----+-----------+--------------------------------------------------------------+-------+---------------------+
5 rows in set (0,00 sec)
```

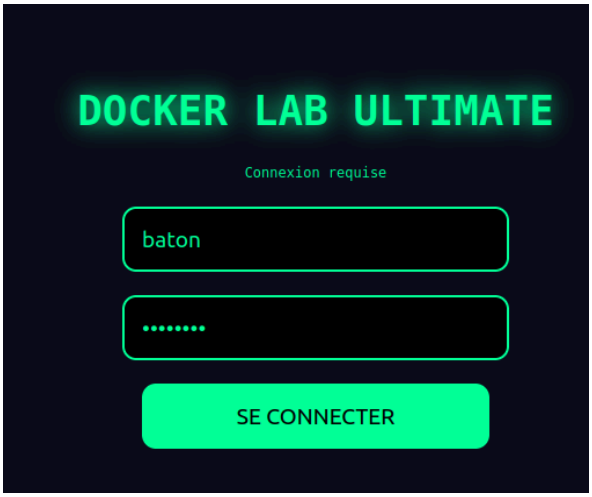
10. Conclusion

Ce projet a permis de concevoir et déployer une plateforme complète dédiée à la gestion d'environnements de travaux pratiques basés sur des conteneurs. Grâce à l'utilisation de Docker et d'une architecture simple mais robuste, nous avons pu fournir aux utilisateurs des machines virtuelles isolées, rapides à déployer et adaptées à différents besoins pédagogiques. L'intégration d'un backend Flask, d'une base de données MySQL, ainsi que d'une interface web intuitive offre une expérience fluide aussi bien pour les étudiants que pour les administrateurs.

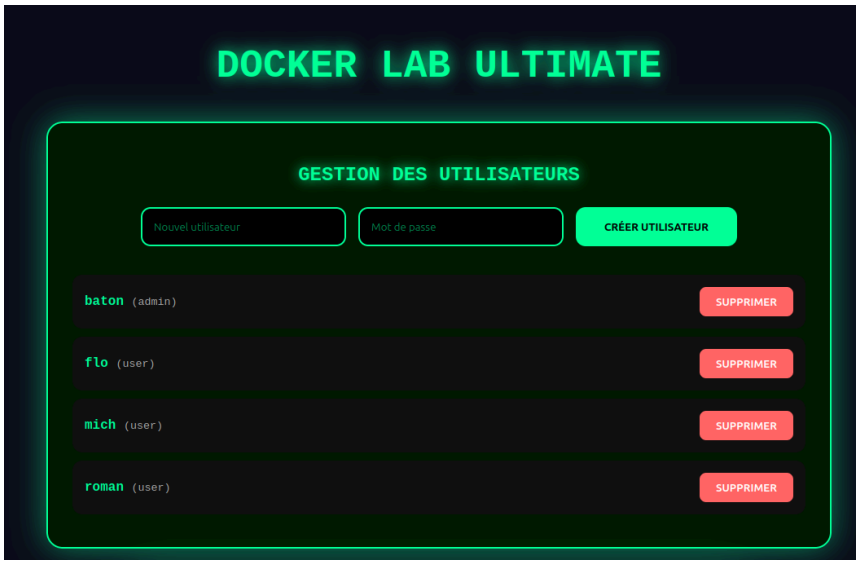
Les choix techniques réalisés, tels que l'utilisation de Portainer pour la supervision ou l'écartement de Kubernetes pour éviter une complexité inutile, ont contribué à rendre la solution à la fois fiable, performante et facilement maintenable. La mise en place d'un système d'authentification sécurisé, la gestion fine des rôles, l'isolation stricte des conteneurs et l'intégration d'outils de monitoring assurent un haut niveau de sécurité et de stabilité.

Enfin, la documentation, les outils de sauvegarde et les mécanismes de mise à jour garantissent la pérennité de la plateforme et permettent une prise en main rapide par d'éventuels futurs administrateurs. Cette infrastructure répond ainsi pleinement aux objectifs initiaux : proposer un environnement pédagogique moderne, automatisé, sécurisé et flexible, capable d'évoluer selon les besoins des enseignants et des étudiants.

Capture d’écran:



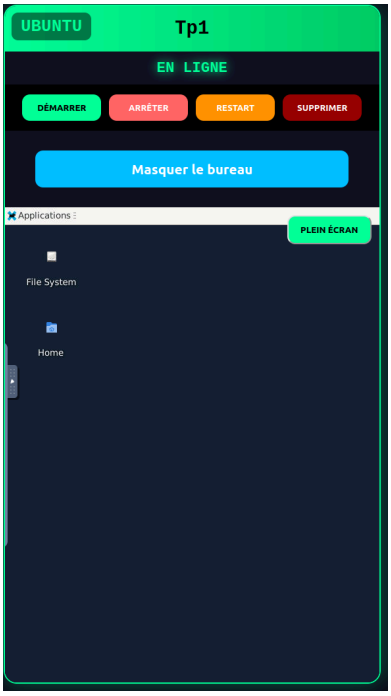
Page de connexion



Gestion des utilisateurs pour l’admin

Gestion des machines pour l’admin

Exemple de création d'une machine virtuelle



Interface d'une machine

Interface de logs pour l’administrateur

Scripts :

app.py

```
# app.py - VERSION FINALE : CHOIX UBUNTU/KALI (même image) + zéro bug +
admin voit tout
from flask import Flask, render_template, request, jsonify, session,
redirect, url_for, flash
import subprocess, re, mysql.connector, bcrypt, time
from functools import wraps

app = Flask(__name__)
app.secret_key = "CHANGE_ME_EN_SECRET_TRES_LONG_1234567890"

db_config = {
    'host': 'localhost',
    'user': 'dockerlab',
    'password': 'Progtr00',
    'database': 'dockerlab'
}

def get_db():
    return mysql.connector.connect(**db_config)

def require_login(f):
    @wraps(f)
    def wrapper(*args, **kwargs):
        if "username" not in session:
            return redirect(url_for("login"))
        return f(*args, **kwargs)
    return wrapper

@app.route("/login", methods=["GET", "POST"])
def login():
```

```
    if request.method == "POST":
        username = request.form["username"]
        password = request.form["password"]
        db = get_db()
        cur = db.cursor(dictionary=True)
        cur.execute("SELECT * FROM users WHERE username = %s",
(username,))
        user = cur.fetchone()
        db.close()
        if user and bcrypt.checkpw(password.encode(),
user["password_hash"].encode()):
            session["username"] = user["username"]
            session["role"] = user["role"]
            return redirect(url_for("index"))
        flash("Mauvais identifiants")
    return render_template("login.html")

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

def get_containers():
    current_user = session["username"].lower()
    is_admin = session["role"] == "admin"
    containers = []

    try:
        result = subprocess.check_output(
            "docker ps -a --format '{{.Names}}|{{.Ports}}|{{.Status}}'",
            shell=True, text=True
        )
        for line in result.strip().split("\n"):
            if not line.strip() or "6080" not in line: continue
            name, ports, status = line.split("|", 2)
            match = re.search(r"0\.0\.0\.0:(\d+)->6080/tcp", ports)
            if not match: continue
            port = int(match.group(1))
```

```
            # Extraction du propriétaire
            owner_match = re.search(r"-([a-z0-9_]+)-novnc$", name,
re.IGNORECASE)
            owner = owner_match.group(1).lower() if owner_match else None

            if is_admin or owner == current_user:
                # Détection OS : si commence par "kali-" → Kali, sinon
Ubuntu
                if name.lower().startswith("kali-"):
                    os_type = "kali"
                    display_name = name[5:]  # enlève "kali-"
                else:
                    os_type = "ubuntu"
                    display_name = name

                # Enlève le owner + -novnc
                if owner:
                    display_name = display_name.rsplit(f"-{owner}-novnc",
1)[0]

                display_name = display_name.replace("-", "
").strip().title()
                if not display_name or display_name.lower() == "machine":
                    display_name = os_type.capitalize()

                running = any(k in status for k in ["Up", "Restarting",
"Up "])

                containers.append({
                    "name": display_name,
                    "id": name,
                    "port": port,
                    "running": running,
                    "os": os_type  # ← pour le badge dans le HTML
                })
    except Exception as e:
        print("Erreur get_containers:", e)

    containers.sort(key=lambda x: x["port"])
```

```
    return containers

@app.route("/")
@require_login
def index():
    return render_template("index.html")

@app.route("/api/containers")
@require_login
def api():
    return jsonify(get_containers())

@app.route("/api/control/<cid>/<action>")
@require_login
def ctrl(cid, action):
    current_user = session["username"].lower()
    owner_match = re.search(r"-([a-z0-9_]+)-novnc$", cid, re.IGNORECASE)
    owner = owner_match.group(1).lower() if owner_match else None
    if session["role"] != "admin" and owner != current_user:
        return jsonify({"error": "Interdit"}), 403

    subprocess.run(f"docker rm -f {cid}", shell=True)
    return jsonify({"ok": True})

@app.route("/api/create", methods=["POST"])
@require_login
def create():
    username = session["username"].lower()
    data = request.json
    raw_name = data.get("name", "machine").strip()
    os_choice = data.get("os", "ubuntu")  # "ubuntu" ou "kali"

    # Même image, mais on marque le type dans le nom
    prefix = "kali" if os_choice == "kali" else "ubuntu"
    clean_name = re.sub(r'[^a-z0-9]', '', raw_name.lower().replace(" ",
"-")) or "machine"
    container_name = f"{prefix}-{clean_name}-{username}-novnc"

    # SUPPRESSION ANCIEN CONTENEUR
```

```
    subprocess.run(f"docker rm -f {container_name}", shell=True)

    # PORT LIBRE GARANTI
    used_ports = []
    try:
        result = subprocess.check_output("docker ps -a --format
'{{.Ports}}'", shell=True, text=True)
        for line in result.splitlines():
            m = re.search(r"0\.0\.0\.0:(\d+)->6080/tcp", line)
            if m:
                used_ports.append(int(m.group(1)))
    except: pass
    port = 6080
    while port in used_ports:
        port += 1

    # CRÉATION + START (toujours la même image)
    subprocess.run([
        "docker", "create",
        "--name", container_name,
        "-p", f"{port}:6080",
        "--shm-size=2g",
        "--restart", "unless-stopped",
        "ubuntu-novnc-clean:latest"
    ], check=True)

    subprocess.run(["docker", "start", container_name])
    time.sleep(4)
    return jsonify({"success": True, "port": port})

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

requirement.txt

```
Flask==3.0.3
```

index.html

```
<!DOCTYPE html>
<html lang="fr">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Docker Lab Ultimate</title>
<style>
 body { background: #0d0d1a; color: #0f9; font-family: 'Courier New',
monospace; margin:0; padding:20px; }
 h1 { text-align: center; font-size: 3em; text-shadow: 0 0 20px #0f9;
margin: 60px 0 30px 0; }
 .logout-btn { position: absolute; top: 20px; right: 30px;
background:#f66; color:white; padding:12px 22px; border:none;
border-radius:12px; font-weight:bold; cursor:pointer; box-shadow:0 0 20px
#f66; z-index:1000; }
 .logout-btn:hover { background:#ff4444; transform:scale(1.05); }

 .create-card { background: linear-gradient(135deg, #00ff9d, #00cc7a);
color: black; padding: 30px; border-radius: 20px; text-align: center;
font-size: 2em; font-weight: bold; cursor: pointer; max-width: 700px;
margin: 0 auto 40px; box-shadow: 0 0 50px rgba(0,255,157,0.6); }
 .create-card:hover { transform: scale(1.03); }

 .grid { display: grid; grid-template-columns: repeat(auto-fill,
minmax(460px, 1fr)); gap: 30px; padding-bottom:50px; }
 .card { background: #111122; border: 2px solid #0f9; border-radius: 20px;
overflow: hidden; box-shadow: 0 0 30px rgba(0,255,157,0.3);
position:relative; }
 .title { background: linear-gradient(90deg, #0f9, #0c6); color: black;
padding: 18px; text-align: center; font-size: 1.8em; font-weight: bold; }
 .os-badge { position: absolute; top: 10px; left: 10px; background: #000c;
padding: 6px 14px; border-radius: 10px; font-size: 0.9em; font-weight:
bold; }
 .status { text-align: center; padding: 12px; font-size: 1.4em;
font-weight: bold; }
 .running { color: #0f9; text-shadow: 0 0 15px #0f9; }
 .stopped { color: #f66; }
```

```
 .buttons { padding: 15px; text-align: center; background: #000; display:
flex; flex-wrap: wrap; gap: 10px; justify-content: center; }
 .btn { padding: 12px 20px; border: none; border-radius: 12px;
font-weight: bold; cursor: pointer; transition: all 0.2s; }
 .start { background: #0f9; color: black; }
 .stop { background: #f66; color: white; }
 .restart { background: #ff9500; color: white; }
 .delete { background: #c00; color: white; }
 .toggle-vnc { background: #00bfff; color: white; font-size: 1.3em;
padding: 16px 40px; width: 90%; }
 .btn:hover { transform: scale(1.1); box-shadow: 0 0 20px
rgba(255,255,255,0.4); }
 .vnc-wrapper { position: relative; background: black; }
 .vnc-container { height: 0; overflow: hidden; transition: height 0.6s
ease; }
 .vnc-container.open { height: 720px; }
 .vnc-container iframe { width: 100%; height: 100%; border: none; }
 .fullscreen-btn { position: absolute; top: 12px; right: 12px; z-index:
10; background: #0f9; color: black; padding: 12px 18px; border-radius:
12px; font-weight: bold; cursor: pointer; }
</style>
</head>
<body>

<button class="logout-btn"
onclick="location.href='/logout'">DÉCONNEXION</button>

<h1>DOCKER LAB ULTIMATE</h1>

<div class="create-card"
onclick="document.getElementById('modal').style.display='flex'">
 + CRÉER UNE NOUVELLE MACHINE
</div>

<div class="grid" id="grid"></div>

<!-- MODAL CHOIX OS -->
<div id="modal"
style="display:none;position:fixed;inset:0;background:rgba(0,0,0,0.95);z-i
ndex:9999;justify-content:center;align-items:center;">
```

```
 <div style="background:#111122;padding:40px;border-radius:24px;border:3px
solid #0f9;width:90%;max-width:600px;box-shadow:0 0 60px
#0f9;position:relative;">
   <span
style="position:absolute;top:10px;right:15px;font-size:3em;color:#f66;curs
or:pointer;"
onclick="this.parentElement.parentElement.style.display='none'">X</span>
   <h2 style="color:#0f9;text-align:center;margin-bottom:30px;">CHOISIR LE
SYSTÈME</h2>

   <div style="display:grid;grid-template-columns:1fr 1fr;gap:20px;">
     <div onclick="selectOS('ubuntu')"
style="background:#1e1e1e;border:3px solid
#0f9;padding:30px;border-radius:16px;cursor:pointer;text-align:center;">
       <h3 style="margin:0;color:#0f9;font-size:2em;">Ubuntu</h3>
       <p style="margin:10px 0 0;color:#0b8;">Légère • Stable</p>
     </div>
     <div onclick="selectOS('kali')" style="background:#1e1e1e;border:3px
solid
#f66;padding:30px;border-radius:16px;cursor:pointer;text-align:center;">
       <h3 style="margin:0;color:#f66;font-size:2em;">Kali Linux</h3>
       <p style="margin:10px 0 0;color:#f88;">Pentest • Full power</p>
     </div>
   </div>

   <div id="name-section" style="margin-top:30px;display:none;">
     <input id="name" placeholder="Nom de la machine (ex: MonKali)"
style="width:100%;padding:16px;margin:15px 0;border-radius:12px;border:2px
solid #0f9;background:#000;color:#0f9;font-size:1.2em;">
     <button onclick="create()"
style="width:100%;padding:20px;background:#0f9;color:black;border:none;bor
der-radius:16px;font-size:1.5em;font-weight:bold;cursor:pointer;">
       CRÉER LA MACHINE
     </button>
   </div>
   <p id="msg"
style="text-align:center;margin-top:20px;color:#0f9;font-size:1.3em;"></p>
 </div>
</div>
```

```
<script>
let selectedOS = "ubuntu";
const activeVNC = new Set();

function selectOS(os) {
 selectedOS = os;
 document.querySelectorAll('#modal [onclick^="selectOS"]').forEach(el => {
   el.parentElement.style.borderColor = "#444";
 });
 event.target.closest('div').style.borderColor = os === 'kali' ? "#f66" :
"#0f9";
 document.getElementById('name-section').style.display = 'block';
 document.getElementById('name').focus();
}

function buildCard(c) {
 const card = document.createElement("div");
 card.className = "card";
 card.id = "card-" + c.port;

 const badgeColor = c.os === "kali" ? "#f66" : "#0f9";
 const badgeText = c.os.toUpperCase();

 card.innerHTML = `
   <div class="title">${c.name} <span class="os-badge"
style="background:${badgeColor}40;color:${badgeColor};">${badgeText}</span
></div>
   <div class="status ${c.running ? 'running' : 'stopped'}">${c.running ?
'EN LIGNE' : 'ARRÊTÉE'}</div>
   <div class="buttons">
     <button class="btn start"
onclick="ctrl('${c.id}','start')">START</button>
     <button class="btn stop"
onclick="ctrl('${c.id}','stop')">STOP</button>
     <button class="btn restart"
onclick="ctrl('${c.id}','restart')">RESTART</button>
     <button class="btn delete" onclick="if(confirm('Supprimer cette
machine ?')) ctrl('${c.id}','remove')">SUPPRIMER</button>
   </div>
   <div style="padding:20px;text-align:center;">
```

```
      <button class="btn toggle-vnc" onclick="toggleVNC(this, ${c.port})">
        ${activeVNC.has(c.port) ? 'Masquer' : 'Afficher'} le bureau
      </button>
    </div>
    <div class="vnc-wrapper">
      <button class="fullscreen-btn" onclick="goFullscreen(this)">PLEIN
ÉCRAN</button>
      <div class="vnc-container ${activeVNC.has(c.port) ? 'open' : ''}"
id="vnc-${c.port}">
        <iframe loading="lazy"></iframe>
      </div>
    </div>
  `;
  return card;
}

function refresh() {
  fetch("/api/containers")
    .then(r => r.json())
    .then(data => {
      const grid = document.getElementById("grid");
      data.forEach(c => {
        if (!document.getElementById("card-" + c.port)) {
          grid.appendChild(buildCard(c));
        }
        // Mise à jour du statut
        const statusEl = document.querySelector(`#card-${c.port} .status`);
        if (statusEl) {
          statusEl.textContent = c.running ? "EN LIGNE" : "ARRÊTÉE";
          statusEl.className = "status " + (c.running ? "running" :
"stopped");
        }
      });
    });
}

function toggleVNC(btn, port) {
  const container = document.getElementById("vnc-" + port);
  const iframe = container.querySelector("iframe");
```

```
 if (container.classList.contains("open")) {
   container.classList.remove("open");
   iframe.src = "";
   btn.textContent = "Afficher le bureau";
   activeVNC.delete(port);
 } else {
   container.classList.add("open");
   iframe.src =
`http://${location.hostname}:${port}/vnc.html?autoconnect=1&resize=scale&q
uality=9`;
   btn.textContent = "Masquer le bureau";
   activeVNC.add(port);
 }
}

function goFullscreen(btn) {
 const wrapper = btn.closest(".vnc-wrapper");
 if (!document.fullscreenElement) {
   wrapper.requestFullscreen();
   btn.textContent = "QUITTER PLEIN ÉCRAN";
 } else {
   document.exitFullscreen();
   btn.textContent = "PLEIN ÉCRAN";
 }
}

function ctrl(id, action) {
 fetch(`/api/control/${id}/${action}`);
 setTimeout(refresh, 3000);
}

function create() {
 const name = document.getElementById("name").value.trim() || selectedOS;
 document.getElementById("msg").innerHTML = "Création en cours...";
 fetch("/api/create", {
   method: "POST",
   headers: {"Content-Type": "application/json"},
   body: JSON.stringify({name: name, os: selectedOS})
 })
 .then(r => r.json())
```

```
 .then(d => {
   if (d.success) {
     document.getElementById("msg").innerHTML = `Créée ! Port:
<b>${d.port}</b>`;
     setTimeout(() => location.reload(), 3000);
   } else {
     document.getElementById("msg").innerHTML = "Erreur :(";
   }
 });
}

// Lancement
refresh();
setInterval(refresh, 10000);
</script>
</body>
</html>
```

login.html

```
<!DOCTYPE html>
<html lang="fr">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Docker Lab Ultimate</title>
<style>
 body { background: #0d0d1a; color: #0f9; font-family: 'Courier New',
monospace; margin:0; padding:20px; }
 h1 { text-align: center; font-size: 3em; text-shadow: 0 0 20px #0f9;
margin: 60px 0 30px 0; }
 .logout-btn { position: absolute; top: 20px; right: 30px;
background:#f66; color:white; padding:12px 22px; border:none;
border-radius:12px; font-weight:bold; cursor:pointer; box-shadow:0 0 20px
#f66; z-index:1000; }
 .logout-btn:hover { background:#ff4444; transform:scale(1.05); }

 .create-card { background: linear-gradient(135deg, #00ff9d, #00cc7a);
color: black; padding: 30px; border-radius: 20px; text-align: center;
```

```
font-size: 2em; font-weight: bold; cursor: pointer; max-width: 700px;
margin: 0 auto 40px; box-shadow: 0 0 50px rgba(0,255,157,0.6); }
 .create-card:hover { transform: scale(1.03); }

 .grid { display: grid; grid-template-columns: repeat(auto-fill,
minmax(460px, 1fr)); gap: 30px; padding-bottom:50px; }
 .card { background: #111122; border: 2px solid #0f9; border-radius: 20px;
overflow: hidden; box-shadow: 0 0 30px rgba(0,255,157,0.3);
position:relative; }
 .title { background: linear-gradient(90deg, #0f9, #0c6); color: black;
padding: 18px; text-align: center; font-size: 1.8em; font-weight: bold; }
 .os-badge { position: absolute; top: 10px; left: 10px; background: #000c;
padding: 6px 14px; border-radius: 10px; font-size: 0.9em; font-weight:
bold; }
 .status { text-align: center; padding: 12px; font-size: 1.4em;
font-weight: bold; }
 .running { color: #0f9; text-shadow: 0 0 15px #0f9; }
 .stopped { color: #f66; }
 .buttons { padding: 15px; text-align: center; background: #000; display:
flex; flex-wrap: wrap; gap: 10px; justify-content: center; }
 .btn { padding: 12px 20px; border: none; border-radius: 12px;
font-weight: bold; cursor: pointer; transition: all 0.2s; }
 .start { background: #0f9; color: black; }
 .stop { background: #f66; color: white; }
 .restart { background: #ff9500; color: white; }
 .delete { background: #c00; color: white; }
 .toggle-vnc { background: #00bfff; color: white; font-size: 1.3em;
padding: 16px 40px; width: 90%; }
 .btn:hover { transform: scale(1.1); box-shadow: 0 0 20px
rgba(255,255,255,0.4); }
 .vnc-wrapper { position: relative; background: black; }
 .vnc-container { height: 0; overflow: hidden; transition: height 0.6s
ease; }
 .vnc-container.open { height: 720px; }
 .vnc-container iframe { width: 100%; height: 100%; border: none; }
 .fullscreen-btn { position: absolute; top: 12px; right: 12px; z-index:
10; background: #0f9; color: black; padding: 12px 18px; border-radius:
12px; font-weight: bold; cursor: pointer; }
</style>
</head>
```

```
<body>

<button class="logout-btn"
onclick="location.href='/logout'">DÉCONNEXION</button>

<h1>DOCKER LAB ULTIMATE</h1>

<div class="create-card"
onclick="document.getElementById('modal').style.display='flex'">
 + CRÉER UNE NOUVELLE MACHINE
</div>

<div class="grid" id="grid"></div>

<!-- MODAL CHOIX OS -->
<div id="modal"
style="display:none;position:fixed;inset:0;background:rgba(0,0,0,0.95);z-i
ndex:9999;justify-content:center;align-items:center;">
 <div style="background:#111122;padding:40px;border-radius:24px;border:3px
solid #0f9;width:90%;max-width:600px;box-shadow:0 0 60px
#0f9;position:relative;">
   <span
style="position:absolute;top:10px;right:15px;font-size:3em;color:#f66;curs
or:pointer;"
onclick="this.parentElement.parentElement.style.display='none'">X</span>
   <h2 style="color:#0f9;text-align:center;margin-bottom:30px;">CHOISIR LE
SYSTÈME</h2>

   <div style="display:grid;grid-template-columns:1fr 1fr;gap:20px;">
     <div onclick="selectOS('ubuntu')"
style="background:#1e1e1e;border:3px solid
#0f9;padding:30px;border-radius:16px;cursor:pointer;text-align:center;">
       <h3 style="margin:0;color:#0f9;font-size:2em;">Ubuntu</h3>
       <p style="margin:10px 0 0;color:#0b8;">Légère • Stable</p>
     </div>
     <div onclick="selectOS('kali')" style="background:#1e1e1e;border:3px
solid
#f66;padding:30px;border-radius:16px;cursor:pointer;text-align:center;">
       <h3 style="margin:0;color:#f66;font-size:2em;">Kali Linux</h3>
       <p style="margin:10px 0 0;color:#f88;">Pentest • Full power</p>
```

```
    </div>
   </div>

   <div id="name-section" style="margin-top:30px;display:none;">
    <input id="name" placeholder="Nom de la machine (ex: MonKali)"
style="width:100%;padding:16px;margin:15px 0;border-radius:12px;border:2px
solid #0f9;background:#000;color:#0f9;font-size:1.2em;">
    <button onclick="create()"
style="width:100%;padding:20px;background:#0f9;color:black;border:none;bor
der-radius:16px;font-size:1.5em;font-weight:bold;cursor:pointer;">
      CRÉER LA MACHINE
    </button>
   </div>
   <p id="msg"
style="text-align:center;margin-top:20px;color:#0f9;font-size:1.3em;"></p>
 </div>
</div>

<script>
let selectedOS = "ubuntu";
const activeVNC = new Set();

function selectOS(os) {
 selectedOS = os;
 document.querySelectorAll('#modal [onclick^="selectOS"]').forEach(el => {
   el.parentElement.style.borderColor = "#444";
 });
 event.target.closest('div').style.borderColor = os === 'kali' ? "#f66" :
"#0f9";
 document.getElementById('name-section').style.display = 'block';
 document.getElementById('name').focus();
}

function buildCard(c) {
 const card = document.createElement("div");
 card.className = "card";
 card.id = "card-" + c.port;

 const badgeColor = c.os === "kali" ? "#f66" : "#0f9";
 const badgeText = c.os.toUpperCase();
```

```
 card.innerHTML = `
   <div class="title">${c.name} <span class="os-badge"
style="background:${badgeColor}40;color:${badgeColor};">${badgeText}</span
></div>
   <div class="status ${c.running ? 'running' : 'stopped'}">${c.running ?
'EN LIGNE' : 'ARRÊTÉE'}</div>
   <div class="buttons">
     <button class="btn start"
onclick="ctrl('${c.id}','start')">START</button>
     <button class="btn stop"
onclick="ctrl('${c.id}','stop')">STOP</button>
     <button class="btn restart"
onclick="ctrl('${c.id}','restart')">RESTART</button>
     <button class="btn delete" onclick="if(confirm('Supprimer cette
machine ?')) ctrl('${c.id}','remove')">SUPPRIMER</button>
   </div>
   <div style="padding:20px;text-align:center;">
     <button class="btn toggle-vnc" onclick="toggleVNC(this, ${c.port})">
       ${activeVNC.has(c.port) ? 'Masquer' : 'Afficher'} le bureau
     </button>
   </div>
   <div class="vnc-wrapper">
     <button class="fullscreen-btn" onclick="goFullscreen(this)">PLEIN
ÉCRAN</button>
     <div class="vnc-container ${activeVNC.has(c.port) ? 'open' : ''}"
id="vnc-${c.port}">
       <iframe loading="lazy"></iframe>
     </div>
   </div>
 `;
 return card;
}

function refresh() {
 fetch("/api/containers")
   .then(r => r.json())
   .then(data => {
     const grid = document.getElementById("grid");
     data.forEach(c => {
```

```
      if (!document.getElementById("card-" + c.port)) {
        grid.appendChild(buildCard(c));
      }
      // Mise à jour du statut
      const statusEl = document.querySelector(`#card-${c.port} .status`);
      if (statusEl) {
        statusEl.textContent = c.running ? "EN LIGNE" : "ARRÊTÉE";
        statusEl.className = "status " + (c.running ? "running" :
"stopped");
      }
    });
  });
}

function toggleVNC(btn, port) {
 const container = document.getElementById("vnc-" + port);
 const iframe = container.querySelector("iframe");

 if (container.classList.contains("open")) {
   container.classList.remove("open");
   iframe.src = "";
   btn.textContent = "Afficher le bureau";
   activeVNC.delete(port);
 } else {
   container.classList.add("open");
   iframe.src =
`http://${location.hostname}:${port}/vnc.html?autoconnect=1&resize=scale&q
uality=9`;
   btn.textContent = "Masquer le bureau";
   activeVNC.add(port);
 }
}

function goFullscreen(btn) {
 const wrapper = btn.closest(".vnc-wrapper");
 if (!document.fullscreenElement) {
   wrapper.requestFullscreen();
   btn.textContent = "QUITTER PLEIN ÉCRAN";
 } else {
   document.exitFullscreen();
```

```
    btn.textContent = "PLEIN ÉCRAN";
  }
}

function ctrl(id, action) {
  fetch(`/api/control/${id}/${action}`);
  setTimeout(refresh, 3000);
}

function create() {
  const name = document.getElementById("name").value.trim() || selectedOS;
  document.getElementById("msg").innerHTML = "Création en cours...";
  fetch("/api/create", {
    method: "POST",
    headers: {"Content-Type": "application/json"},
    body: JSON.stringify({name: name, os: selectedOS})
  })
  .then(r => r.json())
  .then(d => {
    if (d.success) {
      document.getElementById("msg").innerHTML = `Créée ! Port:
<b>${d.port}</b>`;
      setTimeout(() => location.reload(), 3000);
    } else {
      document.getElementById("msg").innerHTML = "Erreur :(";
    }
  });
}

// Lancement
refresh();
setInterval(refresh, 10000);
</script>
</body>
</html>
```

Dockerfile:

Dockerfile – Ubuntu 24.04 + XFCE + noVNC – marche même sans internet parfait
FROM ubuntu:24.04

```
# Variables pour éviter les interactions
ENV DEBIAN_FRONTEND=noninteractive \
    TZ=Europe/Paris

# Miroir alternatif + installation des paquets (sans firefox qui bloque)
RUN sed -i 's/archive.ubuntu.com/fr.archive.ubuntu.com/g' /etc/apt/sources.list && \
    sed -i 's/security.ubuntu.com/fr.archive.ubuntu.com/g' /etc/apt/sources.list && \
    apt update -qq && \
    apt install -y --no-install-recommends \
        xfce4 xfce4-goodies \
        x11vnc xvfb \
        novnc websockify \
        net-tools \
        supervisor \
        dbus-x11 \
        mousepad thunar \
        nano wget curl && \
    apt clean && \
    && rm -rf /var/lib/apt/lists/*

# Répertoire VNC
RUN mkdir -p /root/.vnc

# Copie du script de démarrage propre
COPY entrypoint.sh /entrypoint.sh
RUN chmod +x /entrypoint.sh

EXPOSE 6080

STOPSIGNAL SIGTERM

ENTRYPOINT ["/entrypoint.sh"]
```

entrypoint.sh :

```
#!/bin/bash
set -e

echo "Démarrage propre Ubuntu + noVNC..."

# Nettoyage propre à l'arrêt
cleanup() {
    echo "Arrêt propre en cours..."
```

```
    pkill -f x11vnc 2>/dev/null || true
    pkill -f novnc_proxy 2>/dev/null || true
    pkill -f Xvfb 2>/dev/null || true
    exit 0
}
trap cleanup SIGTERM SIGINT

# Démarrage des services
Xvfb :1 -screen 0 1280x1024x24 +extension GLX > /dev/null 2>&1 &
export DISPLAY=:1
xfce4-session > /dev/null 2>&1 &

sleep 8

x11vnc -display :1 -forever -nopw -listen 0.0.0.0 -xkb > /dev/null 2>&1 &

echo "Attente x11vnc sur 5900..."
timeout=60
while ! netstat -tln 2>/dev/null | grep -q ":5900 " && [ $timeout -gt 0 ]; do
    sleep 1
    ((timeout--))
done

if [ $timeout -eq 0 ]; then
    echo "ERREUR : x11vnc n'a pas démarré"
    exit 1
fi

echo "Lancement noVNC sur le port 6080..."
exec /usr/share/novnc/utils/novnc_proxy \
    --vnc localhost:5900 \
    --listen 0.0.0.0:6080 \
    --web /usr/share/novnc
```