Rapport de Solution
SAE - Cloud

La vie privée n'est pas un luxe, c'est une infrastructure.

Bossou Hugo
Salhi Morgan
Boschian Mathis

Sommaire

Introduction

Le projet **InfraGuard** a pour objectif de concevoir et déployer une infrastructure informatique sécurisée, automatisée et **libre de tout coût de licence**, spécifiquement dimensionnée pour les **TPE et PME**. Dans un contexte de dépendance numérique accrue et de budgets restreints, il est devenu critique pour ces structures de disposer d’un système fiable sans subir l'enfermement propriétaire (lock-in).

Parties prenantes

Le projet est porté par **Jacky'sDev**, une startup agile composée de trois collaborateurs aux rôles complémentaires :

- **Mathis Boschian** : Product Owner, garant de la vision produit et des besoins clients.
- **Morgan Salhi** : Ingénieur DevOps, chargé de l'automatisation et de la robustesse technique.
- **Hugo Bossou** : Architecte Solutions, concepteur de la structure globale et de la sécurité.

Description technique

La solution repose exclusivement sur des technologies Open Source modélisées sur mesure. L’infrastructure comprend des serveurs applicatifs, une base de données redondante, un système de load balancing et un accès sécurisé via VPN. L'usage du modèle DevOps permet une gestion fluide et une répétabilité du déploiement.

Bénéfices attendus et Valeur Ajoutée

Le projet InfraGuard se distingue par trois piliers majeurs :

1. **Réduction de l'exposition (Anti-OSINT) :** Partant du constat que les données publiques des entreprises et de leurs collaborateurs sont massivement exploitées pour du phishing ou des piratages, InfraGuard intègre des composants configurés selon les meilleures pratiques de sécurité. L'objectif est de réduire drastiquement la surface d'attaque "source ouverte" de la structure en sécurisant nativement les flux et les accès.
2. **Souveraineté et contrôle total :** En éliminant les coûts de licences grâce à l'utilisation exclusive de technologies Open Source modélisées sur mesure, nous offrons une solution économiquement imbattable. Ce choix garantit aux clients un accès total et un pouvoir absolu sur leur propre infrastructure et leurs données, sans aucune dépendance vis-à-vis d'un éditeur tiers ou d'un fournisseur de Cloud.
3. **Résilience et Haute Disponibilité :** L’architecture multi-tiers redondante, composée de load balancers et de bases de données en mode maître-esclave, assure une continuité de service de niveau industriel. Cette configuration permet de surmonter des pannes matérielles majeures : en cas de défaillance du serveur principal, le système permet une reprise immédiate de l'activité sans perte de données.

Contexte et portée du projet

Aujourd'hui, les TPE et PME font face à un paradoxe : elles sont les premières cibles des cyberattaques (phishing, vol de données via OSINT), mais disposent rarement du budget ou de l'expertise pour gérer des infrastructures complexes. Elles se retrouvent souvent prisonnières de solutions propriétaires coûteuses et opaques.

La problématique de **Jacky'sDev** est donc la suivante : **Comment offrir une infrastructure de niveau industriel, souveraine et hautement disponible, avec un coût d'entrée minimal et une gestion automatisée ?**.

2. Périmètre du projet

Le projet **InfraGuard** englobe la conception et le déploiement automatisé d'une "bulle" informatique complète sur la plateforme de virtualisation **Proxmox VE**. Le périmètre inclut :

- **L'architecture réseau :** Mise en place d'un tunnel VPN chiffré (passerelle unique) et segmentation des flux (backend/frontend).
- **La haute disponibilité :** Déploiement de deux Load Balancers et d'un cluster de bases de données répliquées (Maître/Esclave).
- **Le socle applicatif :** Installation des outils de productivité sécurisés (**Vaultwarden**, **Authentik**, **Gitea**, **SimpleLogin**).
- **L'automatisation :** Livraison du code de déploiement via **Ansible**, **OpenTofu** et **Packer**.

3. Limites et Exclusions

Pour garantir la rentabilité et la clarté de l'offre, certaines prestations sont exclues du périmètre initial :

- **Maintenance matérielle :** Le projet se concentre sur la couche logicielle et système ; la gestion physique des serveurs (hardware) reste à la charge du client ou de l'hébergeur (sauf abonnement hébergement voir **Budget et ressources)**.
- **Développement applicatif :** Nous fournissons la plateforme et les outils, mais nous n'intervenons pas dans le développement du code métier propre à l'entreprise cliente.
- **Gestion des terminaux finaux :** La sécurisation des postes de travail individuels (laptops, smartphones des employés) est hors périmètre ; le projet s'arrête à la passerelle VPN.

L'Intégration du DevOps dans le projet InfraGuard

1. Définition

Le **DevOps** est une approche conceptuelle et technique visant à unifier le développement logiciel (**Dev**) et l'administration des infrastructures (**Ops**) pour accélérer et sécuriser le cycle de vie des applications. En s'appuyant sur l'automatisation de la chaîne **CI/CD** et l'infrastructure as code (**IaC**), il fluidifie et facilite la transition entre le développement et la génération du livrable final. Son rôle est de mettre en place un environnement où les tests automatisés, la surveillance continue et les mesures de sécurité garantissent des versions reproductibles et sans erreur. En industrialisant ces processus, le DevOps s'assure du bon fonctionnement et de la conformité aux normes de qualité, permettant ainsi une mise à disposition fluide, fiable et performante de la solution logicielle, quel que soit son mode de déploiement.

2. Le DevOps comme pilier central d'InfraGuard

Dans le cadre de notre projet **InfraGuard**, le DevOps n'est pas un simple ajout, c'est le cœur de notre démarche. Notre objectif est de fournir une infrastructure sécurisée capable de restreindre l'empreinte numérique des utilisateurs pour prévenir les fuites de données et les cyberattaques.

Pour y parvenir, nous avons adopté une approche **"Infrastructure as Code" (IaC)** et une industrialisation complète du déploiement :

- **Provisioning et Configuration :** L'utilisation de **Packer** (création d'images), **OpenTofu** (déploiement d'infrastructure) et **Ansible** (configuration automatisée) garantit que chaque brique de notre système est reproductible, versionnée et sécurisée dès sa création.
- **Conteneurisation et Services :** Le recours à **Docker** permet d'isoler les services, tandis que le déploiement d'outils comme **Gitea** (gestion du code) et **Authentik** (gestion des identités) structure un environnement collaboratif et sécurisé, pilier de toute chaîne de livraison moderne.

3. Les Avantages pour le projet

- **Industrialisation et Fiabilité :** Le DevOps nous offre une vision globale sur le cycle de vie du livrable. En construisant la solution "brique par brique" de manière automatisée, nous créons un système durable où chaque composant est testé et validé avant sa mise en service.
- **Sécurité Native (DevSecOps) :** L'automatisation permet d'intégrer les normes de sécurité directement dans le processus de création. Cela réduit drastiquement les risques d'erreur humaine lors du déploiement, assurant ainsi une protection constante des données de l'entreprise.

- **Agilité du livrable :** En automatisant le passage du développement à la mise à disposition, nous pouvons faire évoluer l'infrastructure rapidement pour répondre aux nouvelles menaces sans casser l'existant.
- **La Chaîne de Provisionnement Automatisée :** L'infrastructure est auto-documentée par le code. En cas de sinistre total, nous pouvons remonter l'intégralité du SI de la TPE en moins de 10 minutes à partir d'un simple dépôt Git.

4. Les Inconvénients et Points de Vigilance

- **Complexité et Courbe d'Apprentissage :** La mise en place d'une telle chaîne d'outils (OpenTofu, Ansible, Packer, Docker…) demande une expertise technique pointue et un temps de configuration initial important qu' il faut donc mesurer et vérifier à quel point c’est bénéfique pour chaque projet.

- **Le risque de Sur-Optimisation :** L'un des pièges majeurs est de perdre de vue la mission principale au profit d'une optimisation technique excessive. Le DevOps peut encourager à automatiser des détails mineurs au détriment de la valeur métier. Notre défi reste donc de rester concentrés sur l'essentiel : la sécurité et l'efficacité du produit final, sans ajouter de complexité inutile.

- **Dépendance à la Toolchain :** Utiliser autant d'outils impose une maintenance constante de la chaîne de production elle-même (mises à jour de sécurité de Docker, Ansible, etc.) pour éviter que nos outils ne deviennent une vulnérabilité.

Architecture

La solution est déployée sur une station de travail dédiée, offrant un équilibre optimal entre performance et coût pour une TPE/PME.

Hardware

- **Référence :** Serveur Custom Jacky'sDev.
  - **CPU :** AMD Ryzen 5
  - **RAM :** 64Go.
  - **Stockage :** 2 * 500 Go de SSD en miroir (RAID 1) pour la résilience matérielle.

- Tableau matériel des différentes machines de l’architectures

Rôle de la machine	OS (Template)	CPU (Cœurs)	RAM (Go)	Disque (Go)	Instance(s)
VPN Gateway	AlmaLinux 10	3	2 Go	30 Go	1
Load Balancers	Ubuntu	2	2 Go	30 Go	2
Backend Servers	Ubuntu	4	16 Go	30 Go	2
Bases de données	AlmaLinux 10	2	4 Go	30 Go	2

Operating System

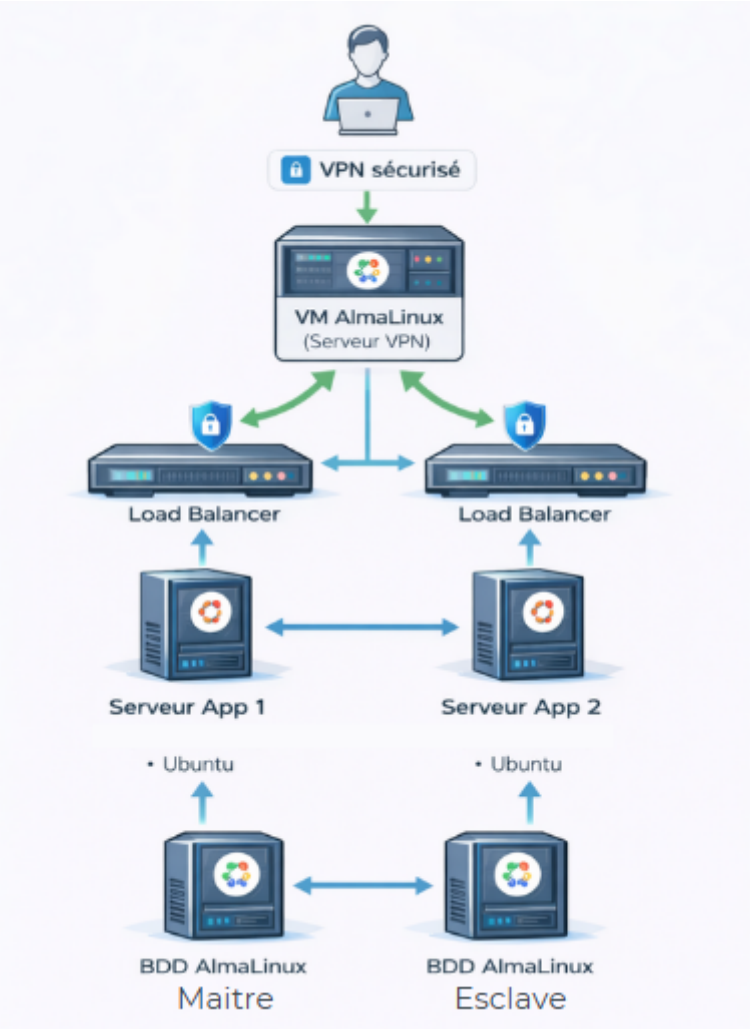
Pour héberger notre solution, nous utilisons **Proxmox VE 9.1.2**, une plateforme de virtualisation open source basée sur Linux. Elle permet de créer et gérer facilement plusieurs machines virtuelles sur un même serveur. Nous avons choisi Proxmox car il est gratuit (Open Source sans coût de licence), simple à administrer grâce à son interface web et bien adapté à la gestion d’infrastructures virtualisées.

Au seins meme de notre infrastructure, nous utilisons principalement deux systèmes d’exploitation :

- **AlmaLinux 10 :** Choisi pour le VPN et les bases de données pour sa stabilité "Enterprise-grade" et sa sécurité renforcée

- **Ubuntu Server 24.04.4 LTS :** Utilisé pour la couche applicative en raison de sa flexibilité et de son support communautaire étendu.

D’un point de vue architectural, notre produit repose sur une infrastructure composée de sept machines virtuelles, chacune ayant un rôle spécifique dans le fonctionnement global du système, comme illustré dans le schéma ci-dessous.



Architecture Système et Réseau

L'architecture logicielle et réseau d'InfraGuard s'appuie sur une structure multi-tiers rigoureuse, pensée pour garantir une segmentation optimale des flux et une protection sans faille de l'intégrité des données. L'ensemble de l'infrastructure est composé de sept machines virtuelles orchestrées pour répondre aux exigences de sécurité et de disponibilité des TPE/PME.

1. Couche d'Accès et Distribution

L'accès à l'intégralité du système est exclusivement centralisé par une **passerelle VPN sécurisée** sous AlmaLinux, qui régule strictement l'administration de l'infrastructure et fait office de point d'entrée unique. Une fois l'authentification validée, les requêtes sont prises en charge par deux **load balancers Nginx**. Leur rôle est de répartir intelligemment la charge vers le pôle applicatif, éliminant ainsi tout point de défaillance unique (SPOF).

2. Couche Applicative et Conteneurisation

Le cœur du système repose sur deux serveurs **Ubuntu Server** redondants. Ces instances servent d'hôtes pour l'orchestration des services via **Docker**, permettant d'isoler chaque brique applicative de l'infrastructure (Authentik, Vaultwarden, SimpleLogin, Gitea). Cette approche par conteneurs garantit une gestion agile des services et une continuité de service transparente : en cas de défaillance d'un nœud, la redondance assure une reprise immédiate sans interruption pour l'utilisateur final.

3. Couche de Données et Sécurité Profonde

La protection des actifs critiques se manifeste par une étanchéité stricte entre les couches applicatives et le stockage. Le segment de données est **totalement isolé** du réseau public et dépourvu d'adresses IP publiques. Les serveurs de base de données AlmaLinux y sont configurés en **architecture maître-esclave**, assurant une réplication des données en temps réel.

La sécurité est renforcée par un **filtrage IP restrictif** : les serveurs de base de données n'acceptent de flux que de la part des instances applicatives autorisées. Cette segmentation rigoureuse garantit que les informations sensibles de l'entreprise restent hermétiques aux menaces externes tout en restant parfaitement disponibles et fiables pour l'organisation.

Conception

Industrialisation et déploiement

Pour garantir une répétabilité parfaite et un Plan de Reprise d'Activité (PRA) inférieur à 10 minutes (sur la partie monté de l’infrastructure sans données clientes), nous avons banni toute configuration manuelle au profit de l'Infrastructure as Code (IaC):

- **Packer :** Utilisé pour la création d'images "Golden Images" pré-configurées. Cela nous permet de déployer des VM AlmaLinux et Ubuntu avec un socle de sécurité déjà durci.

- **OpenTofu (Fork Open Source de Terraform) :** Cet outil assure le provisionnement automatique des ressources sur l'hyperviseur Proxmox VE. Il définit les ressources CPU, RAM et réseaux pour les 7 machines virtuelles de l'architecture.(cf annexe)

- **Ansible :** Une fois les machines démarrées, Ansible prend le relais pour la configuration logicielle et le déploiement des services. C'est le cerveau de notre automatisation qui installe et configure Docker, les tunnels VPN et les bases de données répliquées.(cf github)

Supervision de l’Infra Système et Réseaux

La supervision constitue les "yeux" de l'approche DevOps mise en œuvre dans le projet InfraGuard. Elle permet de garantir la conformité aux normes de qualité, d'assurer une surveillance continue et de valider la disponibilité réelle des ressources. Notre stratégie s'articule autour de trois axes de contrôle :

A. Vérification de la Connectivité et de l'État du Parc (Ansible)

L'outil **Ansible** est utilisé pour assurer un contrôle rapide et global de l'accessibilité de l'ensemble des sept machines virtuelles.

- **Module Ping :** Nous exécutons régulièrement la commande ansible all -m ping pour confirmer que chaque nœud (VPN, Load Balancers, Backends, Bases de données) est en ligne et joignable.
- **Inventaire Dynamique :** Cette supervision s'appuie sur un fichier d'inventaire (inventory.yml) qui segmente les machines par rôle, permettant une surveillance ciblée par couche technique (ex: tester uniquement la couche database).

B. Monitoring de l'Hyperviseur (Proxmox VE)

La plateforme **Proxmox VE 9.1.2** offre une première couche de supervision matérielle et système indispensable.

- **Interface d'Administration :** Elle permet de surveiller en temps réel la consommation des ressources critiques (CPU, RAM, Stockage) allouées à chaque instance.
- **Gestion des Nœuds :** L'interface facilite la détection immédiate d'une panne au niveau de l'hôte Ryzen 5 ou d'une saturation des disques SSD configurés en miroir.

C. Intégrité des Flux Réseaux et des Tunnels (VPN & Load Balancers)

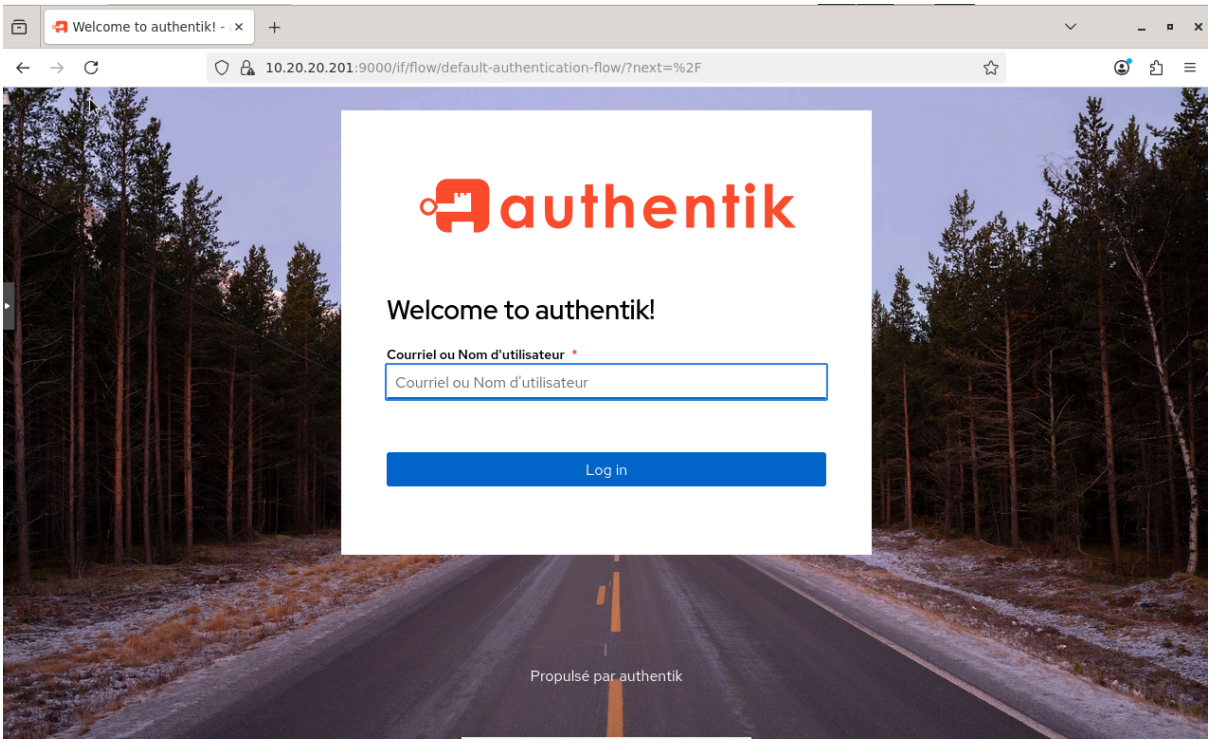
La supervision réseau se concentre sur la santé des points d'entrée et de la segmentation.

- **Statut du Tunnel VPN :** Le serveur sous AlmaLinux fait l'objet d'une surveillance constante pour garantir que le tunnel chiffré est opérationnel, condition *sine qua non* pour l'accès sécurisé des utilisateurs et des administrateurs.
- **Health Checks des Load Balancers :** Les deux instances Nginx agissent comme des sondes de disponibilité pour le pôle applicatif Ubuntu. Si un serveur Backend ne répond plus, le Load Balancer le détecte et redirige le flux vers le nœud sain, assurant ainsi la continuité de service.

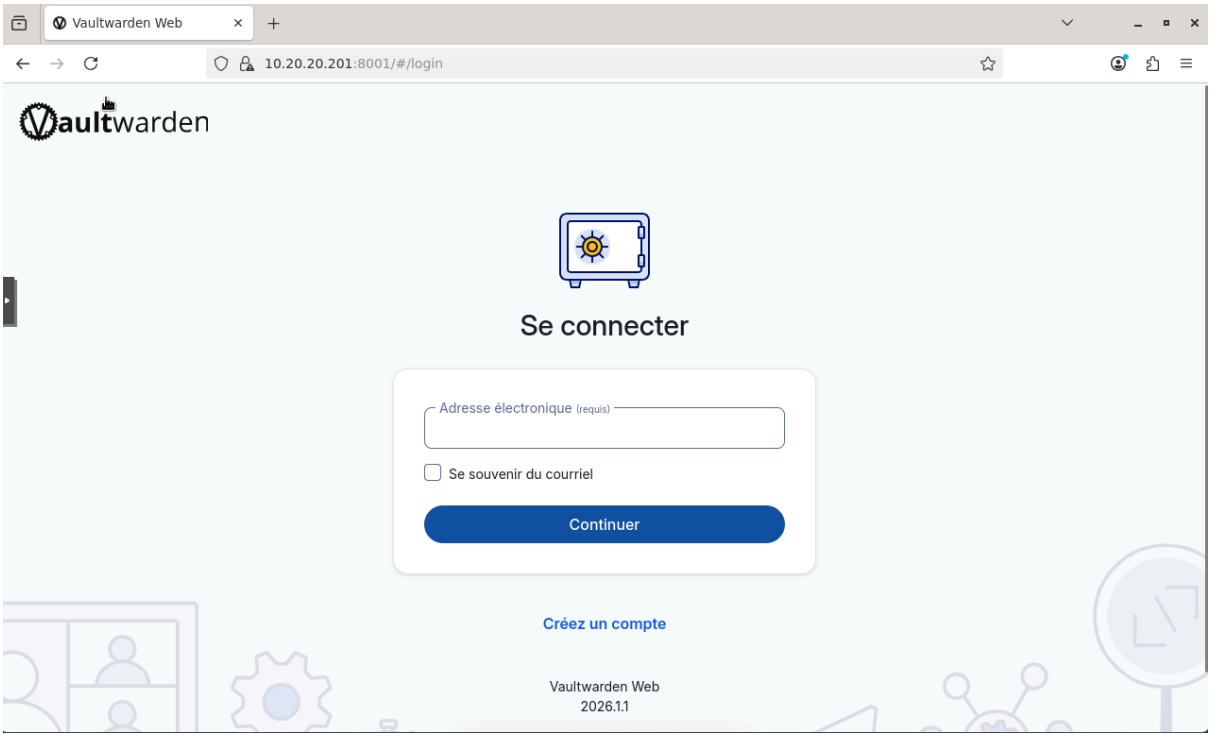
Supervision Applicative

Nous validons la mise en œuvre opérationnelle des différentes applications en vérifiant leur accessibilité effective depuis une station de test connectée au tunnel VPN sécurisé.

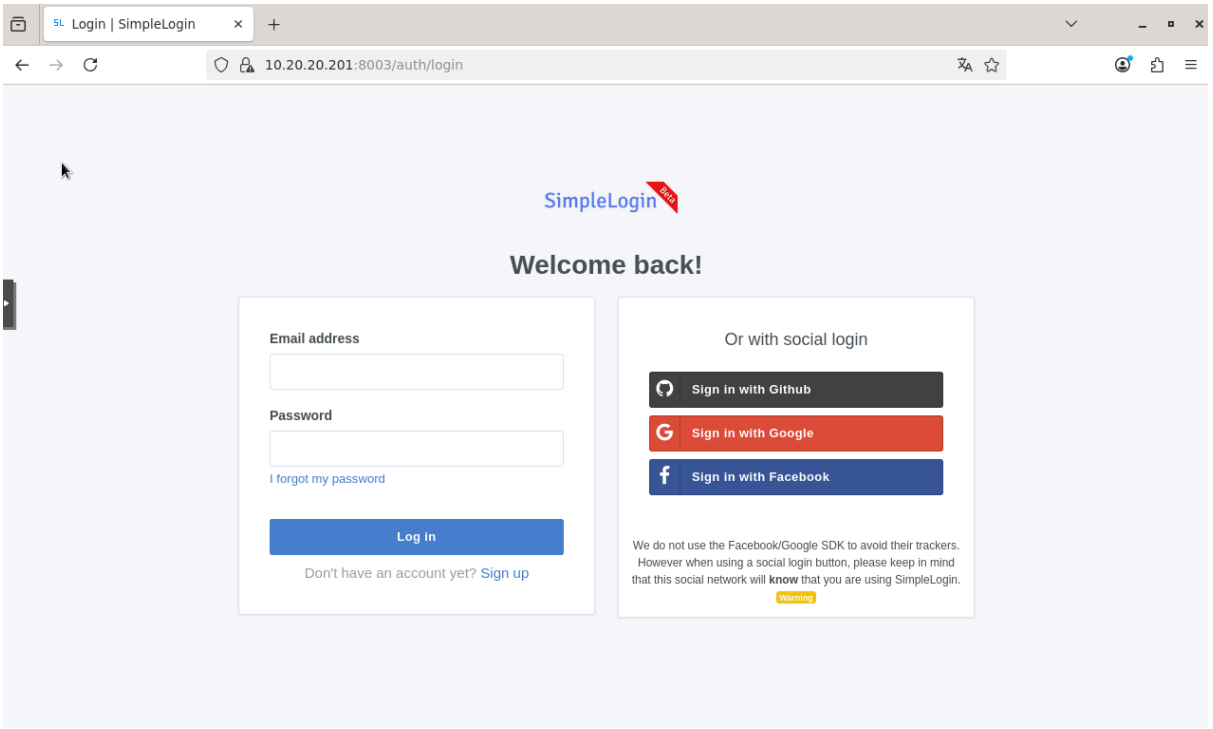
- **Authentik :** Plateforme d'Identity Provider permettant de centraliser l'authentification (SSO) et d'appliquer des politiques de contrôle d'accès granulaires pour sécuriser l'entrée sur chaque service de l'infrastructure.



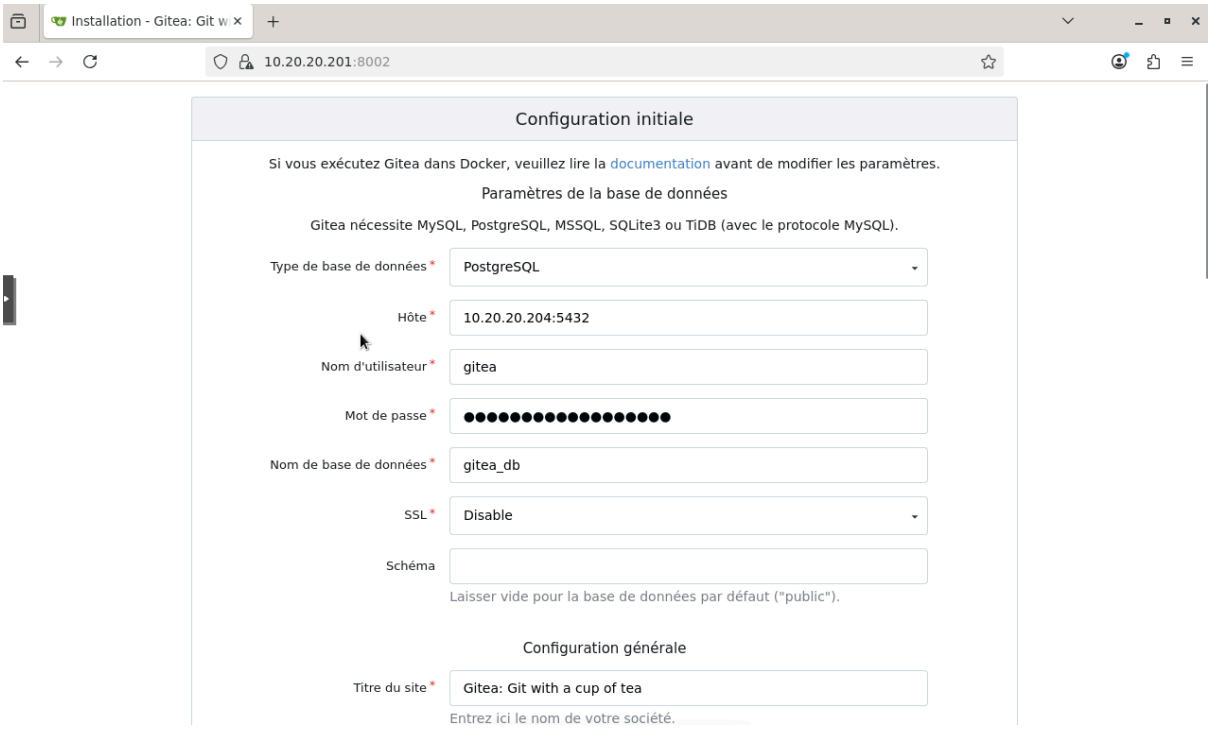
- **VaultWarden :** Alternative légère et auto-hébergée à Bitwarden, offrant un coffre-fort numérique chiffré pour la gestion centralisée et souveraine des mots de passe et des secrets au sein de l'organisation.



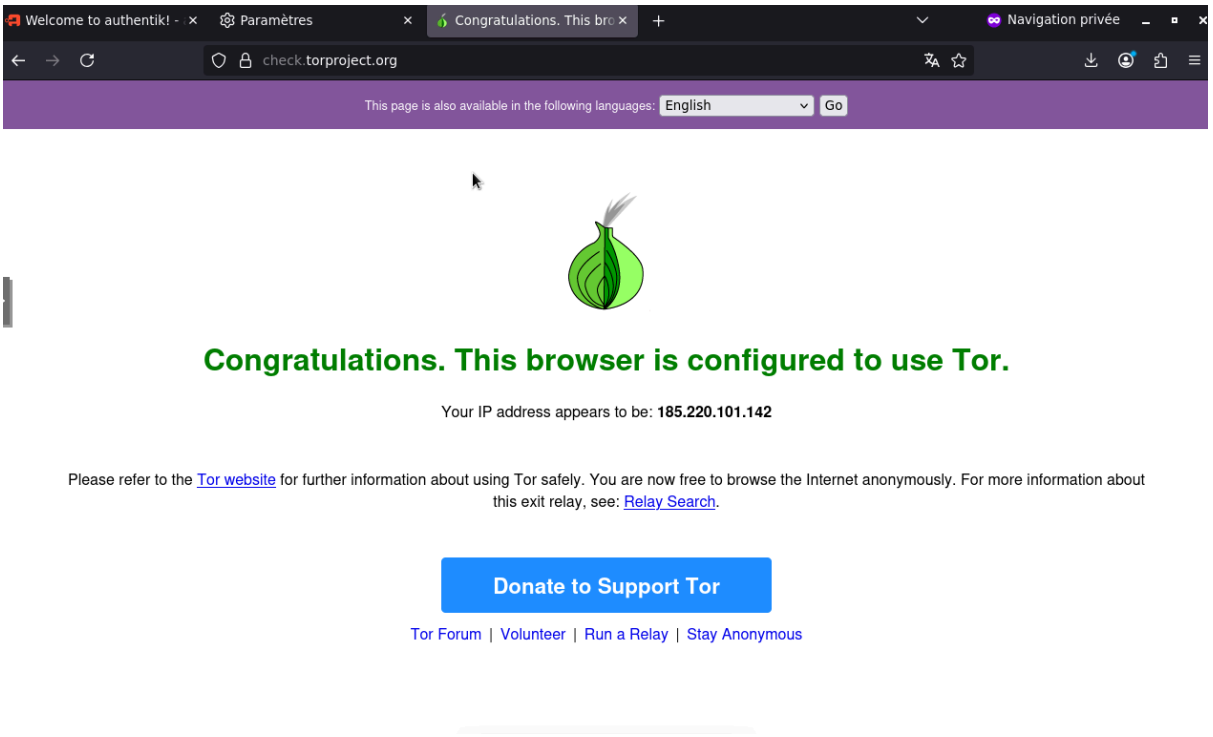
- **SimpleLogin :** Solution de protection de la vie privée permettant de générer des alias e-mail pour les utilisateurs, masquant ainsi leur adresse réelle afin de prévenir le spam et les tentatives de phishing.



- **Gitea :** Forge logicielle légère basée sur Git, facilitant la gestion du code source et la collaboration entre développeurs tout en garantissant que les données restent stockées sur nos propres serveurs.



- **Tor :** Réseau de communication décentralisé utilisé pour renforcer l'anonymat des accès et protéger l'identité numérique de l'infrastructure en masquant son adresse IP publique lors des échanges sortants.



En somme, cette phase de conception témoigne de l’engagement de **Jacky'sDev** à fournir une infrastructure robuste, transparente et souveraine. L'alliance stratégique de l'automatisation via l'Infrastructure Code (Packer, Open Tofu, Ansible) et d'un monitoring

multi couche garantit non seulement un déploiement éclair en moins de 10 minutes, mais aussi une résilience critique face aux pannes matérielles ou logicielles.

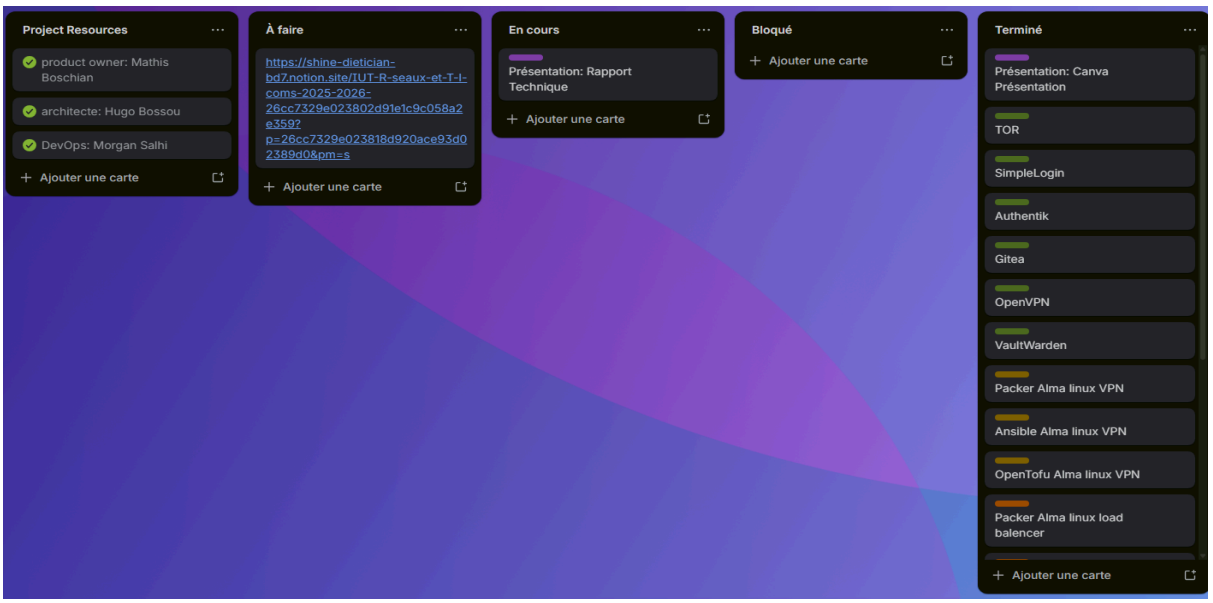
Ce socle technologique, intégralement basé sur des solutions **Open Source**, permet aux TPE/PME de s'affranchir des coûts de licence tout en bénéficiant de services de sécurité de niveau industriel (SSO, coffre-fort de mots de passe, gestion d'alias e-mail). En plaçant la confidentialité et le contrôle des données au centre de l'architecture, InfraGuard transforme la gestion informatique de contrainte budgétaire en un véritable atout de protection numérique pour l'entreprise et ses collaborateurs.

Plan de développement et mise en oeuvre

Le développement du projet a été organisé en utilisant une méthode de gestion de projet de type Kanban. Cette méthode permet de suivre l’avancement des tâches de manière visuelle et d’organiser efficacement le travail de l’équipe.

Un tableau de suivi des tâches a été mis en place sur Trello, fonctionnant sur le principe d’un tableau Kanban. Celui-ci est divisé en plusieurs colonnes représentant l’état d’avancement des tâches :

- **À faire** : tâches planifiées mais non commencées
- **En cours** : tâches actuellement en développement
- **Bloqué** : tâches rencontrant un problème ou nécessitant une résolution
- **Terminé** : tâches finalisées et validées



Chaque tâche correspond à une fonctionnalité ou un composant de l’infrastructure, comme par exemple la mise en place du VPN OpenVPN, du gestionnaire de mots de passe VaultWarden, de l’authentification Authentik, ou encore la configuration des load balancers Nginx.

Avant le démarrage du projet, nous avons également réalisé une estimation du temps de travail nécessaire pour chaque tâche. Cela nous a permis d’évaluer le nombre d’heures de travail effectif nécessaires à la réalisation du projet, tout en prévoyant de petites marges afin de gérer d’éventuels imprévus ou difficultés techniques.

Déploiement de l'infrastructure

Le déploiement de l'infrastructure est réalisé sur la plateforme de virtualisation Proxmox VE. Plusieurs outils d'automatisation sont utilisés afin de faciliter la création et la configuration des machines.

- **Packer** : utilisé pour créer des images de machines virtuelles prêtes à être déployées
- **OpenTofu** : utilisé pour automatiser le déploiement de l'infrastructure
- **Ansible** : utilisé pour configurer automatiquement les services sur les différentes machines

Ces outils permettent de mettre en place une approche Infrastructure as Code, ce qui rend l'infrastructure plus reproductible, maintenable et automatisée.

Au début du projet, nous avons estimé le temps nécessaire à sa réalisation afin d'organiser le travail de l'équipe. Nous avons prévu une durée d'environ une semaine et demie de travail.

Durant les cinq premiers jours, nous avons travaillé environ 2 heures par jour afin de mettre en place les bases du projet et commencer la conception de l'infrastructure. Par la suite, deux journées complètes de 8 heures ont été consacrées au déploiement et à la configuration des différents services.

Enfin, les trois derniers jours ont permis de finaliser certains éléments du projet, pour un total d'environ 5 heures supplémentaires. Et 3 heures ou nous ne somme pas concentré

Au total, ce projet représente environ 28 heures de travail effectif.

Partie	Temps de mise en place
Packer	5/31h
OpenToFu	5/31h
Load Balancer	2/31h
OpenVPN	4/31h
VaultWarden	2/31h
Authentik	1/31h
SimpleLogin	2/31h
TOR	3/31h
Test	4/31h

Budget et ressources

La solution InfraGuard a été pensée pour offrir un niveau de service industriel à un coût maîtrisé, typiquement adapté aux contraintes budgétaires d'une TPE/PME. Notre stratégie repose sur le transfert des coûts de licence vers l'investissement dans l'automatisation.

Ressources Logicielles : La Souveraineté par l'Open Source L'intégralité de notre stack logicielle (Proxmox, OpenTofu, Ansible, Docker et les services applicatifs) repose sur des technologies Open Source. Ce choix stratégique permet de réduire à zéro les coûts de licence récurrents, évitant ainsi la dépendance vis-à-vis des grands fournisseurs de Cloud (Vendor Lock-in). L'investissement financier est ainsi réorienté vers l'expertise technique et la personnalisation de l'infrastructure.

Ressources Matérielles : Optimisation et Résilience Pour garantir la haute disponibilité, l'infrastructure est déployée sur deux serveurs physiques redondants. Chaque nœud est équipé de processeurs multi-cœurs et de 16 Go de mémoire vive, permettant de supporter la charge des 6 machines virtuelles tout en conservant une marge de progression pour l'évolution des services. L'utilisation de disques SSD garantit les performances nécessaires aux entrées/sorties des bases de données répliquées.

Ressources Humaines et Maintenance Le coût principal du projet réside dans la phase de conception et d'automatisation initiale menée par l'équipe Jacky's Dev (3 collaborateurs). Grâce à l'utilisation de l'Infrastructure as Code (IaC), les coûts de maintenance opérationnelle sont drastiquement réduits : les mises à jour, la surveillance et les procédures de récupération (DRP) sont automatisées, minimisant ainsi le temps d'intervention humaine et le risque d'erreur.

Pour s'adapter aux besoins et aux capacités financières de chaque structure, deux modèles d'acquisition sont proposés pour la solution InfraGuard :

- **Forfait Mensuel (60 € / utilisateur)** : Cette formule "tout inclus" comprend l'accès complet à la plateforme, l'hébergement des données sur nos infrastructures sécurisées, ainsi que l'intégralité du support technique et de la maintenance.
- **Offre Unitaire (600 € / utilisateur)** : Ce modèle repose sur un paiement unique donnant accès à la solution logicielle. Il exclut toutefois les services d'hébergement, de maintenance et de support.

Afin de garantir une exploitation sereine de l'infrastructure, nous avons mis en place un accompagnement client rigoureux :

- **Disponibilité du Support** : Une assistance technique est assurée 5 jours sur 7, sur une plage horaire allant de 8h à 18h.
- **Support Bilingue** : Le service client est disponible en français et en anglais pour répondre aux besoins de communication des équipes.
- **Services et Personnalisation** : Des prestations à la carte sont également disponibles, comme une journée de formation pour la prise en main de la solution ou l'établissement de devis spécifiques pour l'ajout de fonctionnalités et de plugins personnalisés.

	Mensuel	Unitaire
Accès à la solution	✓	✓
Support et Maintenance	✓	✗
Hebergement	✓	✗
Prix	60 € / utilisateur	600 € / utilisateur

Conclusion

Le projet InfraGuard démontre qu'il est possible de concilier haute disponibilité, sécurité et budget maîtrisé pour les petites structures. En combinant la puissance de la virtualisation sous Proxmox et la flexibilité de l'automatisation DevOps, nous avons conçu une infrastructure capable de survivre à des pannes majeures sans perte de données. Au-delà de la réponse technique, cette solution offre une véritable souveraineté numérique à l'entreprise cliente : les données sont hébergées localement, chiffrées et protégées par une architecture multicouche.

D’un point de vue pédagogique, la réalisation de ce projet a été une expérience extrêmement formatrice pour notre équipe. Elle nous a permis de confronter les concepts théoriques de la virtualisation et du DevOps à la réalité technique du terrain. Nous avons énormément appris sur l’interopérabilité des outils d’automatisation et sur la rigueur nécessaire à la gestion d’un cluster de haute disponibilité. Nous sommes particulièrement satisfaits du résultat final : une infrastructure robuste, cohérente et entièrement automatisée. Ce projet nous a non seulement permis de monter en compétences techniques, mais il a aussi renforcé notre capacité à travailler en équipe pour livrer une solution industrielle dont nous sommes fiers.

Annexe

Lien GitHub

Veuillez trouvez ci-joint le lien de notre GiHub pour inspecter le projet en profondeur :
https://github.com/Jacky-sDev/infra_JackysDev

Script de déploiement

```
neticien@debian-base:~/infra_JackysDev/ansible$ cat site.yml
---
# MASTER PLAYBOOK - Infrastructure JackysDev
# Ordre : Base -> DB -> Apps -> LB -> VPN


- name: "Phase 9 : Sécurisation VPN & Accès Admin (Alma VM)"
  import_playbook: alma10Vpn/main.yml

# 2. BASES DE DONNÉES (Ordre logique : Couche DB d'abord)
- name: "Phase 2 : Configuration Database Master"
  import_playbook: coucheDB/master_db/setup-database-master.yml

- name: "Phase 3 : Haute Disponibilité & Réplication"
  import_playbook: coucheDB/replication/ha-db-replication.yml

- name: "Phase 1 : Préparation des serveurs"
  import_playbook: backendServ/base.yml
```

```
# 3. APPLICATIONS (Backends)
- name: "Phase 4 : Déploiement des Web Apps"
  import_playbook: backendServ/web_apps/deploy-vaultwarden.yml

- name: "Phase 5 : Déploiement de SimpleLogin"
  import_playbook: backendServ/web_apps/deploy-simplelogin.yml

- name: "Phase 6 : Déploiement de Gitea"
  import_playbook: backendServ/web_apps/deploy-gitea.yml

- name: "Phase 7 : Déploiement d'Authentik"
  import_playbook: backendServ/web_apps/authentik/deploy-authentik.yml

- name: "Phase 3.5 : Haute Disponibilité des Fichiers Gitea"
  import_playbook: coucheDB/replication/ha-gitea-files.yml

# 4. RÉSEAU & ACCÈS
- name: "Phase 8 : Configuration du Load Balancer"
  import_playbook: loadbalancer/setup-loadbalancer.yml
```

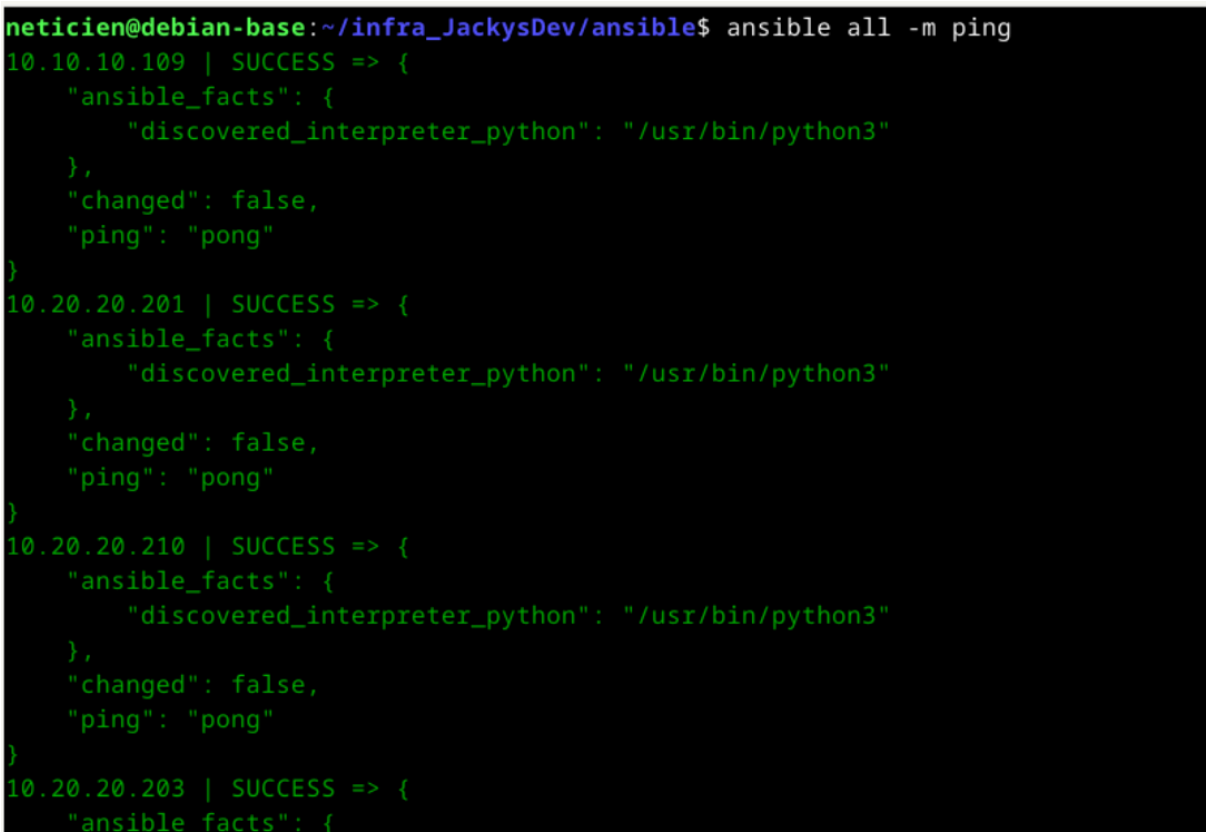
Inventaire yaml

```
neticien@debian-base:~/infra_JackysDev/ansible$ cat inventory.yml
all:
  vars:
    ansible_user: sysadm
    ansible_ssh_private_key_file: /home/neticien/.ssh/id_rsa
    ansible_become_pass: 'chezJ@ckys0n3stSecvre.'

  children:
    vpn:
      hosts:
        10.10.10.109:
    loadbalancer:
      hosts:
        10.20.20.201:
        10.20.20.210:
    backends:
      hosts:
        10.20.20.202:
        10.20.20.203:
    database:
      hosts:
        10.20.20.204:
          db_role: master
        10.20.20.205:
          db_role: slave
```

Cette commande Ansible permet de vérifier rapidement que l’ensemble des machines du parc est accessible et correctement géré par l’outil d’automatisation.
Le module ping envoie une requête de test à chaque hôte présent dans l’inventaire. Lorsque la réponse "pong" est retournée avec le statut SUCCESS, cela confirme que la machine est en ligne, joignable et prête à être administrée via Ansible.

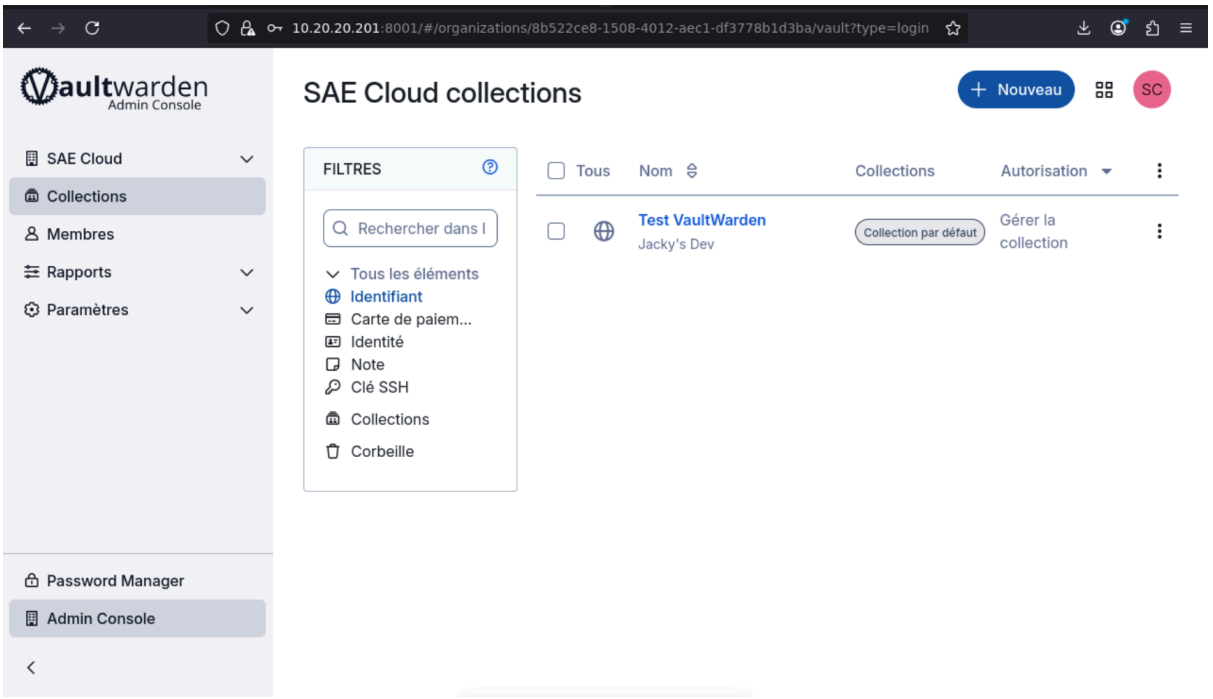
Cette vérification permet donc de s’assurer que tous les serveurs sont opérationnels et sous contrôle avant le déploiement ou l’exécution de tâches automatisées.



Vaultwarden

Dans cette étape, nous utilisons Vaultwarden, un gestionnaire de mots de passe sécurisé auto-hébergé. Il permet de créer et stocker des identifiants et mots de passe de manière chiffrée au sein de l’infrastructure.

Chaque utilisateur peut enregistrer ses identifiants dans une collection, ce qui permet d’organiser et de partager les accès de façon sécurisée entre les membres autorisés. Cette solution permet ainsi de centraliser la gestion des identifiants tout en garantissant la confidentialité et la sécurité des informations sensibles.



Après la création de l’identifiant dans VaultWarden, nous vérifions que l’entrée est bien enregistrée dans la base de données. Pour cela, nous exécutons une requête SQL sur le serveur principal afin d’afficher la dernière entrée créée. Nous effectuons ensuite la même vérification sur le serveur répliqué afin de confirmer que la réplication de la base de données fonctionne correctement. Sur la capture, on peut constater que la même entrée apparaît sur le serveur maître ainsi que sur le serveur esclave, ce qui confirme que la réplication des données est opérationnelle.

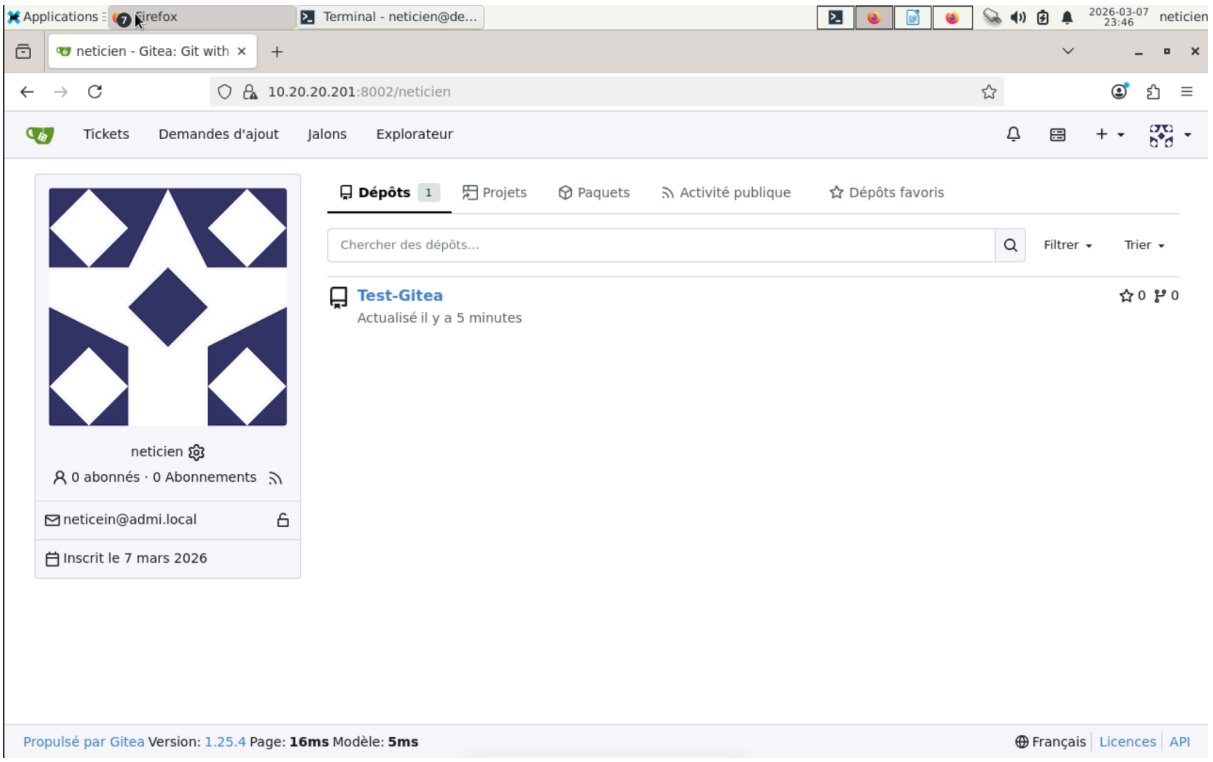
```
neticien@debian-base:~/infra_JackysDev/ansible$ ssh sysadm@10.20.20.204 "docker exec vaultwarden_db psql -U vw_admin -d vault
warden -c '\x' -c 'SELECT uuid, created_at, name FROM ciphers ORDER BY created_at DESC LIMIT 1;'"
Expanded display is on.
-[ RECORD 1 ]----------------------------------------------------------------------------------------------------------------
----
uuid       | 101f4627-0b09-4860-9087-79e938dd8a66
created_at | 2026-03-07 22:15:29.792796
name       | 2.Qiq0xeknJC6qjzzjZGt9Fg==|0kDIgrUQN5q3kEr/TQ/YgU3RfGHSb04wP4lBoST5Wso=|MdrrNVVvpyelOcSqgwYwXq4KADgYrPmzLhXWEYmq
4so=

neticien@debian-base:~/infra_JackysDev/ansible$ ssh sysadm@10.20.20.205 "docker exec vaultwarden_db psql -U vw_admin -d vault
warden -c '\x' -c 'SELECT uuid, created_at, name FROM ciphers ORDER BY created_at DESC LIMIT 1;'"
Expanded display is on.
-[ RECORD 1 ]----------------------------------------------------------------------------------------------------------------
----
uuid       | 101f4627-0b09-4860-9087-79e938dd8a66
created_at | 2026-03-07 22:15:29.792796
name       | 2.Qiq0xeknJC6qjzzjZGt9Fg==|0kDIgrUQN5q3kEr/TQ/YgU3RfGHSb04wP4lBoST5Wso=|MdrrNVVvpyelOcSqgwYwXq4KADgYrPmzLhXWEYmq
4so=
```

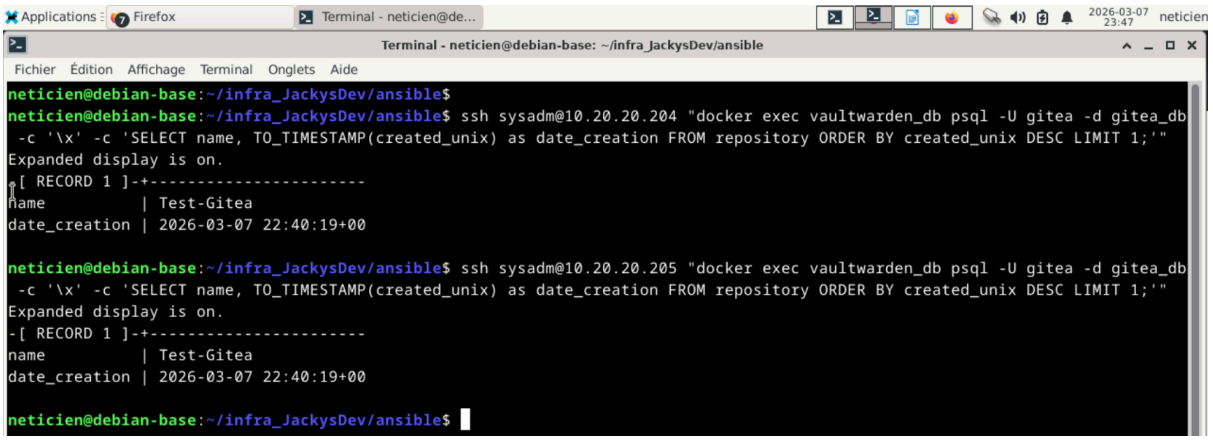
Gitea

Dans cette étape, nous créons un dépôt Git sur l’interface Gitea.
Gitea est une plateforme de gestion de dépôts Git auto-hébergée qui permet de centraliser le code source et de faciliter la collaboration entre les développeurs.

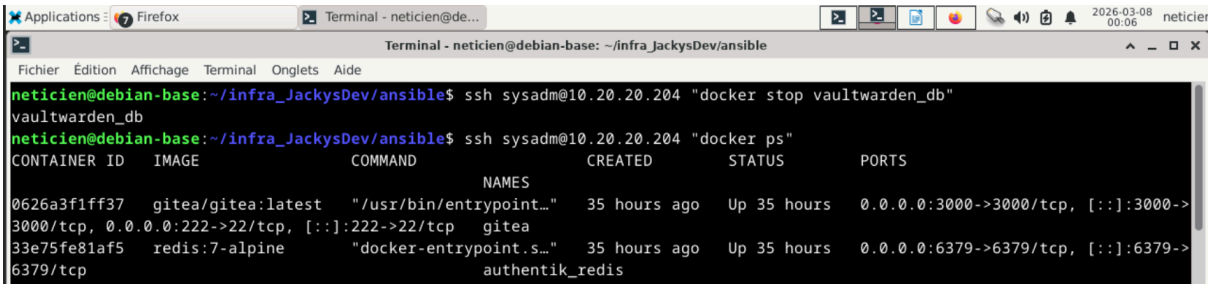
Le dépôt créé servira à stocker et versionner les fichiers du projet, permettant ainsi de suivre les modifications, de gérer les différentes versions du code et de faciliter le travail collaboratif au sein de l’infrastructure.



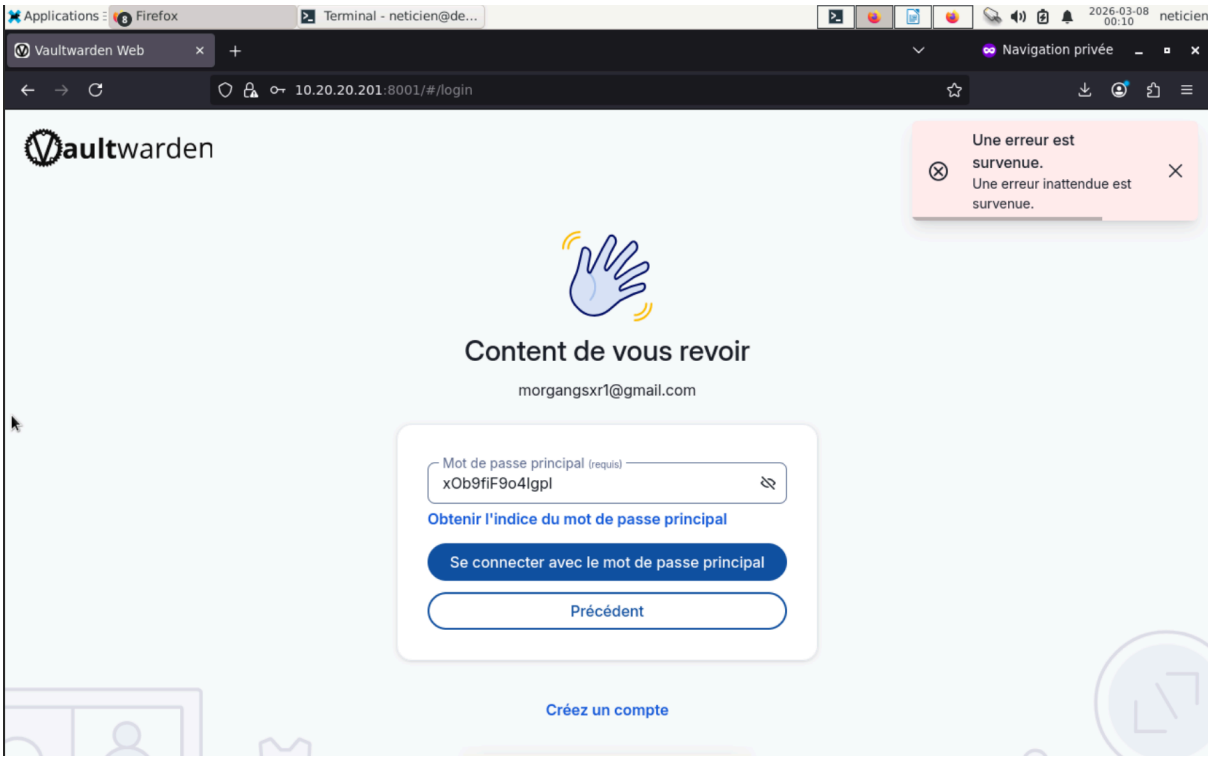
encore une fois on vérifie :



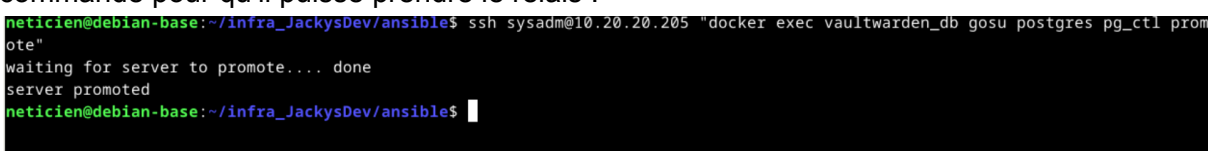
On test le FailOver:
On vient stopper le service vaultwarden sur le maître



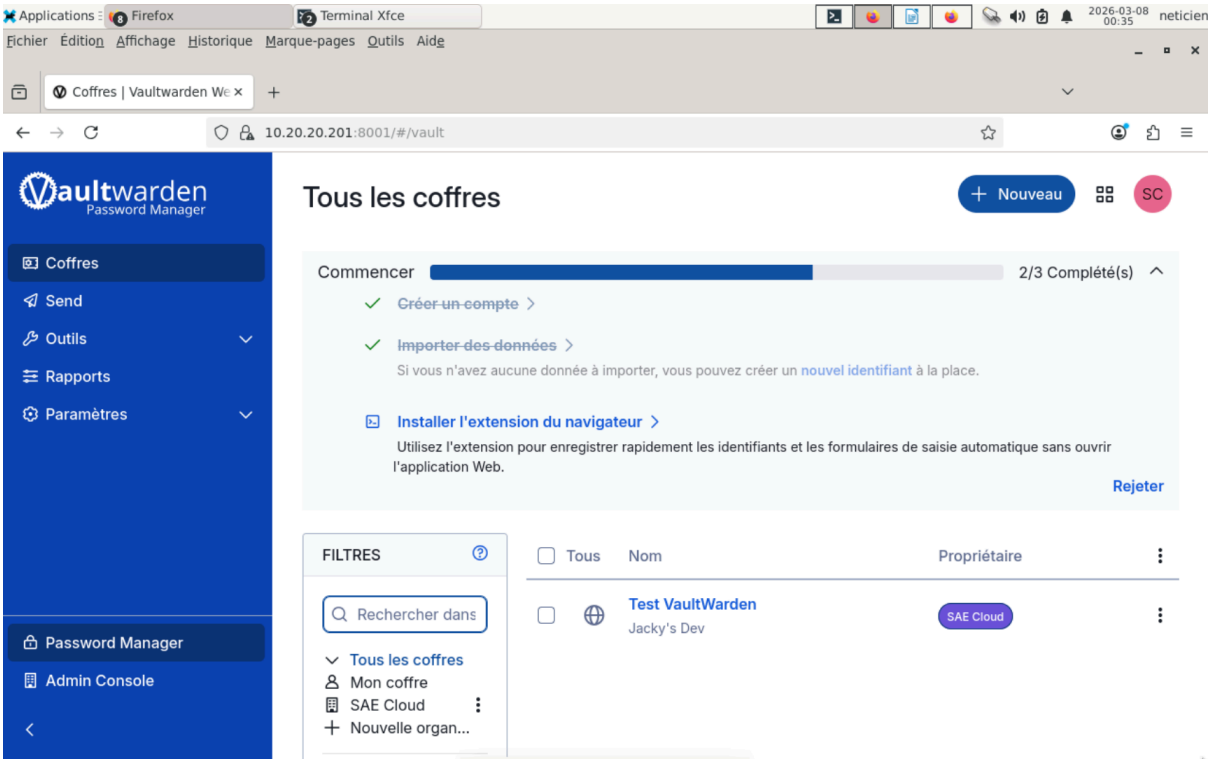
Lorsque nous essayons de nous reconnecter sur Vaultwarden, nous avons une erreur, cette erreur est dû au fait que nous venons de couper la base de donnée sur le maître, elle n’arrive donc pas à la joindre.



Pour régler ce problème, nous venons assigner le rôle de maître à l’esclave en une seule commande pour qu’il puisse prendre le relais :



Et comme nous pouvons l’observer, la connexion à Vaultwarden marche de nouveau:



Ficher OpenTofu

```
neticien@debian-base:~/infra_JackysDev/opentofu/jdp1$ cat main.tf
terraform {
  required_providers {
    proxmox = {
      source  = "telmate/proxmox"
      version = "3.0.2-rc07"
    }
  }
}

# 1. Connexion à Proxmox
provider "proxmox" {
  pm_api_url                  = "https://10.10.10.1:8006/api2/json"
  pm_api_token_id             = "root@pam!tofu01"
  pm_api_token_secret         = "a7bdf188-d6db-4a38-9bbe-0d19239f07a8"
  pm_tls_insecure             = true
  pm_minimum_permission_check = false
}

# ---------------------------------------------------------
# 2. La porte d'entrée VPN (AlmaLinux)
# ---------------------------------------------------------
resource "proxmox_vm_qemu" "vpn_gateway" {
  name        = "vpn-server"
  description = "Accès VPN unique au sous-réseau sécurisé"
  target_node = "Proxmox-VE"
  clone       = "alma10-template"
  agent       = 1
  memory      = 2048

  full_clone  = true
  scsihw      = "virtio-scsi-pci"
  boot        = "order=scsi0"

  cpu {
    cores   = 3
    sockets = 1
    type    = "host"
  }
  disks {
    scsi {
      scsi0 {
        disk {
          storage = "local-lvm"
          size    = 30
        }
      }
    }
```

```
    ide {
      ide1 {
        cloudinit {
          storage = "local-lvm"
        }
      }
    }
  }

  # --- CONFIGURATION CLOUD-INIT ---
  os_type = "cloud-init"
  # Interface publique (vmbr1)
  ipconfig0 = "ip=10.10.10.109/24,gw=10.10.10.1"
  # Interface privée (vmbr2)
  ipconfig1 = "ip=10.20.20.254/24" # Le VPN agit comme passerelle pour le réseau privé

  network {
    id     = 0
    model  = "virtio"
    bridge = "vmbr1"
  }
  network {
    id     = 1
    model  = "virtio"
    bridge = "vmbr2"
  }
}

# ------------------------------------------------------------
# 3. Les Load Balancers Redondants (Ubuntu)
# ------------------------------------------------------------
resource "proxmox_vm_qemu" "load_balancers" {
  count       = 2 # On déploie 2 instances pour la Haute Disponibilité
  name        = "load-balancer-0${count.index + 1}"
  description = "Répartiteur de charge HA - Noeud ${count.index + 1}"
  target_node = "Proxmox-VE"
  clone       = "ubuntu-template"
  agent       = 1
  memory      = 2048

  full_clone  = true
  scsihw      = "virtio-scsi-pci"
  boot        = "order=scsi0"

  cpu {
    cores   = 2
    sockets = 1
    type    = "host"
```

```
  disks {
    scsi {
      scsi0 {
        disk {
          storage = "local-lvm"
          size    = 30
        }
      }
    }
    ide {
      ide1 {
        cloudinit {
          storage = "local-lvm"
        }
      }
    }
  }

  # --- CONFIGURATION CLOUD-INIT ---
  os_type = "cloud-init"
  # Si count.index = 0 -> .201. Si count.index = 1 -> .210
  ipconfig0 = "ip=10.20.20.2${count.index == 0 ? "01" : "10"}/24,gw=10.20.20.254"

  network {
    id     = 0
    model  = "virtio"
    bridge = "vmbr2"
  }
}

# ------------------------------------------------------------
# 4. Les serveurs Backend redondants (Ubuntu)
# ------------------------------------------------------------
resource "proxmox_vm_qemu" "backend_servers" {
  count       = 2
  name        = "backend-srv-0${count.index + 1}"
  description = "Serveur d'application redondant ${count.index + 1}"
  target_node = "Proxmox-VE"
  clone       = "ubuntu-template"
  agent       = 1
  memory      = 16384

  full_clone  = true
  scsihw      = "virtio-scsi-pci"
  boot        = "order=scsi0"

  cpu {
    cores   = 4
    sockets = 1
    type    = "host"
```

```
  disks {
    scsi {
      scsi0 {
        disk {
          storage = "local-lvm"
          size    = 30
        }
      }
    }
    ide {
      ide1 {
        cloudinit {
          storage = "local-lvm"
        }
      }
    }
  }

  # --- CONFIGURATION CLOUD-INIT ---
  os_type = "cloud-init"
  # Génère automatiquement les IPs 10.20.20.202 et 10.20.20.203
  ipconfig0 = "ip=10.20.20.20${count.index + 2}/24,gw=10.20.20.254"

  network {
    id     = 0
    model  = "virtio"
    bridge = "vmbr2"
  }
}

# ---------------------------------------------------------
# 5. Les serveurs Vaultwarden (AlmaLinux)
# ---------------------------------------------------------
resource "proxmox_vm_qemu" "vaultwarden_servers" {
  count       = 2
  name        = "vw-srv-0${count.index + 1}"
  description = "Serveur Vaultwarden Sécurité ${count.index + 1}"
  target_node = "Proxmox-VE"
  clone       = "alma10-template"
  agent       = 1
  memory      = 4096

  full_clone  = true
  scsihw      = "virtio-scsi-pci"
  boot        = "order=scsi0"

  cpu {
    cores   = 2
    sockets = 1
    type    = "host"
```

```
  disks {
    scsi {
      scsi0 {
        disk {
          storage = "local-lvm"
          size    = 30
        }
      }
    }
    ide {
      ide1 {
        cloudinit {
          storage = "local-lvm"
        }
      }
    }
  }

  # --- CONFIGURATION CLOUD-INIT ---
  os_type = "cloud-init"
  # Génère automatiquement les IPs 10.20.20.204 et 10.20.20.205
  ipconfig0 = "ip=10.20.20.20${count.index + 4}/24,gw=10.20.20.254"

  network {
    id     = 0
    model  = "virtio"
    bridge = "vmbr2"
  }
}
```