Proxmox Propre

DIN — Serveur Proxmox

Sommaire

ACTE I — CONTEXTE ET OBJECTIFS

I.1. Introduction

I.1.1 Objectif de la DIN

Cette DIN (Document d'Installation) a pour objectif de décrire de manière claire, structurée et reproductive l'installation et la configuration initiale d'un serveur Proxmox VE 9.1 au sein du projet d'infrastructure mené à l'IUT. Elle formalise les paramètres retenues et les étapes réalisées afin de permettre :

- la traçabilité des choix techniques,
- la reproductibilité de l'installation sur un serveur équivalent,
- la validation par l'équipe projet et l'encadrement,
- la mise à disposition d'une base fiable pour les documents associés (DAT /DEX).

I.1.2 Contexte du projet global

Le serveur Proxmox documenté ici s'inscrit dans un projet pédagogique d'infrastructure DevOps dont l'objectif finale est de mettre en place un environnement complet de gestion de projet de développement, comprenant notamment :

- une plateforme de virtualisation permettant d'héberger des services,
- des composants liés au DevOps (Ex : serveur Git, serveur Web),
- une logique de stockage centralisée (baie SAN) et de sauvegarde,
- des éléments d'automatisation (outillage, script, IaC).

Le parc matériel prévu pour ce projet comprend notamment trois serveurs Dell R640 destinés à former un environnement Proxmox, ainsi qu'une baie SAN (30 To - ~22 To) et les équipements réseaux associés (switch/ routeur). L'installation décrite dans cette DIN constitue donc une étape fondatrice du projet, en posant un socle de virtualisation exploitable pour les phases suivantes.

I.1.3 Place du serveur Proxmox dans le projet

Le serveur Proxmox VE 9.1 couvert par ce document correspond à un nœud de virtualisation destiné à accueillir des machines virtuelles (et, selon les besoins, des conteneurs) nécessaires aux différents services du projet.

À l'échelle de l'infrastructure cible, ce serveur a vocation à :

- participer à un ensemble Proxmox multi-noeud (objectif de cluster),
- s'appuyer sur un stockage centralisé (SAN) pour héberger les données critiques,
- permettre l'hébergement et la bascule VM dans une logique de continuité de service,
- servir de base à l'intégration des solutions de sauvegarde (Ex : Proxmox Backup Server).

La DIN se concentre cependant sur l'installation et la configuration initiale de ce serveur, en distinguant clairement ce qui relève de la mise en place effective et ce qui relève des évolutions prévues.

I.1.4 Périmètre de la DIN

Le périmètre de la DIN couvre les éléments suivants :

- l'installation de Proxmox VE 9.1 (choix du mode d'installation, sélection du disque, paramètre régionaux),
- la configuration initiale du serveur (identifiants d'administration, adresse mail d'alerte),
- la configuration réseau de management (interface, adressage IP, passerelle, DNS, FQDN),
- les ajustements post-installation nécessaires à la conformité de la configuration (fichiers systèmes et vérifications).
- un volet troubleshooting (diagnostic et vérifications de base), adapté au contexte de projet.

Les captures d'écrans utilisées dans le document sont fournies à titre illustratif afin de présenter les interfaces et étapes de l'installateur de Proxmox VE 9.1 ; elles peuvent ne pas refléter exactement les valeurs finales de configuration appliquées sur l'infrastructure de l'IUT.

I.1.5 Éléments hors périmètre (cluster finale, automatisation, DevOps)

Ne font pas partie du périmètre de cette DIN, car relevant, d'autres livrables (DAT/ DEX) ou phases ultérieures du projet :

- la conception détaillée et la mise en œuvre complète du cluster Proxmox à 3 nœuds (bascule/HA, règles, tests),
- la configuration approfondie du stockage SAN (formatage, ZPS, intégration avancée),
- L'installation et la configuration complète de Proxmox Backup Server et des politiques de sauvegarde,

- le déploiement des services DevOps (serveur Git, services web, applications),
- les mécanismes d’automatisation (scripts de déploiement VM, webapp de provisionning, Ansible/Terraform),
- l’outillage de type imagerie/restauration des serveurs et les scénarios de reprise avancés.

Ces éléments sont néanmoins mentionnés lorsque nécessaire afin de situer l’installation du serveur Proxmox dans la trajectoire du projet global, sans anticiper sur les décisions détaillées d’architecture et d’exploitation

ACTE II — PRÉSENTATION DE LA SOLUTION PROXMOX

II.1 Présentation de Proxmox VE

II.1.1 Présentation générale de Proxmox VE

Proxmox VE (Virtual Environment) est une plateforme de virtualisation open source permettant d'héberger et d'administrer des machines virtuelles (VM) et des conteneurs au sein d'un même environnement. Elle s'appuie sur des technologies éprouvées :

- KVM pour la virtualisation complète des VM,
- LXC pour la conteneurisation,
- une administration centralisée via une interface web et des outils en ligne de commande.

Proxmox VE regroupe, dans une console unique, les fonctions attendues d'une plateforme de virtualisation moderne : gestion des ressources (CPU/RAM/stockage), supervision de base, administration réseau, gestion de stockage local ou partagé, et fonctionnalités orientées haute disponibilité et cluster lorsque l'infrastructure le permet.

Dans le cadre de ce projet, la version retenue et installée est Proxmox VE 9.1.

II.1.2 Choix de Proxmox VE pour le projet

Le choix de Proxmox VE 9.1 a été retenu au regard des contraintes et objectifs du projet pédagogique :

- Adéquation au besoin : héberger des services (Git, web, automatisation) sous forme de VM/CT dans un environnement contrôlé.
- Approche "infrastructure réelle" : Proxmox est une solution largement utilisée en contexte professionnel, ce qui permet de travailler sur des pratiques transférables.
- Gestion multi-nœuds : la solution permet la construction d'un cluster et l'activation de mécanismes de bascule/HA à partir de plusieurs serveurs, ce qui correspond à l'objectif projet (3 serveurs).
- Compatibilité stockage partagé : Proxmox s'intègre avec des stockages externes et peut s'appuyer sur une baie SAN, ce qui correspond à l'architecture matérielle prévue.

- Facilité de déploiement et d'administration : installation relativement directe et administration unifiée via interface web, adaptée à un contexte de travaux pratiques et de validation.

Ce choix vise donc à fournir un socle de virtualisation stable pour la suite du projet, tout en permettant l'apprentissage de notions clés (virtualisation, réseau, stockage, administration, continuité de service).

II.1.3 Rôle du serveur Proxmox (nœud de virtualisation)

Le serveur Proxmox décrit par cette DIN correspond à un nœud de virtualisation, c'est-à-dire un serveur physique exécutant Proxmox VE afin de :

- héberger des VM et/ou conteneurs nécessaires au projet,
- fournir des ressources calcul (CPU/RAM) et un accès au stockage,
- exposer une interface d'administration (web) pour la gestion et la supervision,
- s'intégrer à terme à un environnement Proxmox multi-nœuds (cluster), en vue d'expérimenter la bascule et la continuité de service.

Dans l'immédiat, l'objectif du serveur est d'être pleinement opérationnel sur les aspects fondamentaux (installation, accès, réseau de management), afin de servir de base aux étapes ultérieures (stockage partagé, sauvegardes, services DevOps, automatisation).

ACTE III — ARCHITECTURE DU SERVEUR

III.1. Architecture matérielle

III.1.1 Serveur Dell R640

L'infrastructure matérielle mise à disposition pour le projet repose sur des serveurs Dell PowerEdge R640. Dans le cadre de cette DIN, le serveur concerné correspond à l'un des nœuds destinés à héberger Proxmox VE.

Le Dell R640 est un serveur rack 1U orienté datacenter, adapté à des charges de virtualisation grâce à :

- une architecture bi-processeur,
- une capacité mémoire élevée,
- une intégration standard en baie,
- des interfaces réseau multi-débits selon la carte/ports disponibles.

III.1.2 Ressources matérielles

D'après le cadrage du projet, les ressources matérielles disponibles pour les nœuds Proxmox sont dimensionnées pour permettre l'hébergement de plusieurs VM/CT.

Les ressources globales annoncées pour chaque serveur sont :

- Processeurs : 2 processeurs (bi-proc)
- Mémoire vive : 256 Go de RAM
- Stockage local : entre 1 To et 2 To (selon configuration)

Ces ressources sont cohérentes avec les objectifs pédagogiques du projet : héberger des services (Git, web, automatisation), tester des scénarios de bascule, et disposer d'une marge pour des environnements clients (ex. VDI) si nécessaire.

(Si vous avez la configuration précise des CPU — modèle, nombre de cœurs/threads — on l'ajoutera ici pour une description complète.)

III.1.3 Contraintes matérielles et environnement IUT

Le déploiement s'effectue dans un contexte pédagogique en environnement IUT, ce qui impose plusieurs contraintes et bonnes pratiques :

- Accès matériel : interventions réalisées sur des équipements partagés et/ou accessibles sur créneaux, ce qui peut limiter les tests longs.
- Réseau : dépendance au plan d'adressage et aux services réseau disponibles (DNS/DHCP), avec nécessité de cohérence et de validation.
- Stockage centralisé : présence d'une baie SAN (30 To - ~22 To) destinée à accueillir le "cœur" des données du projet ; l'intégration complète peut être réalisée en phase ultérieure.
- Traçabilité : nécessité de documenter les paramètres retenus et les étapes d'installation afin de faciliter la reprise par l'équipe et l'encadrement.

Dans ce contexte, l'installation Proxmox est réalisée avec une approche pragmatique : mise en service rapide et stable du nœud, puis intégrations progressives (cluster, stockage partagé, sauvegardes) au fil des jalons projet.

III.2. Architecture logicielle

III.2.1 Système hôte (Proxmox VE basé sur Debian)

Proxmox VE s'appuie sur une base Debian (distribution Linux) enrichie par les composants nécessaires à la virtualisation et à l'administration.

Cette architecture logicielle fournit :

- un noyau Linux adapté à la virtualisation,
- l'hyperviseur KVM pour l'exécution des VM,
- LXC pour la gestion des conteneurs,
- une couche de gestion (services Proxmox) et une interface web d'administration.

Cette base Debian permet également de s'appuyer sur des mécanismes standards d'administration système (fichiers de configuration, services, journalisation), ce qui est utile en contexte pédagogique.

III.2.2 Version utilisée

La version installée et utilisée dans le cadre de ce projet est Proxmox VE 9.1.

Cette information est essentielle pour :

- garantir la reproductibilité des procédures,
- assurer la cohérence avec les guides/procédures,

- ❖ faciliter le diagnostic en cas de problème (compatibilités, comportements attendus).

III.2.3 Mode de déploiement (serveur standalone, futur cluster)

Le serveur couvert par cette DIN est installé et configuré comme un serveur Proxmox standalone (nœud autonome) afin de valider :

- ❖ l'installation,
- ❖ l'accès administrateur,
- ❖ la configuration réseau de management,
- ❖ la stabilité initiale.

À l'échelle du projet, ce nœud a vocation à être intégré à un cluster Proxmox multi-nœuds (3 serveurs), permettant de travailler sur :

- ❖ la gestion centralisée multi-serveurs,
- ❖ la bascule et la continuité de service,
- ❖ l'utilisation de stockage partagé (SAN) comme support des VM,
- ❖ l'intégration d'une solution de sauvegarde (Proxmox Backup Server).

Les étapes de création du cluster et les mécanismes avancés (HA/bascule, règles, validations) ne sont pas détaillés dans la présente DIN, et seront traités dans les livrables appropriés (DAT/DEX) et/ou phases ultérieures du projet.

ACTE IV — INSTALLATION DE PROXMOX VE

IV.1. Préparation à l'installation

IV.1.1 Image ISO utilisée

L'installation de l'hyperviseur est réalisée à partir de l'image officielle Proxmox VE 9.1 (ISO). L'ISO est préparée sur un support de démarrage (ex. clé USB bootable) afin de permettre l'installation sur le serveur.

Remarque : les captures d'écran de cette section sont fournies à titre illustratif (interfaces et étapes de l'installateur). Les paramètres de référence (IP, FQDN, identifiants, etc.) sont ceux consignés dans le présent document.

IV.1.2 Choix du mode d'installation

Le mode Install Proxmox VE (Graphical) est utilisé.

Ce choix est justifié par :

- une procédure guidée et standardisée,
- une réduction des erreurs de saisie en contexte pédagogique,
- une facilité de vérification des paramètres avant lancement effectif de l'installation.

IV.1.3 Pré-requis et précautions

Avant de démarrer l'installation, les prérequis et précautions suivants sont pris en compte :

- Le serveur cible est disponible (accès console/écran ou iDRAC selon le contexte).
- Le support d'installation (USB/ISO) est fonctionnel et prioritaire dans l'ordre de boot.
- Attention : l'installation Proxmox implique l'écriture sur le disque sélectionné. Les données présentes sur le disque cible sont susceptibles d'être écrasées.
- Les paramètres nécessaires sont préparés en amont :
  - nom d'hôte / FQDN,
  - adresse IP de management, masque, passerelle, DNS,
  - identifiants d'administration et adresse mail d'alerte.

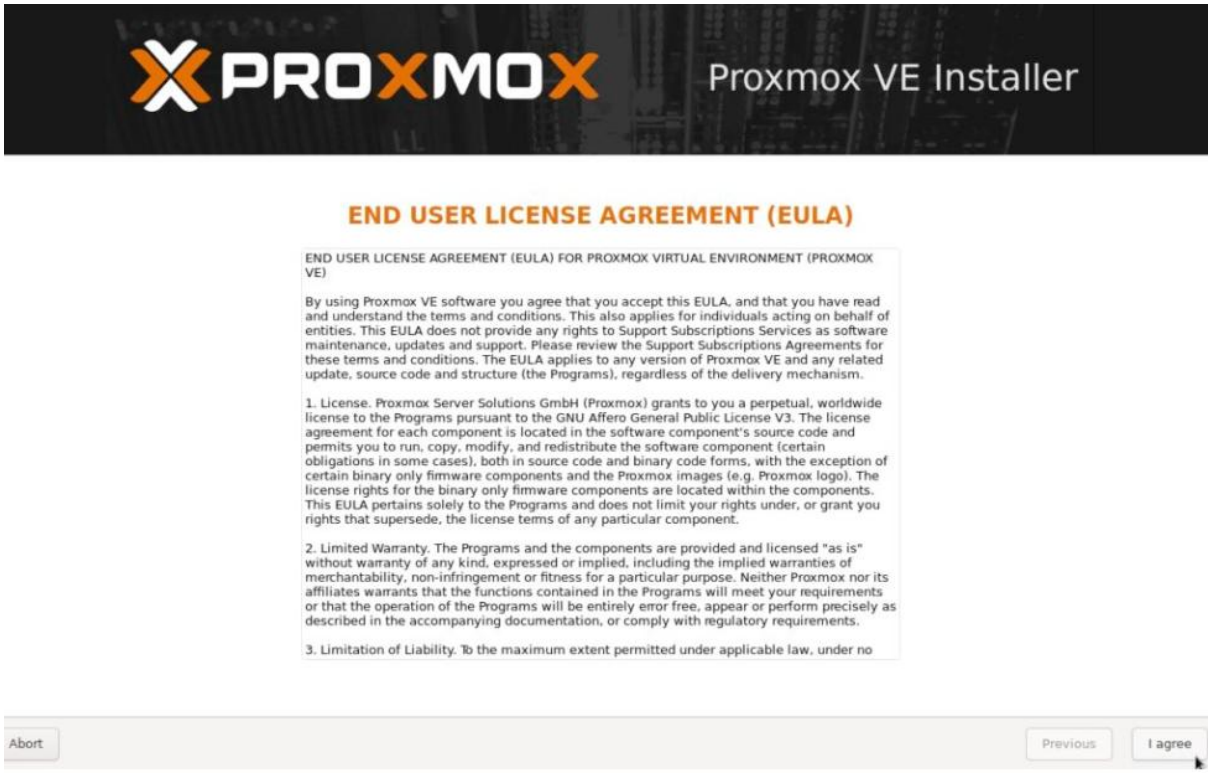
VI.2. Procédure d'installation

IV.2.1 Démarrage sur le support d'installation

1. Démarrer le serveur sur le support contenant l'ISO Proxmox VE.
2. Sélectionner l'option Install Proxmox VE (Graphical).
3. Patienter durant le chargement de l'installateur.

IV.2.2 Acceptation de la licence

1. Lire les conditions d'utilisation.
2. Accepter le End User License Agreement (EULA) afin de poursuivre l'installation.

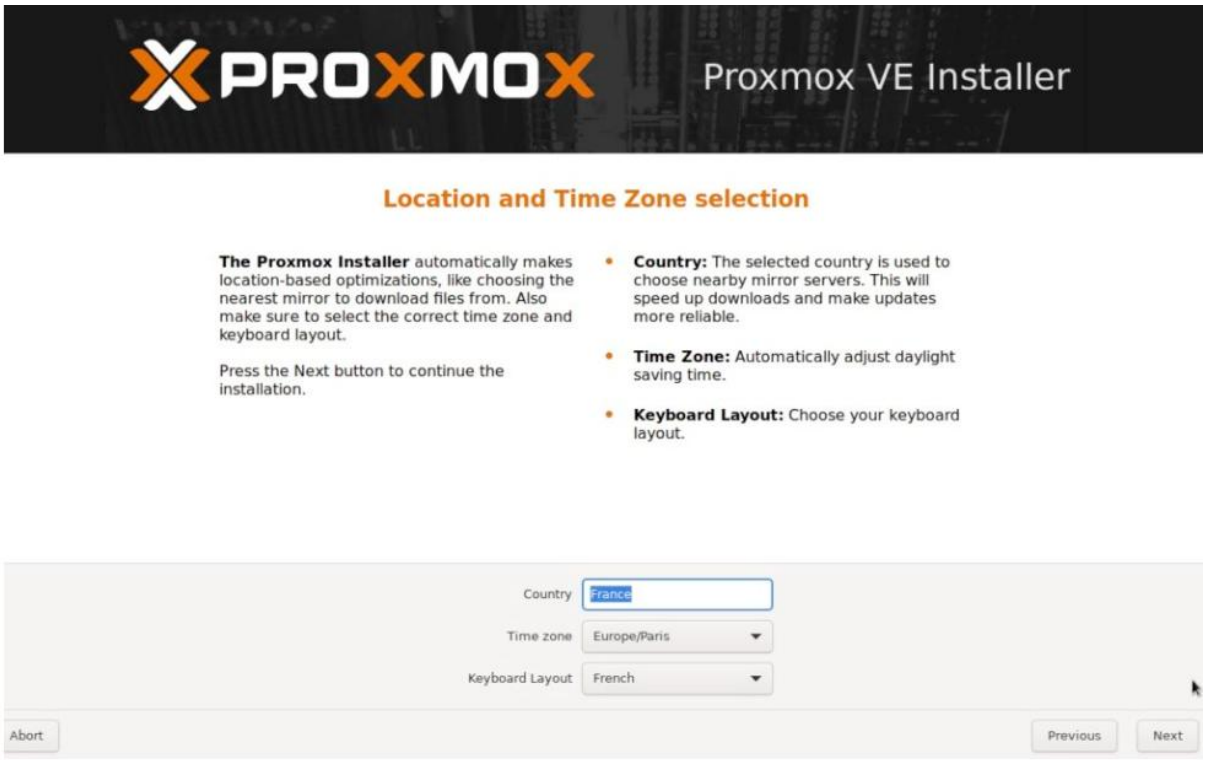


IV.2.3 Sélection du disque

1. Sélectionner le disque destiné à recevoir Proxmox VE.
2. Vérifier que le disque choisi correspond bien au support prévu (afin d'éviter toute installation sur un disque non souhaité).

IV.2.4 Configuration régionale

1. Définir les paramètres régionaux (pays / langue si applicable).
2. Vérifier le fuseau horaire et la configuration clavier si l'assistant le propose.



Ces paramètres garantissent la cohérence des journaux (logs) et des horodatages, utiles pour l'exploitation et le troubleshooting.

IV.2.5 Création des accès administrateur

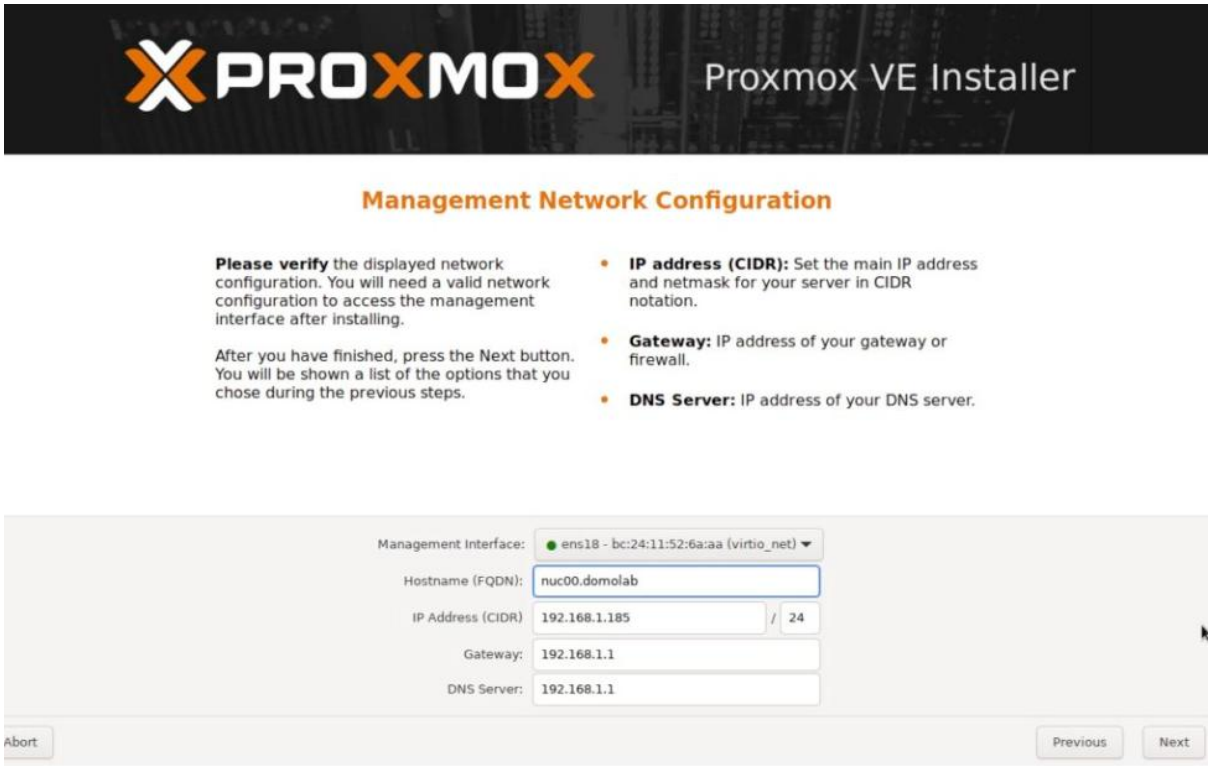
1. Définir le mot de passe administrateur (compte root).

2. Renseigner une adresse mail d'alerte (réception des notifications Proxmox).

Convention projet (contexte pédagogique) : le mot de passe peut être construit à partir du Service Tag, suivi d'un caractère spécial (ex. « ! »), selon la règle définie par l'équipe.

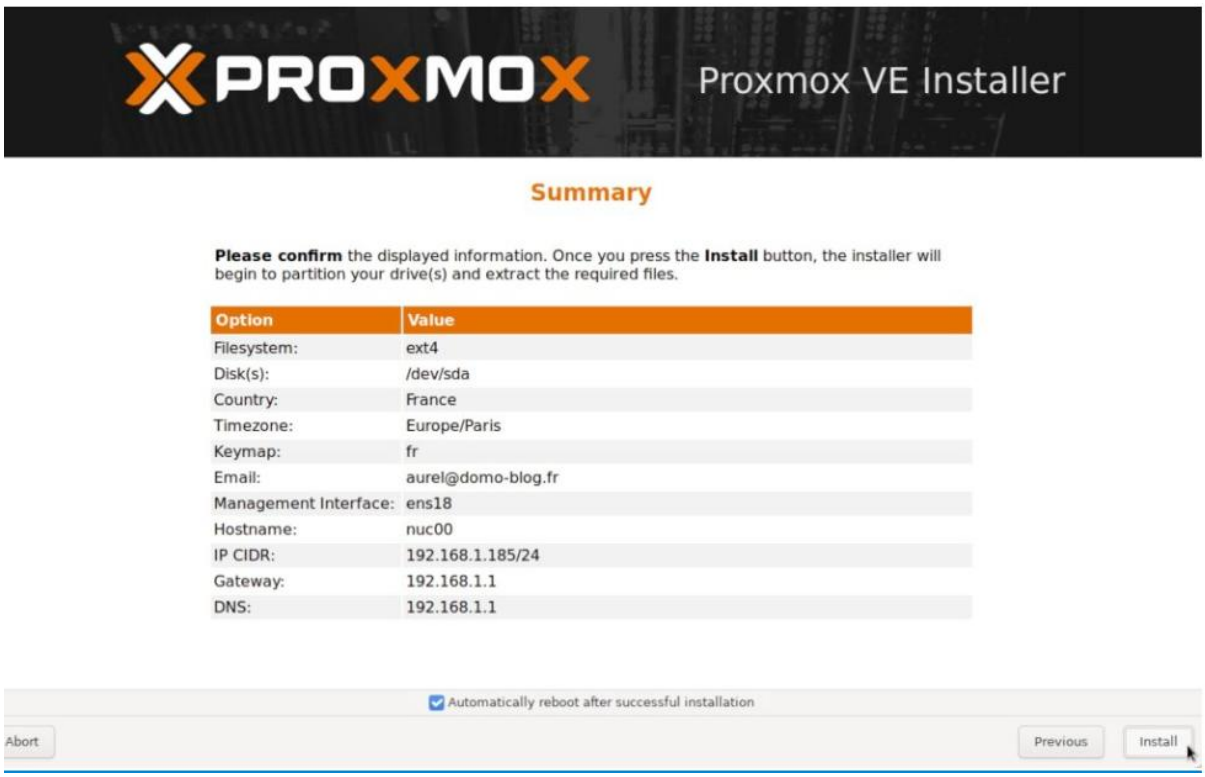
IV.2.6 Configuration réseau initiale

1. Choisir l'interface réseau utilisée pour le réseau de management.
    - Lorsque plusieurs ports sont disponibles, sélectionner de préférence le port offrant le meilleur débit / la meilleure compatibilité avec le câblage en place.
2. Renseigner les paramètres réseau :
    - FQDN du serveur (ex. format `proxmoxX.domaine.local`),
    - Adresse IP de management,
    - Passerelle (gateway),
    - DNS.



IV.2.7 Validation et lancement de l'installation

1. Vérifier le récapitulatif des informations (disque, région, accès, réseau).
2. Lancer l'installation.
3. À la fin de l'installation, redémarrer le serveur si demandé.



Une fois le serveur redémarré, l’accès à l’interface d’administration web Proxmox peut être vérifié (détails traités dans les sections suivantes de la DIN).

ACTE V — CONFIGURATION POST-INSTALLATION

V.1. Paramètres d'administration

V.1.1 Compte administrateur

À l'issue de l'installation, l'administration du serveur Proxmox s'effectue via le compte **root** (administrateur système). Ce compte dispose des privilèges nécessaires pour :

- accéder à l'interface web d'administration,
- réaliser les opérations de configuration initiale,
- administrer l'hyperviseur et les ressources de virtualisation.

Dans le cadre du projet, l'usage du compte root est conservé pour la mise en place initiale et les validations pédagogiques.

V.1.2 Mot de passe

Le mot de passe administrateur est celui défini lors de l'installation.

Convention projet (contexte pédagogique) : afin de faciliter la gestion par l'équipe et l'encadrant, le mot de passe peut suivre une règle de construction simple et traçable (ex. *Service Tag + caractère spécial*). Les valeurs exactes sont consignées dans la fiche de paramètres du projet.

V.1.3 Adresse mail d'alerte

Une adresse mail d'alerte est configurée lors de l'installation afin de permettre la réception des notifications Proxmox (alertes système, informations d'administration, etc.).

L'adresse retenue est documentée dans les paramètres de configuration du serveur.

V.2. Configuration réseau

V.2.1 Interface de management

Le serveur Proxmox dispose d'une interface réseau dédiée au management (accès interface web, administration, supervision).

L’interface retenue est celle sélectionnée lors de l’installation. Lorsque plusieurs ports réseau sont disponibles sur le serveur, l’interface privilégiée est celle offrant :

- un débit supérieur (selon disponibilité),
- une compatibilité directe avec le câblage en place,
- une stabilité d’accès pour l’administration.

V.2.2 Nom d’hôte et FQDN

Le serveur est identifié par un nom d’hôte et un FQDN (Fully Qualified Domain Name) cohérents avec le domaine interne du projet.

Le FQDN permet notamment :

- une identification claire dans l’interface Proxmox,
- une meilleure lisibilité dans les journaux,
- une intégration facilitée avec les services réseau (DNS) lorsqu’ils sont disponibles.

V.2.3 Adressage IP

L’adressage IP du serveur correspond à l’adresse de management utilisée pour l’accès à Proxmox.

Les éléments suivants doivent être définis et documentés :

- adresse IP,
- masque (ou préfixe CIDR),
- réseau associé.

V.2.4 Passerelle et DNS

La configuration réseau inclut :

- une passerelle (gateway) permettant l’accès aux autres sous-réseaux si nécessaire,
- un ou plusieurs serveurs DNS permettant la résolution de noms (notamment du FQDN).

Dans le cadre du projet, un point de vigilance est identifié sur la disponibilité et la cohérence des services DNS/DHCP ; des solutions alternatives peuvent être prévues (poste dédié DNS/DHCP ou usage du routeur en segmentation/DMZ), selon les contraintes de l’environnement.

V.3. Ajustements système

V.3.1 Configuration réseau avancée

Après installation, certains ajustements peuvent être nécessaires afin d'aligner la configuration réseau sur le plan d'adressage réel (notamment si une modification a été effectuée après la phase d'installation).

Les fichiers de configuration réseau peuvent être modifiés pour mettre à jour l'interface concernée et les paramètres associés.

V.3.2 Mise à jour des fichiers système

Les principaux fichiers pouvant être ajustés dans ce contexte sont :

- `/etc/network/interfaces` : mise à jour de la configuration de l'interface (adresse IP, masque, etc.),
- `/etc/hosts` : cohérence entre l'adresse IP locale et le nom d'hôte/FQDN.

Ces modifications permettent d'assurer une configuration stable et cohérente pour l'accès web et la résolution locale.

V.3.3 Vérifications et redémarrage

Après toute modification réseau, des vérifications sont réalisées afin de valider la prise en compte de la configuration :

- vérification des interfaces et des adresses configurées,
- test de connectivité (selon périmètre réseau disponible),
- validation de l'accès à l'interface web Proxmox.

Un **redémarrage** du serveur peut être effectué si nécessaire afin de garantir l'application complète des paramètres et la stabilité de l'environnement.

ACTE VI — INTÉGRATION DANS L'INFRASTRUCTURE GLOBALE

VI.1. Intégration au projet global

> **Principe (très important)** : cette section présente l'intention d'intégration du serveur Proxmox dans l'infrastructure globale du projet. Elle ne détaille pas la mise en œuvre complète (cluster, stockage partagé, sauvegardes), qui relève des phases ultérieures et/ou des livrables dédiés (DAT/DEX).

VI.1.1 Rôle du serveur dans le futur cluster Proxmox

Le serveur Proxmox installé et configuré via cette DIN constitue un nœud de virtualisation destiné à être intégré, à terme, dans un cluster Proxmox multi-nœuds.

Dans la trajectoire du projet, l'intégration en cluster a pour objectifs :

- centraliser l'administration des nœuds (gestion unifiée),
- faciliter la répartition des charges de virtualisation,
- permettre des scénarios de bascule et de continuité de service (selon les capacités mises en place).

À ce stade, la DIN garantit que le nœud est prêt sur les fondamentaux (installation, accès, réseau de management), condition nécessaire avant toute intégration cluster.

VI.1.2 Lien avec la baie SAN

Le projet prévoit l'utilisation d'une baie SAN (30 To - ~22 To) destinée à accueillir le stockage central des données et, potentiellement, le stockage des VM.

L'intention est de disposer d'un stockage partagé permettant :

- une localisation centralisée des données critiques,
- une meilleure portabilité des VM entre nœuds,
- une base technique cohérente pour des tests de bascule.

La présente DIN n'implémente pas l'intégration complète de la baie SAN ; elle positionne toutefois le serveur Proxmox dans une logique compatible avec ce stockage (réseau stable, identification claire du nœud, socle prêt pour les étapes suivantes).

VI.1.3 Préparation à la haute disponibilité et aux sauvegardes

L'infrastructure cible vise à expérimenter des mécanismes de continuité de service et de sauvegarde dans un cadre pédagogique.

L'intention projet inclut notamment :

- l'étude et la mise en place de la bascule de VM (selon les fonctionnalités activées en cluster),
- l'installation d'une solution de sauvegarde dédiée, telle que Proxmox Backup Server,
- la définition de pratiques de reprise (restauration, tests) adaptées aux objectifs du projet.

Dans cette DIN, la préparation se traduit par la mise en place d'un nœud Proxmox stable, accessible et correctement configuré (notamment réseau), afin de servir de base fiable aux intégrations futures (HA/cluster/backup).

ACTE VII — SÉCURITÉ

VII.1. Sécurité et gestion des accès

VII.1.1 Gestion des identifiants

Dans le cadre de ce projet réalisé en contexte pédagogique, la gestion des identifiants vise principalement à :

- garantir la reprise rapide par les membres de l'équipe projet,
- faciliter la vérification et la validation par l'encadrant.

Les identifiants nécessaires à l'administration du serveur (compte et mot de passe) peuvent donc être consignés dans la documentation projet, conformément aux consignes d'encadrement.

Néanmoins, afin de conserver une démarche structurée, les règles suivantes sont appliquées :

- les informations d'accès sont centralisées dans une section dédiée (et non dispersées dans le document),
- les conventions de nommage et de construction de mot de passe sont identifiées, pour éviter les incohérences,
- les accès sont limités aux personnes impliquées dans le projet.

VII.1.2 Accès administrateur

L'administration du serveur Proxmox est réalisée via :

- l'interface web Proxmox (accès management),
- le compte **root** (administrateur), utilisé pour la mise en place initiale et les opérations de configuration.

L'accès administrateur implique des privilèges complets sur l'hyperviseur (configuration réseau, stockage, création/gestion de VM/CT, paramètres système). À ce titre, l'accès est réservé aux membres autorisés de l'équipe projet.

VII.1.3 Limites de sécurité assumées

Le projet étant mené en environnement pédagogique, certaines mesures de sécurité "production" ne sont pas mises en place à ce stade ou ne sont pas prioritaires.

Les limites assumées incluent notamment :

- l’usage du compte root pour l’administration initiale (au lieu de comptes nominatif/roles séparés),
- la consignation des identifiants dans la documentation projet (selon validation de l’encadrant),
- l’absence possible de coffre-fort à secrets (type gestionnaire de mots de passe d’équipe).

Ces choix sont assumés et cadrés par le périmètre du projet. Ils pourront être réévalués lors d’une phase d’industrialisation ou de mise en conditions “production”, et/ou dans les livrables d’architecture et d’exploitation (DAT/DEX) si l’encadrement le demande.

ACTE VIII — TROUBLESHOOTING

VIII.1. Dépannage et vérifications

VIII.1.1 Problèmes potentiels identifiés

Dans un contexte d'installation et de mise en service d'un nœud Proxmox, plusieurs problèmes peuvent apparaître, principalement liés au réseau et à la résolution de noms.

Les problématiques les plus probables dans le cadre du projet sont :

- Accès interface web impossible (adresse IP non joignable, port non accessible, mauvais réseau).
- Erreur de résolution DNS (FQDN non résolu, DNS inadapté ou indisponible).
- Conflit d'adressage IP (IP déjà utilisée sur le réseau).
- Mauvaise interface réseau sélectionnée lors de l'installation (port non câblé, VLAN différent, débit non conforme).
- Incohérence de configuration locale (nom d'hôte/FQDN non aligné avec `/etc/hosts`).

Un point de vigilance spécifique est identifié sur l'environnement réseau du projet : des difficultés potentielles liées aux services **DNS/DHCP** (disponibilité, cohérence, configuration). Des solutions alternatives sont envisagées au niveau projet (poste dédié DNS/DHCP ou usage du routeur pour segmenter une DMZ), afin de garantir un socle réseau stable pour l'infrastructure.

VIII.1.2 Vérifications réseau

Les vérifications suivantes peuvent être réalisées après installation ou après modification de configuration :

- Vérification de l'adresse IP et du masque configurés sur l'interface de management.
- Vérification de l'état du lien (interface UP, câble, port switch).
- Test de connectivité vers la passerelle.
- Test de résolution DNS (nom d'hôte/FQDN et domaines nécessaires).
- Vérification de l'accès à l'interface web Proxmox depuis un poste client.

Ces vérifications permettent d'identifier rapidement si le problème est :

- local au serveur (mauvaise config),
- lié au réseau (câblage, VLAN),
- lié aux services (DNS/DHCP).

VIII.1.3 Commandes de diagnostic

Les commandes suivantes peuvent être utilisées pour diagnostiquer les problèmes courants :

- **Vérification des interfaces et adresses** :
  - `ip a` (ou `ifconfig -a` si disponible)
- **Vérification de la route par défaut** :
  - `ip route`
- **Test de connectivité** :
  - `ping <IP>` (passerelle, DNS, poste client selon le cas)
- **Test de résolution DNS** :
  - `nslookup <nom>` / `dig <nom>` (si disponible)
- **Vérification du nom d'hôte** :
  - `hostname` / `hostname -f`
- **Vérification de la configuration réseau** :
  - `cat /etc/network/interfaces`
  - `cat /etc/hosts`

VIII.1.4 Actions correctives possibles

- Selon le diagnostic, plusieurs actions correctives peuvent être appliquées :
- **Correction de l'IP / masque / passerelle** sur l'interface de management :
  - modification de `/etc/network/interfaces`, puis redémarrage réseau ou redémarrage du serveur.
- **Correction du nom d'hôte et du FQDN** :
  - alignement entre `hostname`, `/etc/hosts` et les paramètres réseau.
- **Correction du DNS** :
  - modification des serveurs DNS configurés, ou mise en place d'une solution DNS locale temporaire si le DNS du réseau est indisponible.
- **Changement d'interface réseau** :
  - si l'interface sélectionnée n'est pas la bonne (port non câblé / VLAN différent), reconfigurer l'interface de management.
- **Redémarrage contrôlé** :
  - lorsque la configuration a été modifiée, un redémarrage peut être effectué afin de garantir l'application complète des paramètres.

L’ensemble de ces étapes vise à rétablir un accès stable au nœud Proxmox et à garantir la cohérence des services réseau indispensables à la suite du projet (cluster, stockage partagé, sauvegardes, services DevOps).

ACTE IX — CONCLUSION

IX.1. Conclusion

IX.1.1 Bilan de l'installation

Cette DIN a permis de formaliser l'ensemble des étapes nécessaires à la mise en service d'un nœud Proxmox VE 9.1 dans le cadre du projet d'infrastructure mené à l'IUT.
L'installation a été réalisée selon une procédure standardisée (mode graphique), suivie d'une configuration initiale cohérente :

- paramétrage des accès administrateur et de l'adresse mail d'alerte,
- configuration du réseau de management (interface, IP, passerelle, DNS, FQDN),
- vérifications et ajustements post-installation afin d'assurer un accès stable à l'interface d'administration.

Le serveur est désormais opérationnel sur les fondamentaux (accès, réseau, base de virtualisation) et peut servir de socle pour les intégrations prévues dans la suite du projet.

IX.1.2 Apports pédagogiques

La mise en place de ce serveur Proxmox constitue un support concret d'apprentissage et de validation de compétences, notamment sur :

- la compréhension d'une installation d'hyperviseur et de ses paramètres critiques,
- la configuration réseau et la cohérence des services associés (résolution DNS, adressage, identification par FQDN),
- la démarche de documentation technique (traçabilité, reproductibilité, structuration d'une DIN),
- l'approche "infrastructure projet" : préparer une brique technique à être intégrée dans un ensemble plus large.

Ces apports s'inscrivent directement dans les objectifs du projet DevOps pédagogique, en rapprochant les pratiques mises en œuvre de situations professionnelles.

IX.1.3 Prochaines étapes du projet

Une fois le nœud Proxmox installé et stabilisé, les prochaines étapes du projet consistent à intégrer progressivement cette brique dans l'infrastructure cible, notamment :

- ❖ l’intégration des différents serveurs dans un cluster Proxmox multi-nœuds (3 serveurs) afin d’expérimenter la gestion centralisée et la bascule,
- ❖ l’exploitation du stockage centralisé via la baie SAN comme support des données et potentiellement des VM,
- ❖ la mise en place d’une solution de sauvegarde (ex. Proxmox Backup Server) et la définition d’une stratégie de restauration,
- ❖ le déploiement des services liés au projet (serveur Git, services web) et des mécanismes d’automatisation (scripts, Ansible/Terraform) selon le périmètre défini.

Ces éléments feront l’objet des livrables associés (DAT/DEX) et/ou de sections dédiées en fonction de l’avancement du projet et des consignes d’encadrement.

Gitlab Propre

DIN — Serveur Proxmox

Sommaire

ACTE I — CONTEXTE ET OBJECTIFS

I.1. Introduction

I.1.1 Objectif de la DIN GitLab

Ce DIN (Document d'Installation) a pour objectif de décrire de manière claire, structurée et reproductible l'installation et la configuration initiale de GitLab Community Edition (CE) sur une machine virtuelle Debian déployée dans l'infrastructure du projet.

Elle formalise les paramètres retenus, les étapes réalisées et les validations effectuées afin de :

- garantir la traçabilité des choix techniques,
- permettre la reproductibilité du déploiement sur une VM équivalente,
- faciliter la validation par l'équipe projet et l'encadrement,
- fournir une base de référence pour les phases ultérieures (exploitation, automatisation, intégrations).

Dans le cadre pédagogique du projet, certaines informations d'accès (identifiants/mots de passe) peuvent être consignées dans la DIN afin de faciliter la reprise et la vérification par l'encadrant.

I.1.2 Contexte du projet DevOps pédagogique

GitLab s'inscrit dans un projet pédagogique d'infrastructure DevOps visant à mettre en place un environnement complet de gestion de projet de développement, incluant notamment :

- une plateforme de virtualisation (Proxmox) permettant l'hébergement de services,
- un serveur de gestion de versions et de collaboration,
- des composants web et d'automatisation prévus dans les phases suivantes.

Dans ce contexte, GitLab est déployé sur une **machine virtuelle Debian 13 (Trixie)** afin de fournir un service centralisé de dépôts et de gestion de projet, utilisable par l'équipe dans la suite des travaux.

I.1.3 Rôle de GitLab dans l'architecture globale

GitLab CE constitue la brique de gestion de code et de collaboration du projet. Son rôle est de :

- héberger les dépôts Git utilisés par l'équipe,
- centraliser la gestion des projets (organisation, dépôts, utilisateurs),
- fournir un point d'intégration pour les étapes ultérieures (ex. automatisation, déploiements, CI/CD si mis en place).

Dans la phase couverte par cette DIN, l'objectif principal est de disposer d'une instance GitLab fonctionnelle, accessible et sécurisée au niveau nécessaire (pare-feu, accès SSH/HTTP/HTTPS) afin de supporter les usages de base (accès web, récupération/synchronisation de dépôts).

I.1.4 Périmètre du DIN

Le périmètre de cette DIN couvre :

- la création et le dimensionnement de la VM GitLab (ressources CPU/RAM/stockage),
- l'installation du système Debian 13 (Trixie) et la résolution d'un incident initial lié aux dépôts,
- l'installation des dépendances nécessaires (gnupg, openssh-server, ufw, git, curl),
- l'installation de GitLab CE via le dépôt officiel,
- la configuration du pare-feu UFW (ports 22, 80, 443),
- la configuration initiale côté GitLab (création d'un utilisateur),
- une validation fonctionnelle de base (ex. synchronisation du dépôt via Git).

Les captures d'écran associées à la DIN sont utilisées pour illustrer les étapes clés et les validations (services GitLab, interface web, utilisateur, récupération de projet).

I.1.5 Éléments hors périmètre (CI/CD avancé, runners, industrialisation)

Ne font pas partie du périmètre de cette DIN, car relevant de phases ultérieures, de choix d'architecture (DAT) ou d'exploitation (DEX) :

- la mise en place avancée de CI/CD (pipelines complets, templates, registres, politiques),
- le déploiement et la gestion de GitLab Runners (installations, configuration, exécution),
- l'industrialisation (reverse proxy, certificats/PKI interne, haute disponibilité),
- les stratégies complètes de sauvegarde/restauration GitLab,

- ❖ l’intégration avancée avec l’automatisation (Ansible/Terraform) et les applications de provisioning.

Ces éléments pourront être abordés dans des sections dédiées selon l’avancement du projet et les consignes d’encadrement.

ACTE II — ARCHITECTURE DE LA VM GITLAB

II.1 Architecture de la machine virtuelle

II.1.1 Création de la VM sur Proxmox

GitLab CE est déployé sur une machine virtuelle hébergée sur l'hyperviseur Proxmox VE 9.1 (infrastructure de virtualisation du projet).

La VM est installée avec le système d'exploitation Debian 13 (Trixie), retenu pour sa stabilité, son approche "serveur" et sa compatibilité avec les dépendances nécessaires à GitLab.

II.1.2 Ressources allouées

La machine virtuelle GitLab a été dimensionnée avec les ressources suivantes :

- **Processeur** : **4 vCPU**
- **Mémoire vive** : **4 Go de RAM**
- **Stockage** : **40 Go**

Ces paramètres constituent un compromis adapté au contexte pédagogique : suffisamment de ressources pour exécuter GitLab CE et ses services associés, tout en restant cohérents avec un environnement mutualisé.

II.1.3 Compte système configuré

Un compte système a été configuré sur la VM :

- **Utilisateur** : `srv-git`

Rôle du compte :

- servir de compte opérationnel pour l'administration courante sur la VM (connexion, gestion de fichiers, exécution de commandes),
- permettre la réalisation des actions liées au projet (récupération/synchronisation de dépôts, tests, validations).

(Les opérations nécessitant des privilèges élevés sont réalisées via élévation de droits, selon les besoins d'installation et de configuration.)

II.1.4 Justification du dimensionnement

Le dimensionnement retenu est cohérent avec les objectifs et contraintes du projet :

- Adéquation aux besoins pédagogiques : héberger un service Git centralisé accessible par l'équipe, avec une interface web et une administration à distance.

- Capacité à héberger GitLab CE : GitLab CE s'appuie sur plusieurs services (web, base de données, workers) ; l'allocation de 4 vCPU / 4 Go RAM / 40 Go permet un fonctionnement stable pour un usage de laboratoire (projets, tests, synchronisation).

Ce dimensionnement pourra être ajusté si la charge augmente (multiplication des projets, activation de CI/CD, ajout de runners, stockage de conteneurs, etc.), éléments toutefois hors périmètre de la présente DIN.

ACTE III — INSTALLATION DU SYSTÈME (DEBIAN)

III.1 Installation de Debian 13 (Trixie)

III.1.1 Version choisie

Le système d'exploitation installé sur la machine virtuelle dédiée à GitLab est Debian 13 (Trixie).

Ce choix s'inscrit dans une logique "serveur" adaptée au projet : distribution Linux stable, largement utilisée en environnement d'infrastructure, et compatible avec les dépendances nécessaires à l'installation de GitLab CE.

III.1.2 Problème rencontré lors de la mise à jour

Lors de la phase initiale de mise en service, un incident a été rencontré lors de la mise à jour du système :

- erreur de type : "Release file not found" lors de l'accès aux dépôts.

Impact sur l'installation :

- impossibilité de réaliser correctement les mises à jour (apt update/ upgrade),
- blocage potentiel de l'installation ou de la récupération de paquets nécessaires aux étapes suivantes (dépendances, dépôts GitLab).

Cet incident a été traité comme un point bloquant, car la disponibilité et la cohérence des dépôts conditionnent la fiabilité de l'environnement logiciel.

III.1.3 Action corrective

Afin de rétablir un environnement sain et reproductible, l'action corrective suivante a été appliquée :

- Recréation de la machine virtuelle (nouvelle VM),
- Nouvelle installation de Debian 13 (Trixie),
- Validation du bon fonctionnement des dépôts avant de poursuivre (mise à jour et installation de paquets possibles).

Cette démarche permet de valoriser l'incident dans une logique professionnelle : identification d'un dysfonctionnement bloquant, application d'une correction adaptée au

contexte (réinstallation propre), puis validation technique avant de passer à l’installation de GitLab.

ACTE IV — INSTALLATION DE GITLAB CE

IV.1 Préparation des dépendances

IV.1.1 Installation des dépendances

Avant l'installation de GitLab CE, les paquets nécessaires au bon déroulement de l'installation et au fonctionnement du service sont installés sur la VM Debian.

Les dépendances retenues :

- **gnupp :** gestion des clés GPG (signature et validation des dépôts),
- **openssh-server :** accès distant SSH (administration et intéractions Git via SSH si nécessaire),
- **ufw :** pare-feu simple pour filtrer les accès réseau,
- **git :** outil de gestion de versions pour les manipulations des dépôts,
- **curl :** récupération de ressources externes (notamment le script d'ajout du dépôt GitLab).

Une fois ces paquets installés, l'environnement est prêt pour l'ajout du dépôt officiel GitLab CE.

IV.1.2 Ajout du dépôt GitLab CE

L'installation de GitLab CE s'appuie sur le dépôt officiel GitLab. Le dépôt est ajouté à la VM en utilisant curl, afin de :

- récupérer le script officiel d'installation du dépôt,
- configurer le dépôt GitLab CE dans le gestionnaire de paquets,
- permettre l'installation et les mises à jour via les mécanismes standards (APT).

Cette étape garantit que Gitlab CE est installé à partir d'une source officielle et maintenable.

IV.2 Installation du GitLab CE

IV.2.1 Installation via gestionnaire de paquets

Une fois le dépôt GitLab CE configuré, GitLab CE est installé via le gestionnaire de paquets du système.
Cette installation déploie les composants nécessaires au fonctionnement de GitLab (services Web et services internes associés).

IV.2.2 Initialisation de GitLab

Après installation, GitLab est initialisé afin de rendre l'instance opérationnelle :

- génération des configurations,
- démarrage des services,
- préparation de l'interface web.

IV.2.3 Vérification des services

Le bon fonctionnement de l'installation est validé par une vérification de l'état des services GitLab via la commande

- **`gitlab-ctl status`**

Cette commande permet de confirmer que les services essentiels sont démarrés et que l'instance est dans un état cohérent avant de poursuivre la configuration (pare-feu, accès web, utilisateurs).

```
gitlab Reconfigured!
root@debian:/home/srv-git# gitlab-ctl status
run: alertmanager: (pid 18437) 171s; run: log: (pid 18234) 221s
run: gitaly: (pid 18378) 174s; run: log: (pid 17515) 388s
run: gitlab-exporter: (pid 18405) 173s; run: log: (pid 18113) 238s
run: gitlab-kas: (pid 17828) 372s; run: log: (pid 17847) 370s
run: gitlab-workhorse: (pid 18350) 175s; run: log: (pid 18000) 255s
run: logrotate: (pid 17380) 401s; run: log: (pid 17403) 400s
run: nginx: (pid 18368) 174s; run: log: (pid 18019) 249s
run: node-exporter: (pid 18391) 174s; run: log: (pid 18065) 244s
run: postgres-exporter: (pid 18447) 170s; run: log: (pid 18294) 215s
run: postgresql: (pid 17561) 378s; run: log: (pid 17571) 377s
run: prometheus: (pid 18418) 172s; run: log: (pid 18178) 227s
run: puma: (pid 17900) 271s; run: log: (pid 17908) 268s
run: redis: (pid 17426) 395s; run: log: (pid 17445) 393s
run: redis-exporter: (pid 18408) 173s; run: log: (pid 18157) 233s
run: sidekiq: (pid 17918) 265s; run: log: (pid 17929) 264s
root@debian:/home/srv-git#
```

ACTE V — CONFIGURATION RÉSEAU ET SÉCURITÉ

V.1 Configuration du pare-feu

IV.1.1 Activation de UFW

Le pare-feu UFW (Uncomplicated Firewall) est utilisé afin de contrôler les flux entrant vers la VM GitLab.

Après l'installation du paquet, UFW est activé afin de mettre en place une politique de filtrage simple et lisible, adaptée au contexte du projet.

VI.1.2 Ouverture des ports nécessaires

Les ports indispensables au fonctionnement et à l'administration de GitLab sont autorisés :

- SSH (22) : administration distante de la VM et accès Git via SSH si utilisé,
- HTTP (80) : accès à l'interface web GitLab en HTTP,
- HTTPS (443) : accès à l'interface web GitLab en HTTPS.

Ces ouvertures garantissent que l'instance GitLab est accessible depuis le réseau de management / le réseau projet, tout en évitant d'exposer des services inutiles.

VI.1.3 Justification des règles

Les règles UFW appliqués répondent aux besoins suivants :

- Administration distante : permettre la connexion SSH pour la gestion du système (maintenance, diagnostics, mise à jour).
- Accès interfaces web : permettre l'accès à l'utilisateur à GitLab via navigateur (HTTP/HTTPS).

Le filtrage vie UFW contribue à limiter la surface d'exposition réseau de la VM en n'autorisant que les flux nécessaires.

VI.2 Accès à l'interface GitLab

VI.2.1 URL d'accès

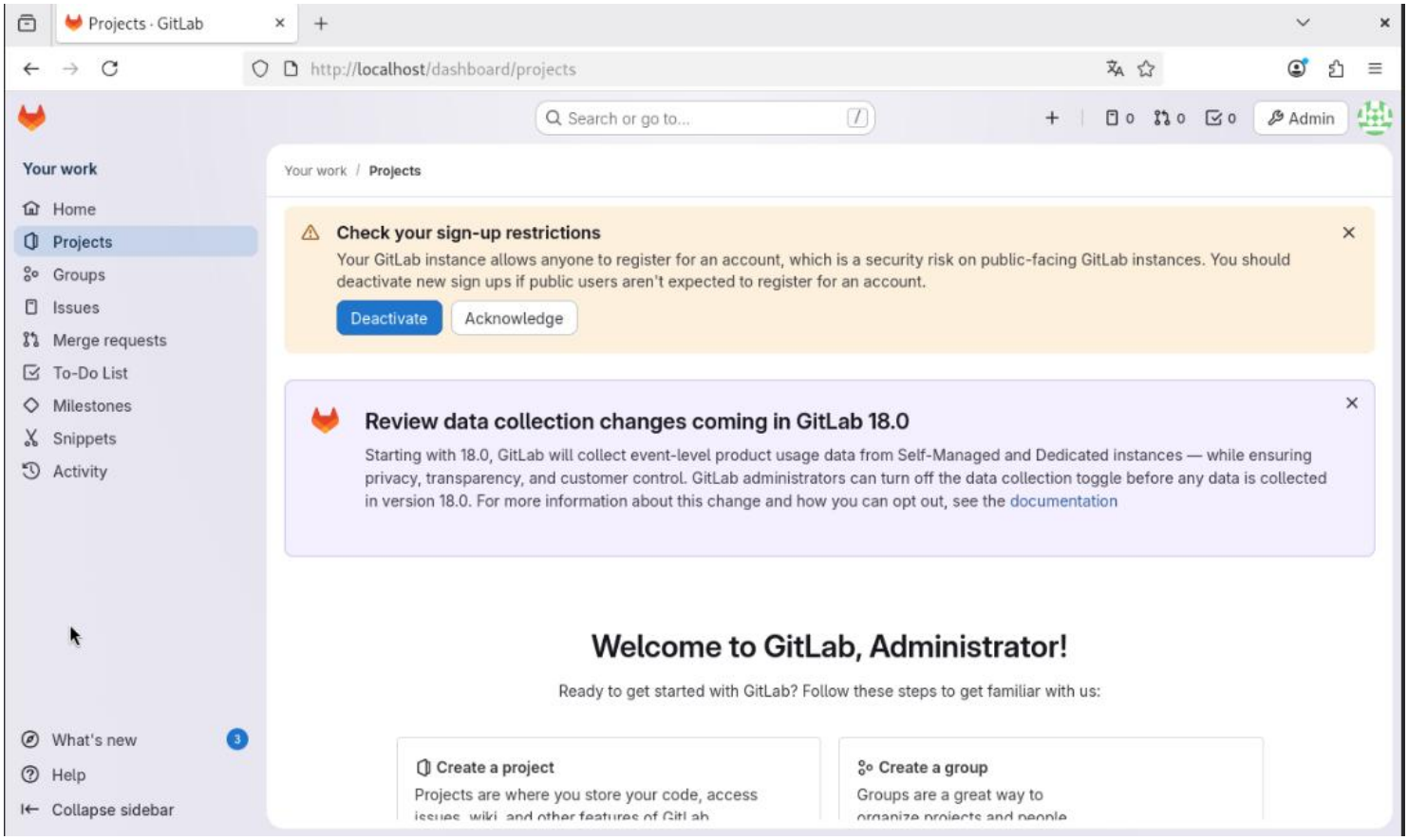
Une fois GitLab installé et le pare-feu configuré, l'accès à la plateforme se fait via l'interface web.

L'URL d'accès correspond à l'adresse IP de la VM.

VI.2.2 Vérification de fonctionnement

La mise en service est validé par :

- l'affichage de la page GitLab dans un navigateur,
- la confirmation que l'instance répond correctement (interface accessible, chargement complet),
- la possibilité de procéder à l'authentification (validation détaillée dans l'acte suivant).



ACTE VI — CONFIGURATION APPLICATIVE GITLAB

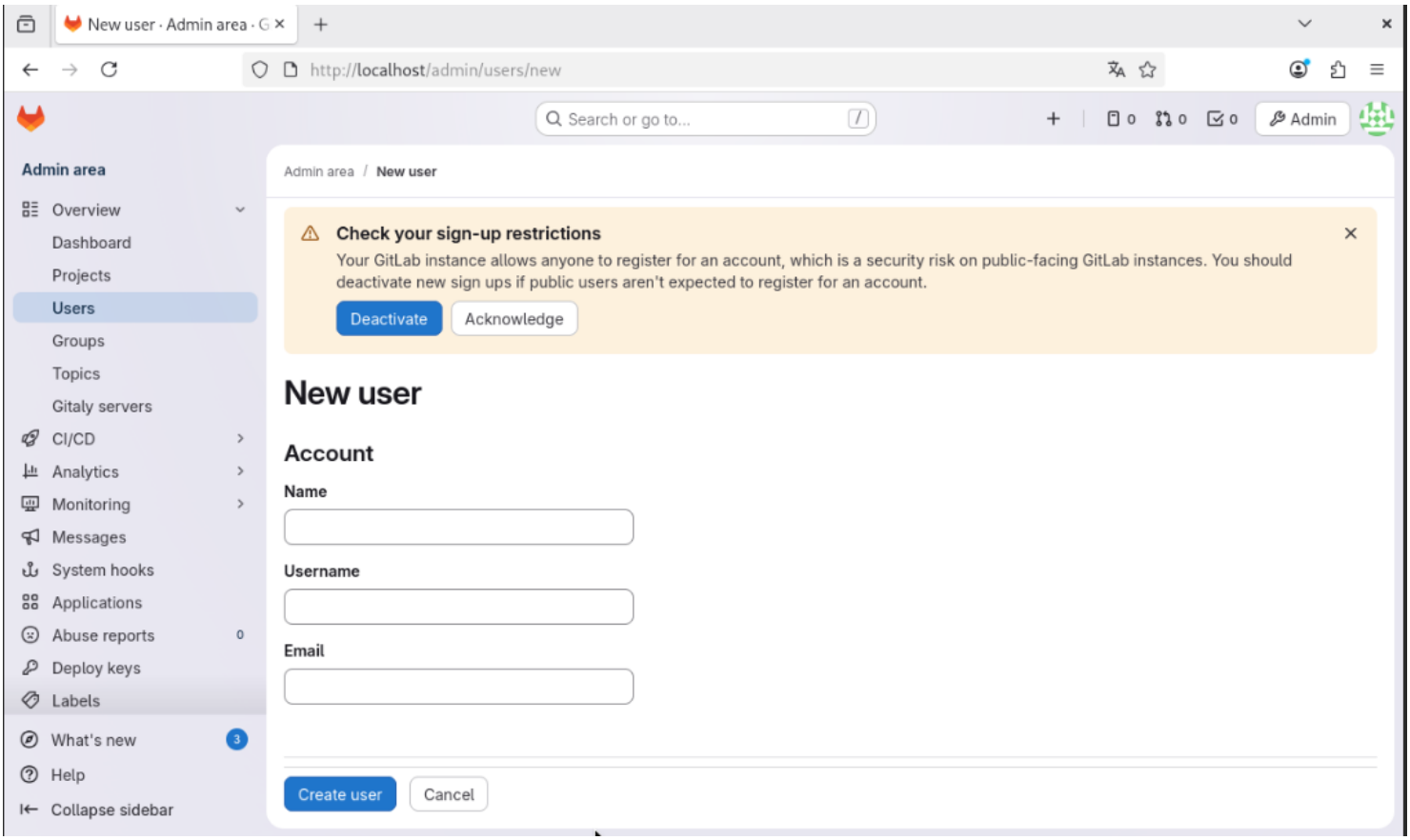
VI.1. Création d'un utilisateur GitLab

VI.1.1 Paramètres utilisateur

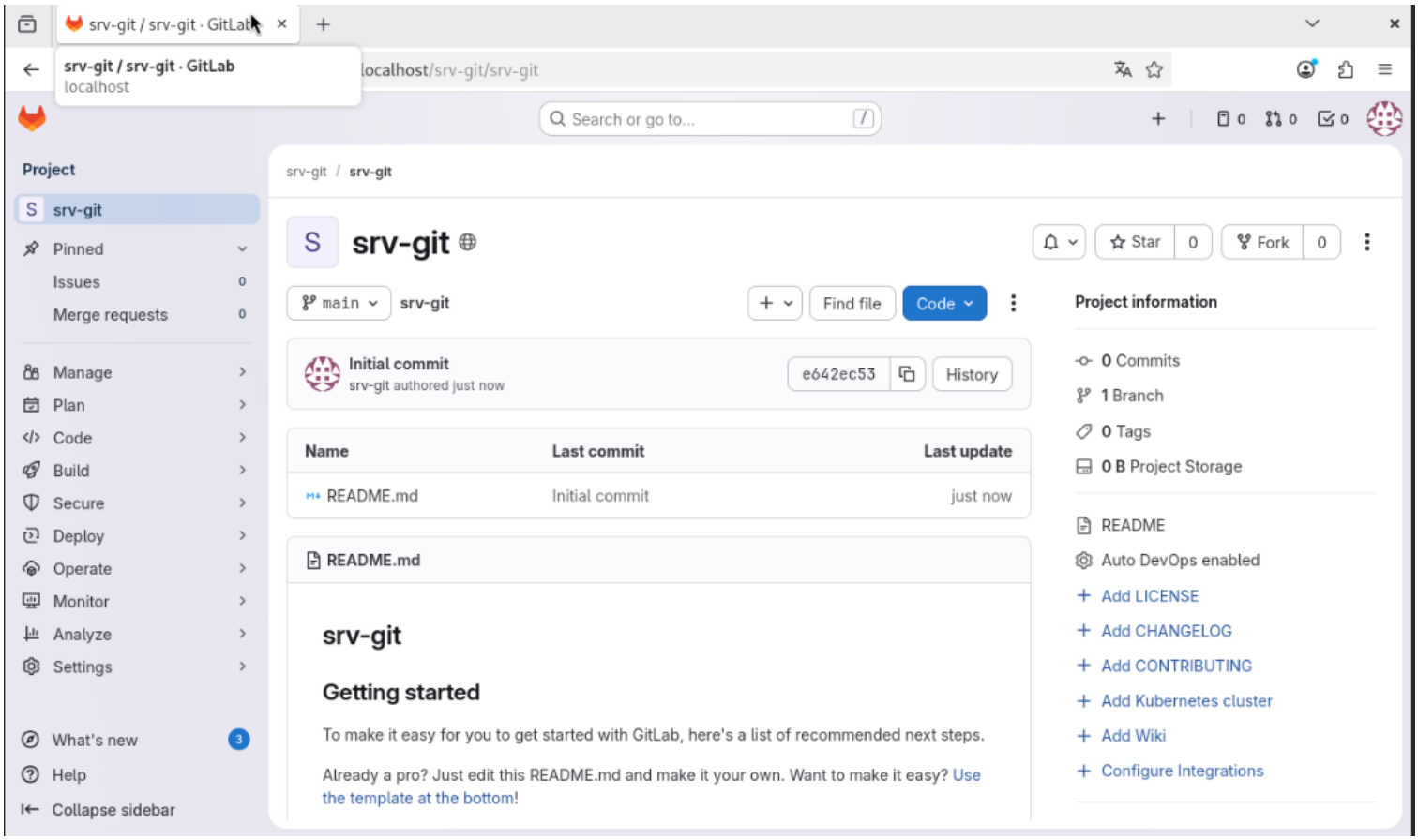
Après validation de l'accès à l'interface web, un utilisateur GitLab est créé afin de permettre l'utilisation de la plateforme dans le cadre du projet.

Les paramètres retenus sont :

- **Nom d'utilisateur** : `srv-git`



- **Mot de passe** : root1234



VI.1.2 Rôle de cet utilisateur

L'utilisateur `srv-git` est utilisé pour :

- la gestion des dépôts (accès, récupération, synchronisation des projets),
- l'administration initiale de la plateforme (premiers paramétrages, validations fonctionnelles).

Dans le cadre pédagogique, cet utilisateur constitue un point d'entrée simple et reproductible pour les tests et la mise en place des usages de base.

VI.2. Validation fonctionnelle

VI.2.1 Accès à la plateforme

La validation fonctionnelle débute par la confirmation que :

- l'interface web est accessible,
- l'authentification avec l'utilisateur créé est possible,

- la navigation de base (tableau de bord, projets) est opérationnelle.

VI.2.2 Création / visualisation d'un projet

Une validation complémentaire consiste à vérifier la capacité à manipuler un projet GitLab, notamment :

- création d'un projet (ou accès à un projet existant),
- visualisation du dépôt via l'interface GitLab,
- vérification que les opérations Git (clone/pull/push selon autorisations) sont possibles.

Cette étape confirme que l'instance GitLab CE est correctement installée et utilisable pour les besoins du projet DevOps pédagogique.

ACTE VII — INTÉGRATION AU PROJET DEVOPS

VII.1. Utilisation de GitLab dans le projet

> **Principe** : cette section décrit l'intention d'intégration de GitLab dans le projet DevOps pédagogique (usage attendu et place fonctionnelle). Elle ne détaille pas l'industrialisation complète (CI/CD avancé, runners, politiques, reverse proxy, etc.), qui relève des phases ultérieures et/ou des livrables associés.

VII.1.1 Hébergement des dépôts

GitLab CE est utilisé comme plateforme centralisée d'hébergement des dépôts Git du projet. Il permet à l'équipe de :

- versionner le code et les fichiers de configuration,
- centraliser les dépôts dans un point unique,
- faciliter la collaboration (accès au dépôt, visibilité du projet, suivi).

Dans le cadre du projet, GitLab constitue donc le référentiel principal pour les développements et la mise à disposition des sources.

VII.1.2 Synchronisation via git pull

Une fois le dépôt disponible sur GitLab, la synchronisation côté VM/clients est réalisée via les commandes Git.

La récupération du projet a été validée par l'exécution d'un `git pull`, permettant de :

- vérifier l'accès au dépôt,
- confirmer la bonne configuration des droits/accès,
- s'assurer que le contenu du projet est correctement récupérable depuis l'instance GitLab.

```
root@debian:/home/srv-git/srv-git# git branch
* main
root@debian:/home/srv-git/srv-git#
```

VII.1.3 Lien avec les futures automatisations (Ansible / Terraform / WebApp VM)

Dans la continuité du projet, GitLab a vocation à devenir le point d'ancrage des éléments d'automatisation et de déploiement, notamment :

- dépôt des playbooks **Ansible** et/ou configurations **Terraform**,
- versionnement des scripts/outils utilisés pour le provisioning,
- stockage et évolution du projet web (ex. WebApp de création/provisionnement de VM).

L'objectif est de garantir une traçabilité et une collaboration efficaces sur ces composants, en s'appuyant sur GitLab comme référentiel central.

À ce stade, la DIN confirme l'intégration fonctionnelle minimale (hébergement + synchronisation de dépôt) et positionne GitLab comme brique essentielle pour les étapes d'automatisation à venir.

ACTE VIII — TROUBLESHOOTING

VIII.1. Problèmes potentiels identifiés

Dans le cadre du déploiement de GitLab CE sur une VM Debian, les problèmes les plus fréquents concernent la disponibilité des dépôts système, l'état des services GitLab et l'accessibilité réseau.

Les problèmes potentiels identifiés sont notamment :

- **Erreur dépôt Debian** : erreurs lors de `apt update` (ex. dépôt non joignable, incohérence de version, "Release file not found").
- **Service GitLab non démarré** : GitLab installé mais services internes arrêtés ou en erreur.
- **Ports non ouverts** : pare-feu non configuré ou règles manquantes empêchant l'accès SSH/HTTP/HTTPS.
- **DNS incorrect** : nom d'hôte/FQDN non résolu, DNS indisponible, ce qui peut impacter l'accès et certaines configurations.

VIII.2. Commandes de diagnostic

Pour diagnostiquer rapidement les incidents, les commandes suivantes peuvent être utilisées :

- **État des services GitLab** :
  - `gitlab-ctl status`
- **État d'un service système** (selon besoin) :
  - `systemctl status <service>`
- **État du pare-feu et des règles** :
  - `ufw status`
- **Vérification des ports en écoute** :
  - `ss -tulpn`

Ces commandes permettent d'identifier si le problème est lié :

- au système (dépôts, services),
- à GitLab (services internes),
- au réseau/sécurité (ports, filtrage).

VIII.3. Actions correctives possibles

Selon la cause identifiée, les actions correctives suivantes peuvent être appliquées :

- **Reconfiguration dépôt** :
  - vérification/correction des sources APT (fichiers de dépôts),
  - relance des mises à jour (`apt update`),
  - validation de la cohérence de version.
- **Réinstallation propre** :
  - en cas d'environnement instable ou non reproductible, repartir d'une VM saine (réinstallation Debian), puis réappliquer la procédure d'installation GitLab.
- **Correction pare-feu** :
  - activation de UFW si nécessaire,
  - ouverture des ports requis (22/80/443),
  - vérification que les règles correspondent au périmètre réseau.
- **Redémarrage des services** :
  - redémarrage contrôlé des services GitLab (et/ou services système concernés),
  - vérification immédiate via `gitlab-ctl status`.

L'objectif de ces actions est de rétablir un accès stable à l'instance GitLab et de garantir un environnement exploitable pour les besoins du projet (hébergement de dépôts, synchronisation, et intégrations ultérieures).

ACTE IX — CONCLUSION

IX.1. Bilan

À l'issue des étapes décrites dans cette DIN, l'instance GitLab Community Edition (CE) est considérée comme opérationnelle dans le cadre du projet.

Les points suivants ont été validés :

- **GitLab CE opérationnel** : services démarrés et état cohérent (validation via `gitlab-ctl status`).
- **VM fonctionnelle** : système Debian stable et environnement prêt pour l'exploitation.
- **Accès distant validé** : accès SSH pour l'administration et accès web via HTTP/HTTPS (selon règles pare-feu mises en place).

Ce socle permet d'utiliser GitLab comme référentiel central pour les dépôts du projet.

IX.2. Apports pédagogiques

Le déploiement de GitLab CE a permis de travailler et valider des compétences directement transférables en contexte professionnel, notamment :

- **Installation d'un service "lourd"** : compréhension d'une solution multi-composants (services internes, dépendances, initialisation).
- **Sécurisation minimale** : mise en place d'un filtrage réseau simple via UFW, ouverture des ports strictement nécessaires.
- **Intégration dans une infrastructure virtualisée** : déploiement et exploitation d'un service applicatif au sein d'une VM hébergée sur Proxmox.

L'incident rencontré sur les dépôts Debian a également permis d'appliquer une démarche de résolution structurée (identification, correction, validation).

IX.3. Prochaines étapes

Une fois GitLab CE en place, les évolutions possibles (selon périmètre et consignes d'encadrement) incluent :

- **Intégration CI/CD** : mise en place de pipelines et de bonnes pratiques de build/test/déploiement.

- **Ajout de runners** : déploiement et configuration de GitLab Runners pour exécuter les jobs CI.
- **Automatisation** : versionnement et exploitation d'outils IaC (Terraform) et des scripts de provisioning.
- **Sauvegardes** : définition d'une stratégie de sauvegarde/restauration GitLab (configuration, dépôts, données), en cohérence avec l'infrastructure globale.

Ces étapes relèvent des phases ultérieures du projet et pourront être détaillées dans les livrables associés (DAT/DEX) ou dans une DIN complémentaire si nécessaire.

Proxmox Backup Serveur

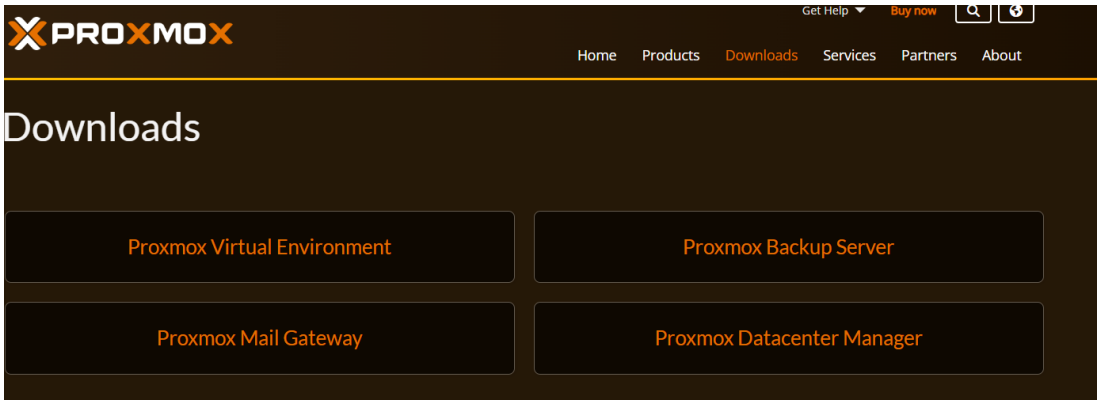
Documentation d’Installation de Proxmox Backup Server

⚠️ WORK IN PROGRESS - WIP

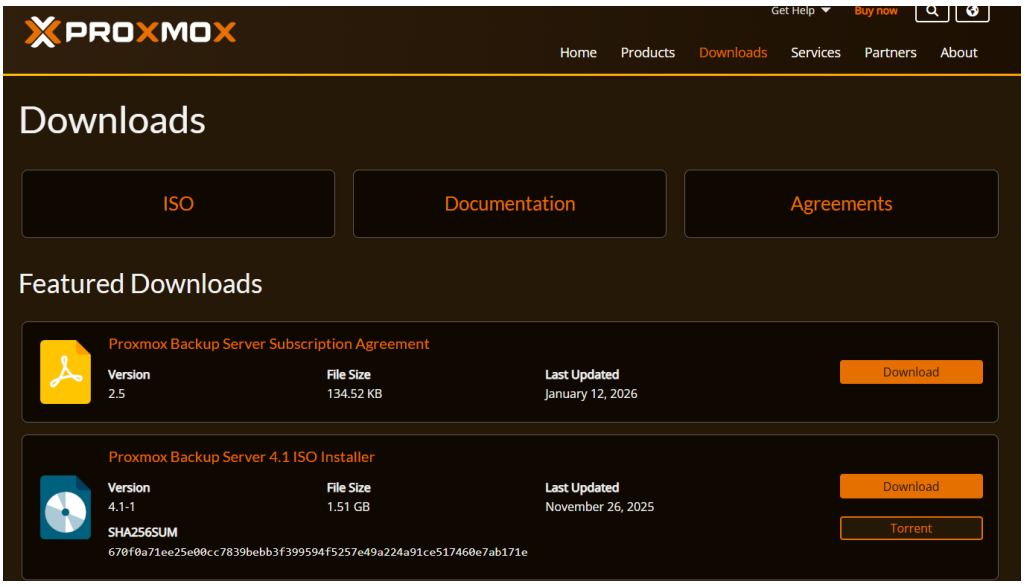
Dans cette documentation, on va voir comment installer un Proxmox Backup Server (PBS) sur un ordinateur dédié afin de réaliser une solution de sauvegarde pour notre réseau Proxmox.

Recherche de PBS

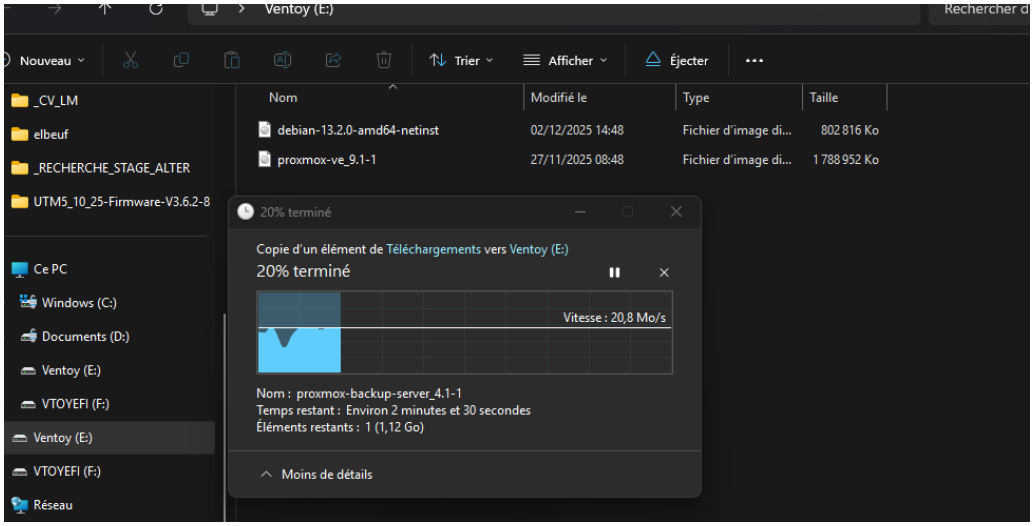
Dans un premier temps, on récupérer la l’ISO la plus récente de notre PBS.
Site : https://proxmox.com/en/downloads?utm_source



On prendra donc la version 4.1.1.



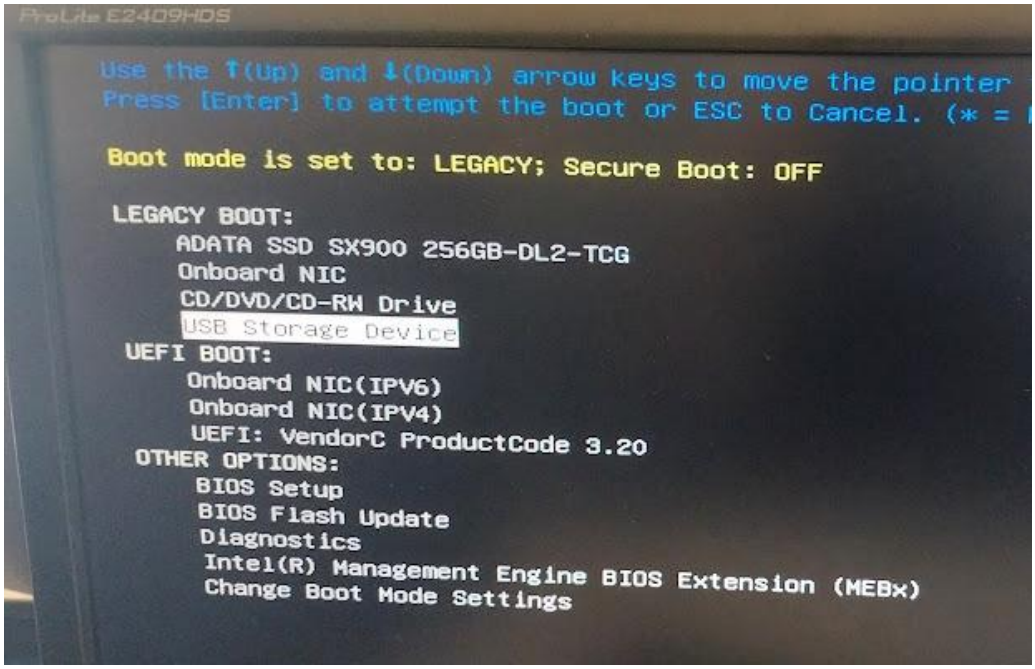
Après l’avoir télécharger, nous allions l’ajouter sur la clé usb utiliser pour installer nos différents iso depuis le début du projet.



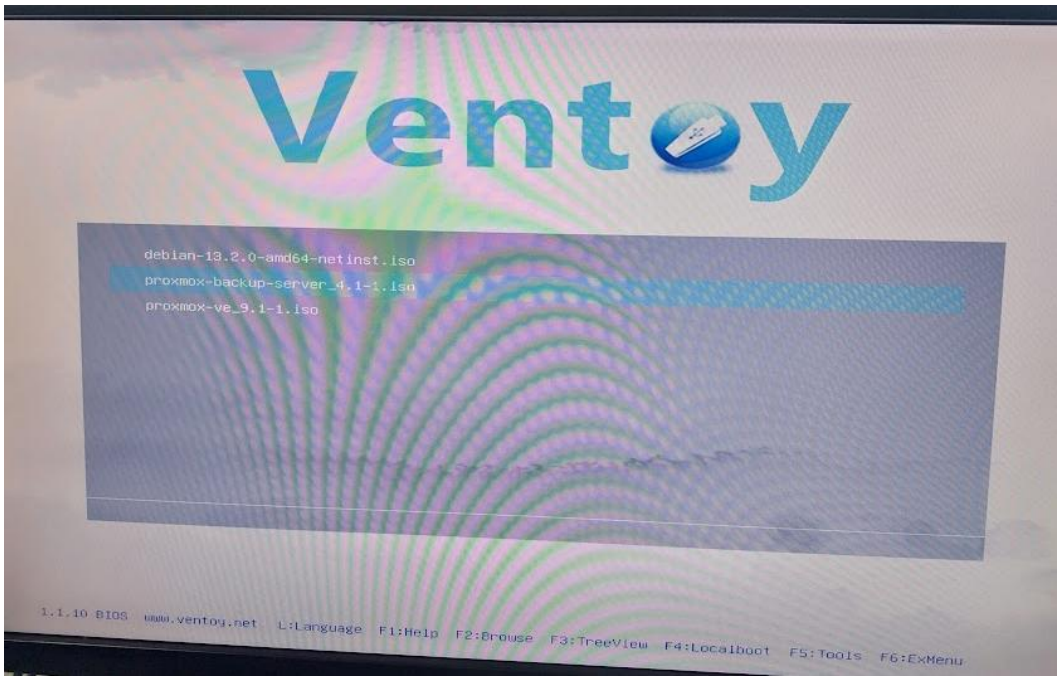
Installation de PBS :

Maintenant que nous avions l'OS sur la clé usb, nous pouvions intaller PBS.

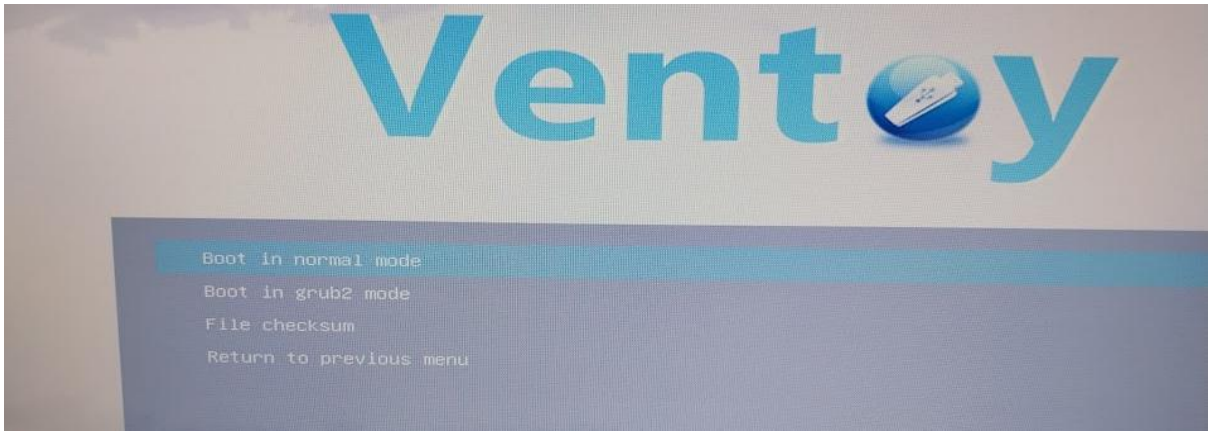
Lors de l'accès au bios avec la touche F12. Nous tombon ssur cette interface. On cherchera USB Storage Device pour avoir accès à notre clé USB.

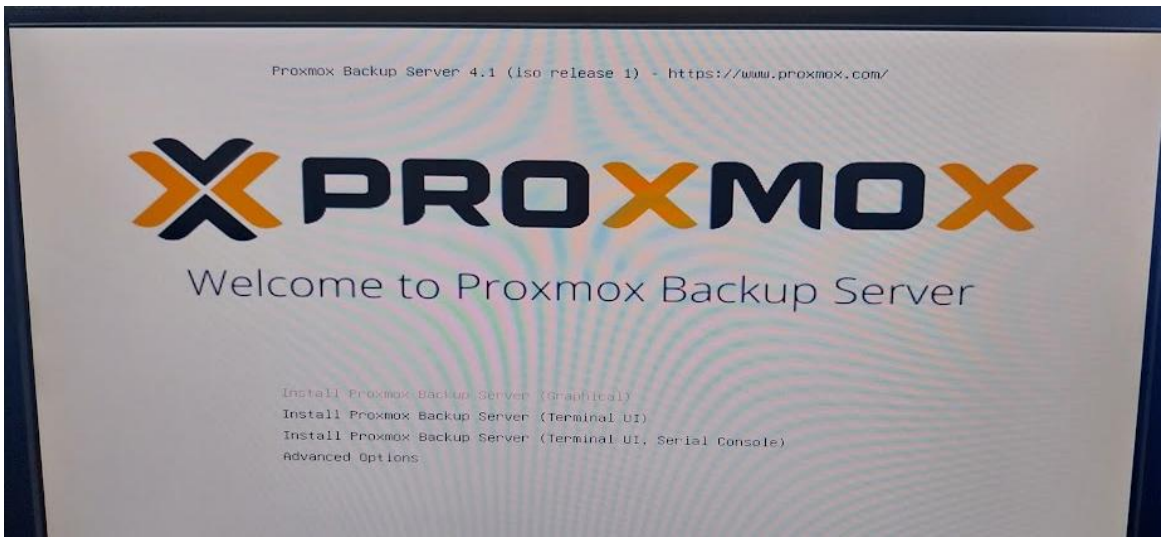


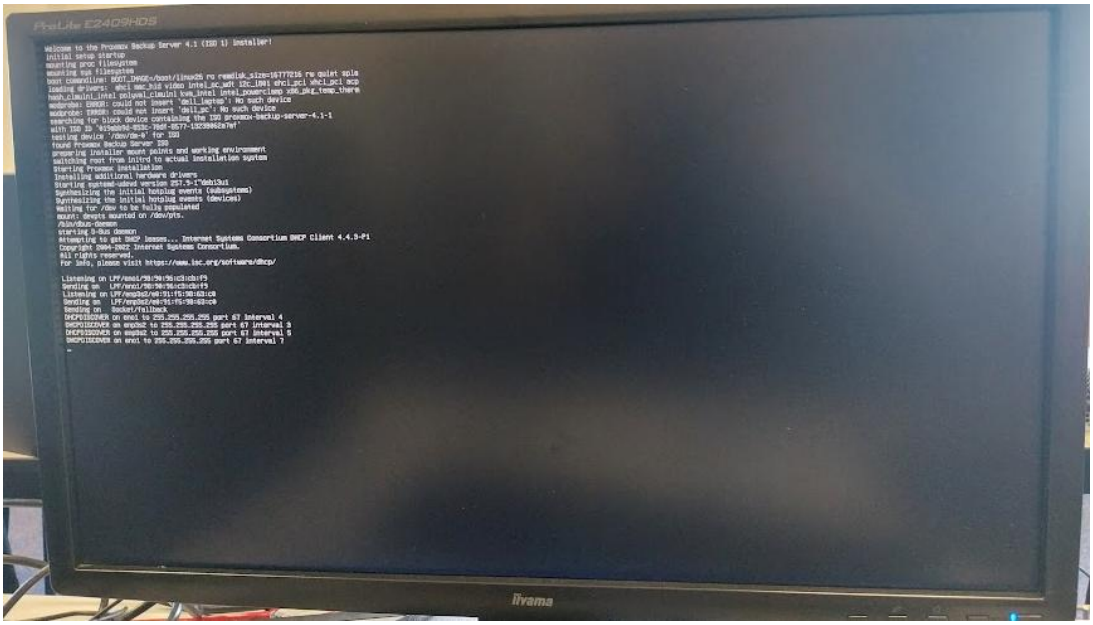
Par la suite nous arrivons à l'interface du choix des ISO de Ventoy. C'est le logiciel qui permet de transfromer la clé usb en clé bootable.

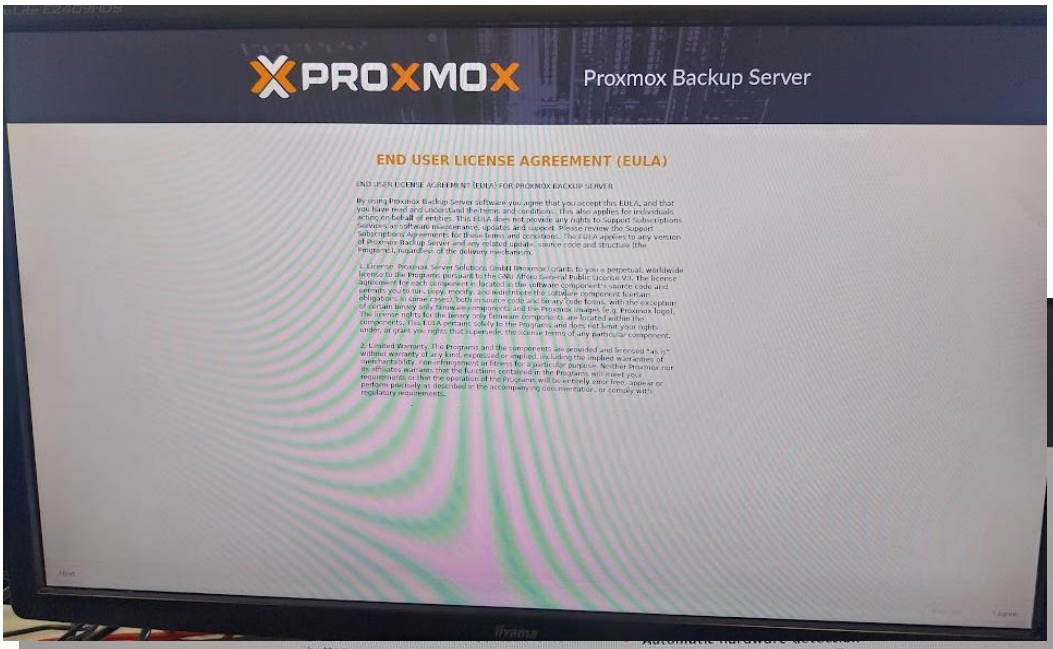


Nous choisions bien sûr notre ISO PBS pour l'installer.
Enfin de commencer notre installation, nous choisisions le "boot in normal mode".

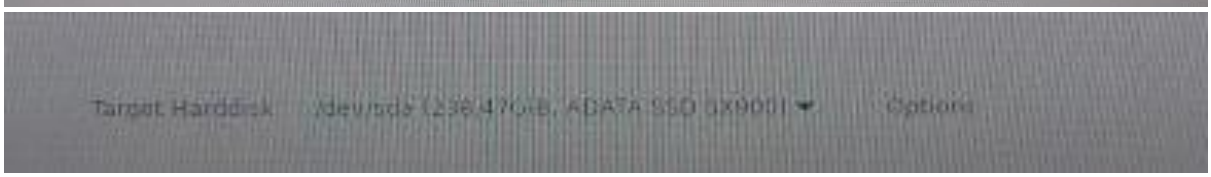



Proxmox Backup Server 4.1 (iso release 1) - https://www.proxmox.com/
PROXMOX
Welcome to Proxmox Backup Server
Install Proxmox Backup Server (Graphical)
Install Proxmox Backup Server (Terminal UI)
Install Proxmox Backup Server (Terminal UI, Serial Console)
Advanced Options

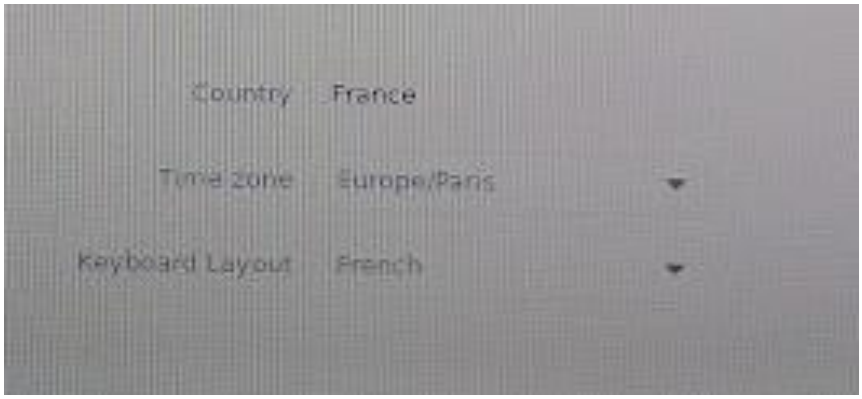

ProLite E2409HDS
Welcome to the Proxmox Backup Server 4.1 (ISO 1) installer!
Starting Proxmox installation
Starting systemd-udevd version 257.9-1~deb13u1
Waiting for /dev to be fully populated
Starting D-Bus daemon...
Copyright 2004-2022 Internet Systems Consortium.
All rights reserved.
For info, please visit https://www.isc.org/software/dhcp/
iiyama


PROXMOX
Proxmox Backup Server
END USER LICENSE AGREEMENT (EULA)
END USER LICENSE AGREEMENT (EULA) FOR PROXMOX BACKUP SERVER
iiyama

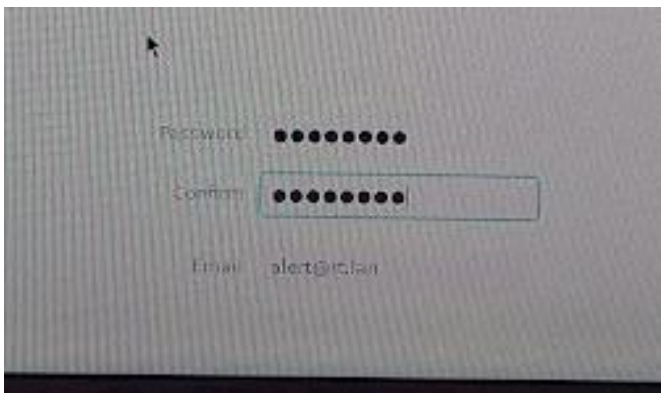
button.
The installer automatically configures your hardware.
Graphical user interface
Final configuration will be done on the graphical user interface, via a web browser.


Target Harddisk
Options

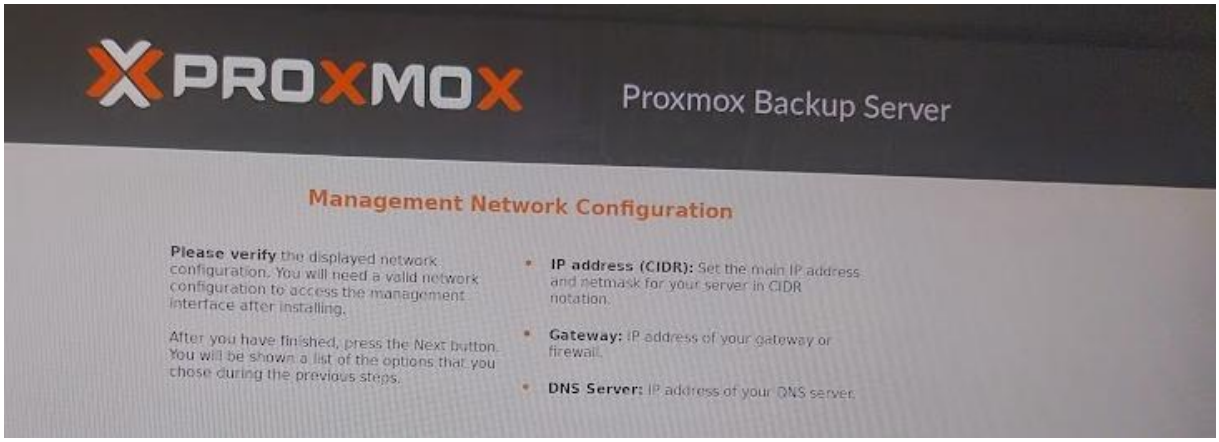

PROXMOX
Proxmox Backup Server
Location and Time Zone selection
The Proxmox Installer will set up your time zone and keyboard layout. Ensuring that your system behaves as intended once it is up and running.
Press the Next button to continue the installation.
Country: Narrows down the available time zones to make selection easier.
Time Zone: Automatically adjust daylight saving time.
Keyboard Layout: Choose your keyboard layout.

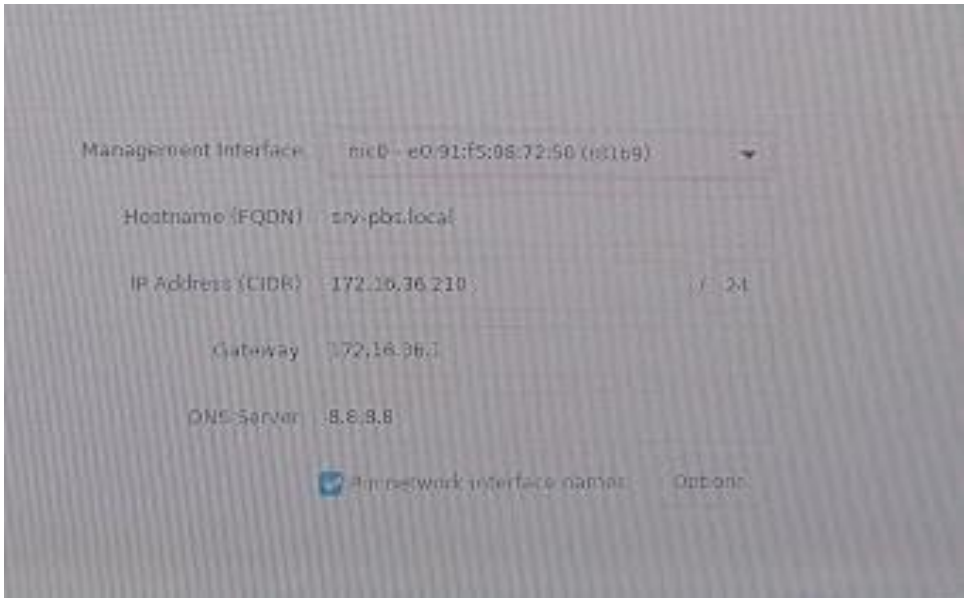

Country France
Time zone Europe/Paris
Keyboard Layout French


PROXMOX
Proxmox Backup Server
Administration Password and Email Address
Proxmox Backup Server is a full-featured, highly secure system, based on Debian GNU/Linux.
In this step, please provide the root password
Password: Please use a strong password. It must be at least 8 characters long, and contain a combination of letters, numbers, and symbols.
Email: Enter a valid email address. Your Proxmox Backup Server will send important alert notifications to this email account (all emails for 'root').
To continue the installation, press the Next button.


Password
Confirm
Email alert@local

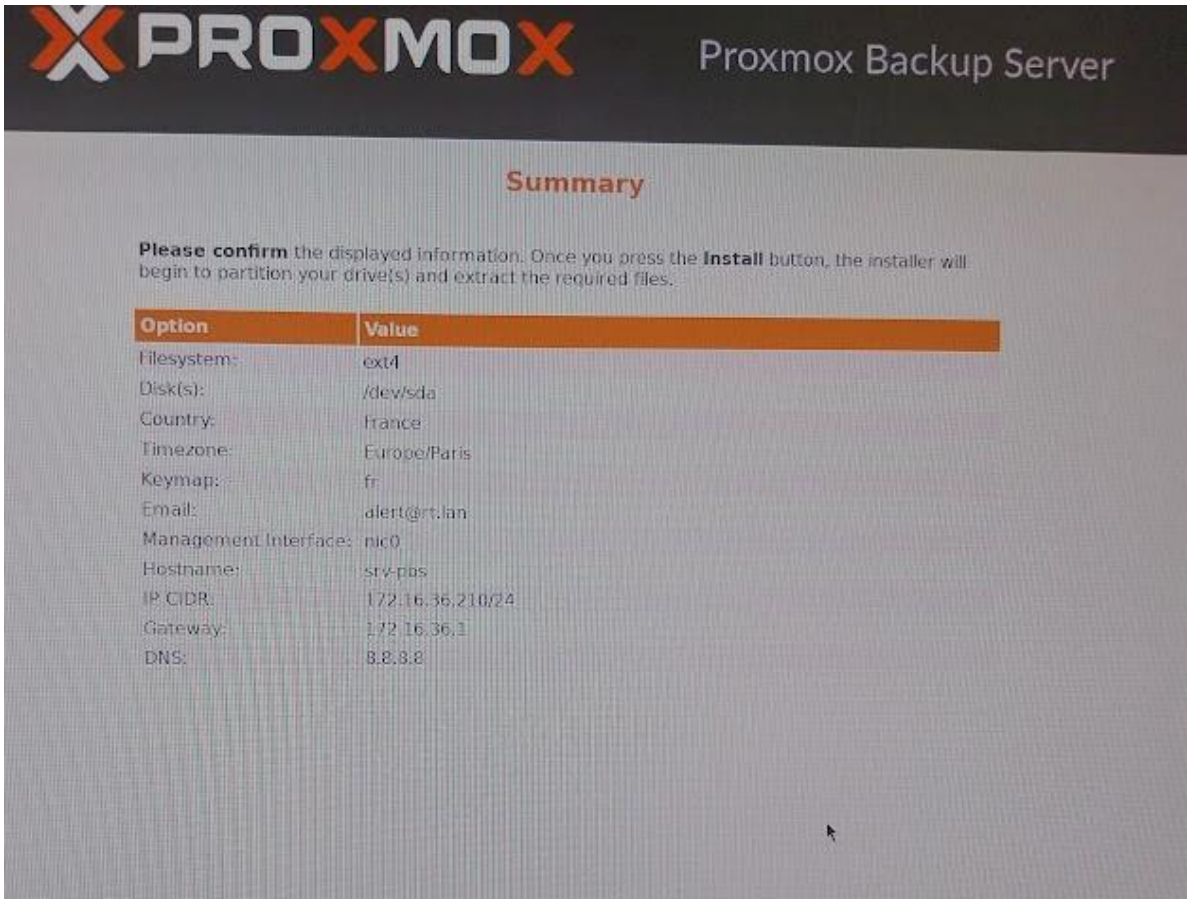
Password: D7TGQ52!
Email: alert@rt.lan

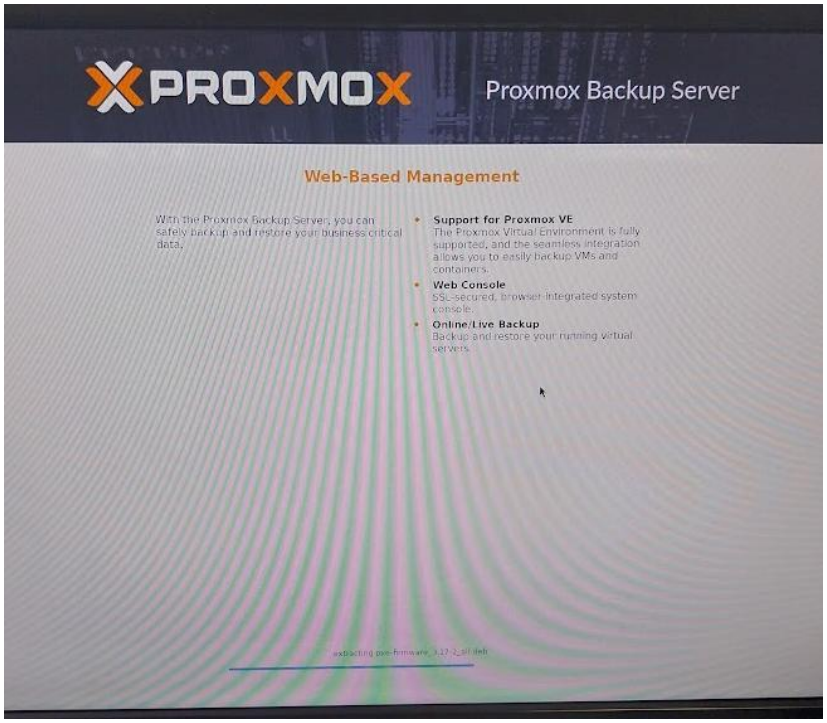




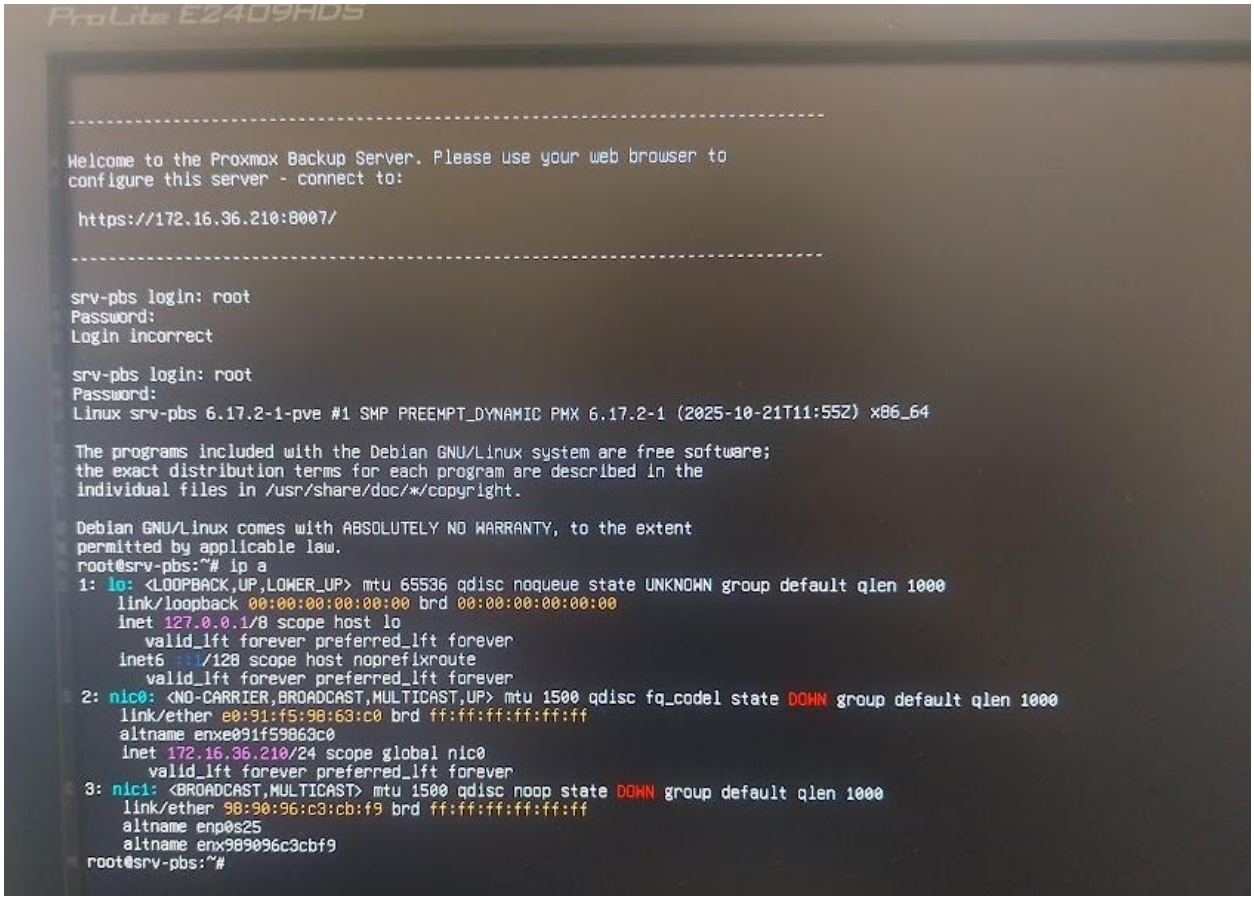
Hostname : srv-pbs.local (pour question de FQDN)
Ip address : 172.16.36.210
Gateway : 172.16.36.1

DNS Server : 8.8.8.8

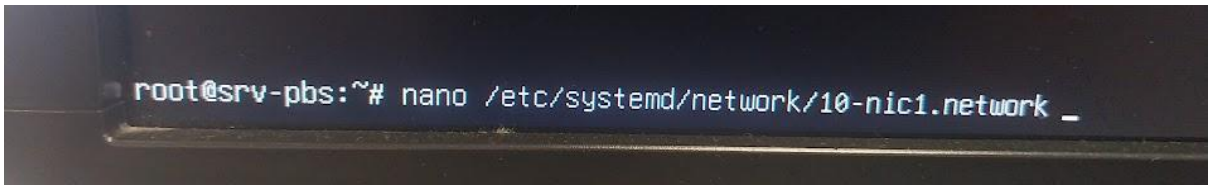


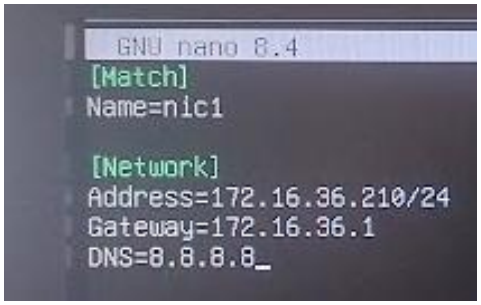


Après avoir installer PBS, on configure les interfaces réseaux pour avoir l'utiliser dans le réseaux.

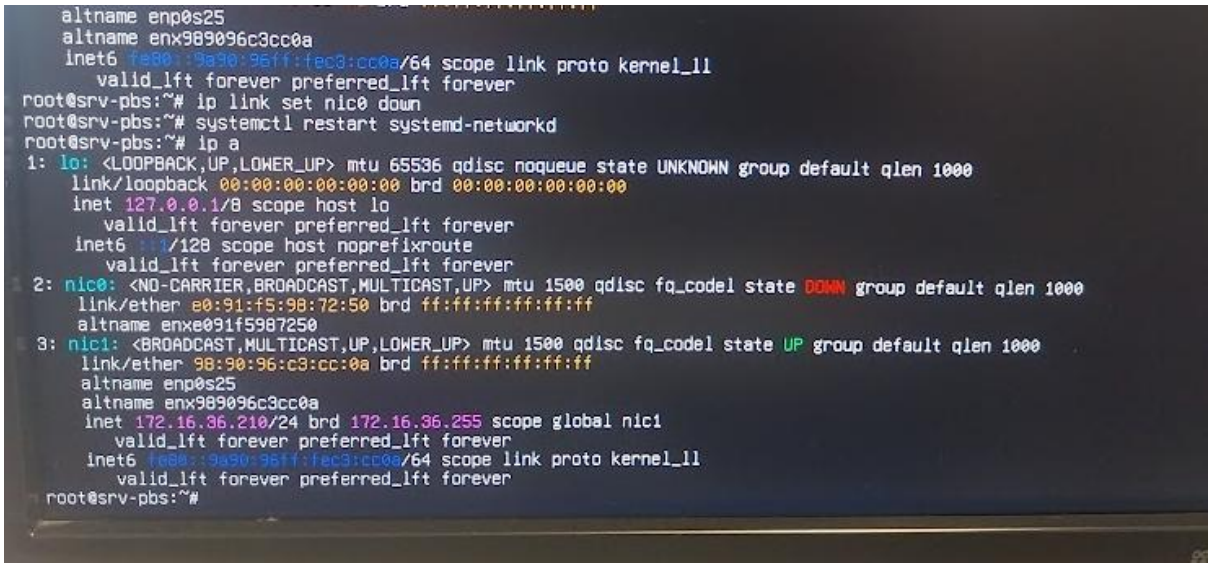


Sur le premier ordinateur, /etc/network. N'avais pas fonctionner de plus pendant mes tests, j'ai remarquer que c'était pas nic0 qui fallait configurer, se dernier ne marcher absolument pas. Et donc, il faillait utiliser la seconde interface nic1. Donc avec l'aide d'une IA. Il ma conseiller de faire ma configuration dans le dossier suivant :

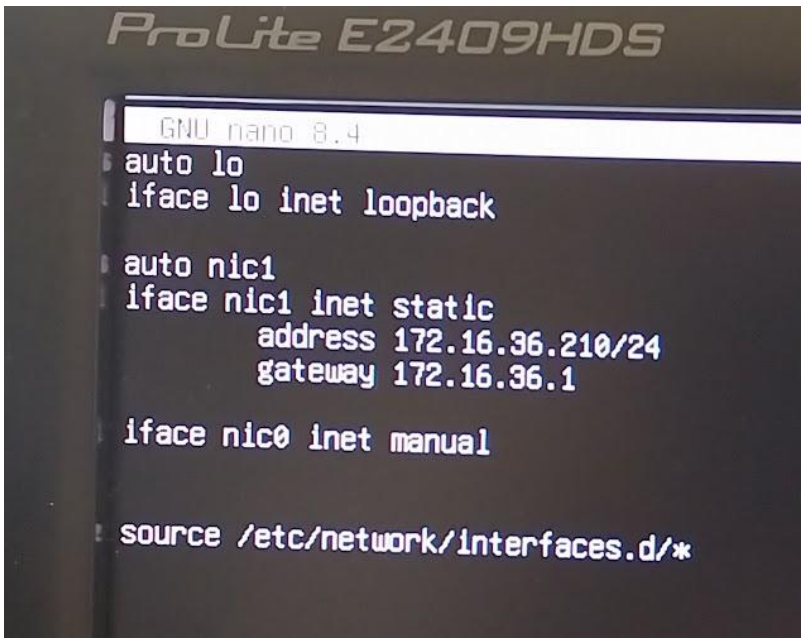


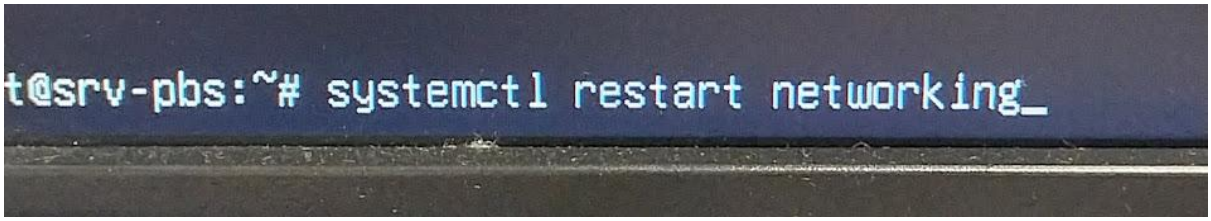


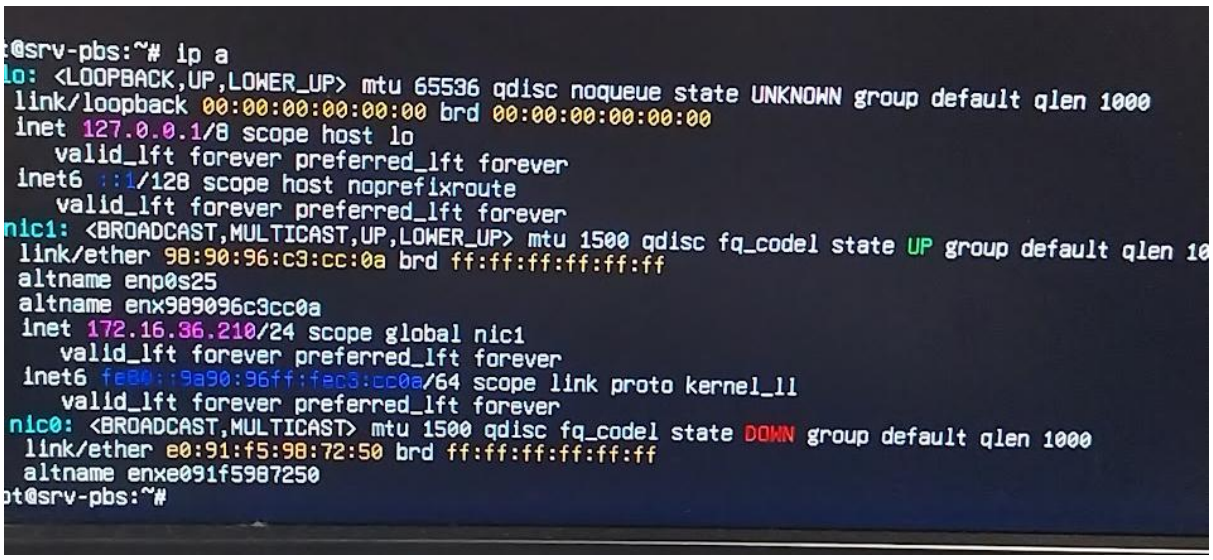
Résultat final :



Cependant quand on le reboot. Il ne garde pas la configuration. Donc, on redemande à l'IA. Et ce dernier me propose de reprendre la configuration normal des interfaces réseaux. Dans /etc/network/interfaces.

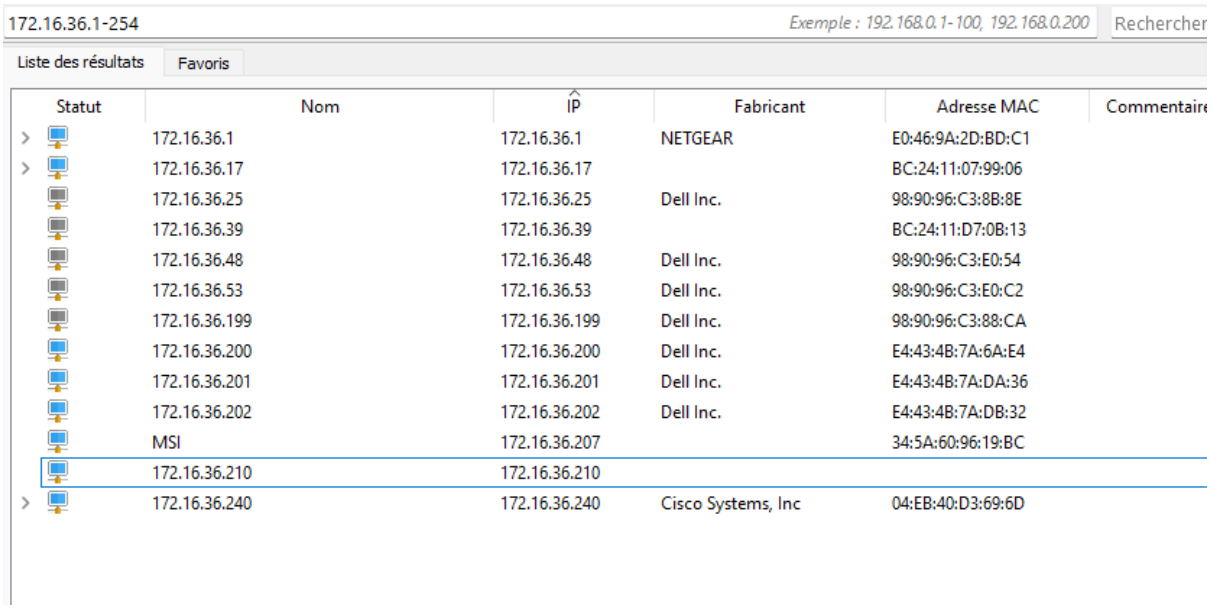


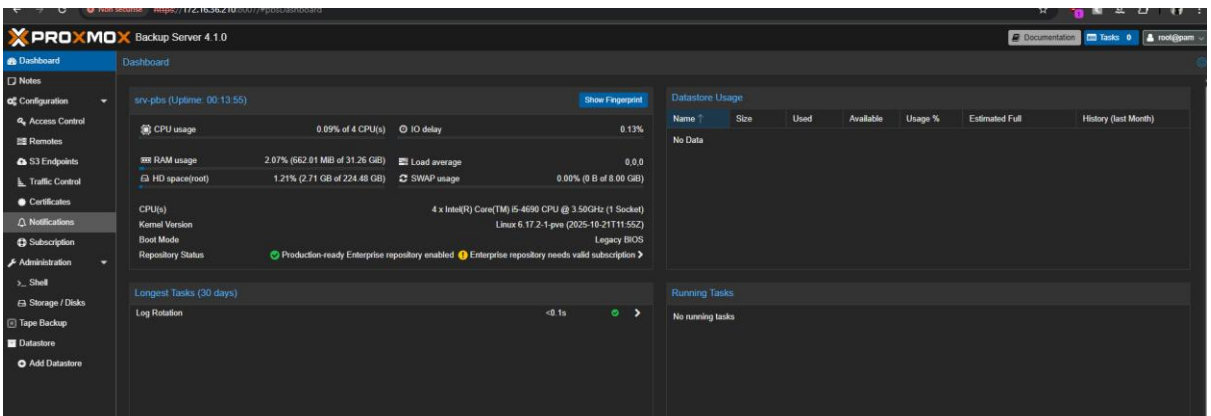




Après des tests. La configuration reste bien après un reboot.

Test de connectivité :





Backup Propre

Terraform

Terraform

terraform.tfvars:

Ce fichier contient les variables nécessaires à l'exécution de la configuration Terraform.

Il permet de définir :

- l'URL de l'endpoint Proxmox
- le token d'authentification API
- le nœud cible (target_node) sur lequel déployer les machines virtuelles
- la liste ou le nombre de VMs à créer
- le nombre de vCPU par VM
- la quantité de mémoire RAM
- la taille du disque
- ainsi que les paramètres réseau et datastore si nécessaire

Ce fichier permet de séparer la configuration dynamique du code Terraform principal (main.tf), facilitant ainsi la réutilisation et la maintenance de l'infrastructure.

Fichier terraform.tfvars:

```
proxmox_endpoint = "https://172.16.36.200:8006/api2/json"
api_token        = "root@pam!terraform=xxxxxxxxxxxxxxxxxxxxxxxxxxxx"
target_node  = "proxmox2"

vm_count = 3
vm_cpu   = 4
vm_ram   = 8192
vm_disk  = 50
```

main.tf:

Ce fichier contient la configuration principale de l'infrastructure Terraform pour le déploiement automatique de machines virtuelles sur Proxmox.

Il comprend :

Déclaration du provider

- Configuration du provider bpg/proxmox
- Définition de l'endpoint Proxmox
- Authentification via API Token
- Activation de la connexion SSH via l'agent local

Déclaration des variables (par défaut si terraform.tfvars est vide)

Les variables permettent de rendre l'infrastructure flexible et réutilisable :

- proxmox_endpoint : URL de l'API Proxmox
- api_token : token d'authentification sécurisé
- target_node : nœud Proxmox cible
- vm_names : liste des machines virtuelles à créer
- vm_cpu : nombre de vCPU par VM
- vm_ram : quantité de mémoire RAM (en MB)
- vm_disk : taille du disque (en GB)
- vm_bridge : bridge réseau utilisé
- vm_datastore : datastore de stockage

Récupération des ressources existantes

Un bloc data permet de récupérer toutes les VMs présentes sur le nœud cible.

Sélection de la template

Un bloc local filtre automatiquement la VM nommée "vm-docker" (notre vm template), utilisée comme template de clonage.
Son vm_id est ensuite utilisé pour créer les nouvelles machines.

Création des machines virtuelles

Les VMs sont créées dynamiquement grâce à for_each à partir de la variable vm_names.

Chaque VM :

- est clonée depuis la template
- reçoit les ressources CPU, RAM et disque définies
- est connectée au bridge réseau spécifié
- active l'agent QEMU
- est protégée contre la suppression accidentelle grâce à

```
lifecycle {
  prevent_destroy = true
}
```

Ce mécanisme garantit un comportement idempotent de plus relancer la commande terraform apply ne supprime pas et ne ne recrée pas les VMs existantes.

Outputs

Un output affiche la liste des machines virtuelles créées et gérées par Terraform.

Fichier main.tf:

```
terraform {
  required_providers {
    proxmox = {
```

```
      source  = "bpg/proxmox"
      version = "~> 0.61"
    }
  }
}
provider "proxmox" {
  endpoint  = var.proxmox_endpoint
  api_token = var.api_token
 insecure  = true
  ssh {
    agent    = true
    username = "root"
  }
}
variable "proxmox_endpoint" {
  type = string
}
variable "api_token" {
  type      = string
  sensitive = true
}
variable "target_node" {
  type = string
}
variable "vm_names" {
  type    = list(string)
```

```
  default = ["tf-vm-1", "tf-vm-2", "tf-vm-3"]
}
variable "vm_cpu" {
  type    = number
  default = 2
}
variable "vm_ram" {
  type    = number
  default = 2048
}
variable "vm_disk" {
  type    = number
  default = 20
}
variable "vm_bridge" {
  type    = string
  default = "vmbr0"
}
variable "vm_datastore" {
  type    = string
  default = "local-lvm"
}
data "proxmox_virtual_environment_vms" "all" {
  node_name = var.target_node
}
locals {
```

```
  template_vm = one([
    for vm in data.proxmox_virtual_environment_vms.all.vms :
    vm
    if vm.name == "vm-docker"
  ])
  template_vm_id = local.template_vm.vm_id
}
resource "proxmox_virtual_environment_vm" "vm" {
  for_each  = { for name in var.vm_names : name => name }
  name      = each.key
  node_name = var.target_node
  clone {
    vm_id = local.template_vm_id
  }
  cpu { cores = var.vm_cpu }
  memory { dedicated = var.vm_ram }
  disk {
    datastore_id = var.vm_datastore
    interface    = "scsi0"
    size         = var.vm_disk
  }
  network_device { bridge = var.vm_bridge }
  agent { enabled = true }
  lifecycle {
    prevent_destroy = true
  }
```

```
}

output "created_vms" {
  value = [for k, v in proxmox_virtual_environment_vm.vm : k]
}
```

Terraform Propre