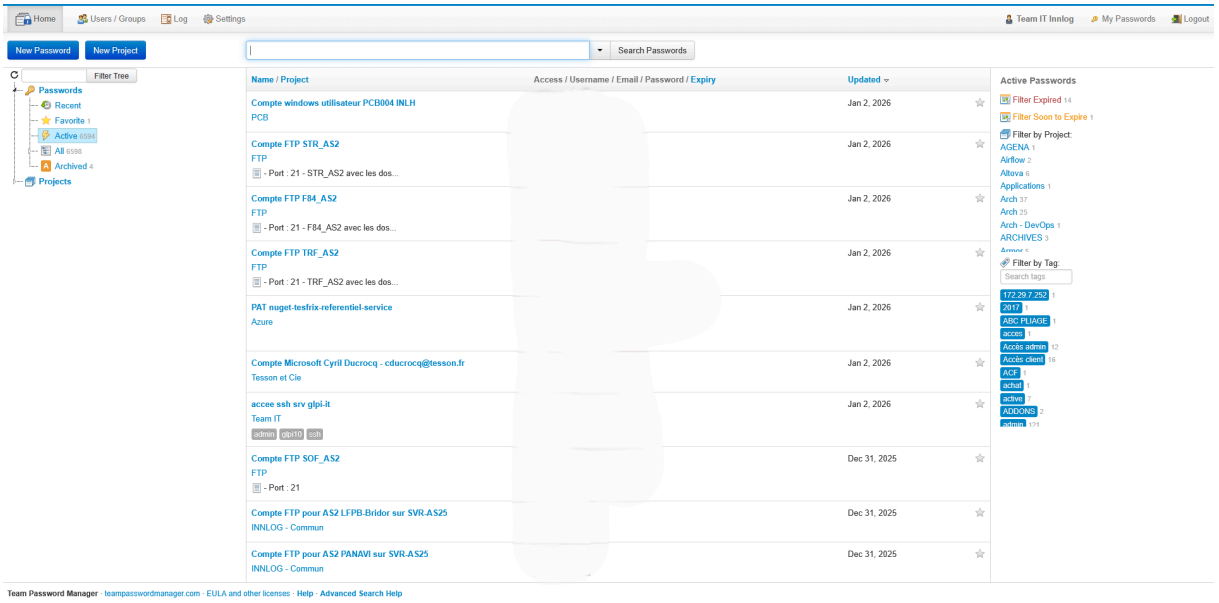
Compte rendu hebdomadaire première semaine

Lundi 5 janvier :
Arrivée dans l'entreprise, mise en place du poste de travail, visite des bureaux et explication du contexte.

Innlog est une filiale du Groupe Tesson, un groupe familial qui a pour cœur de métier l'entreposage et le stockage du froid. Depuis quelques années, le groupe s'est diversifié dans les vins et spiritueux et compte maintenant 5 filiales :

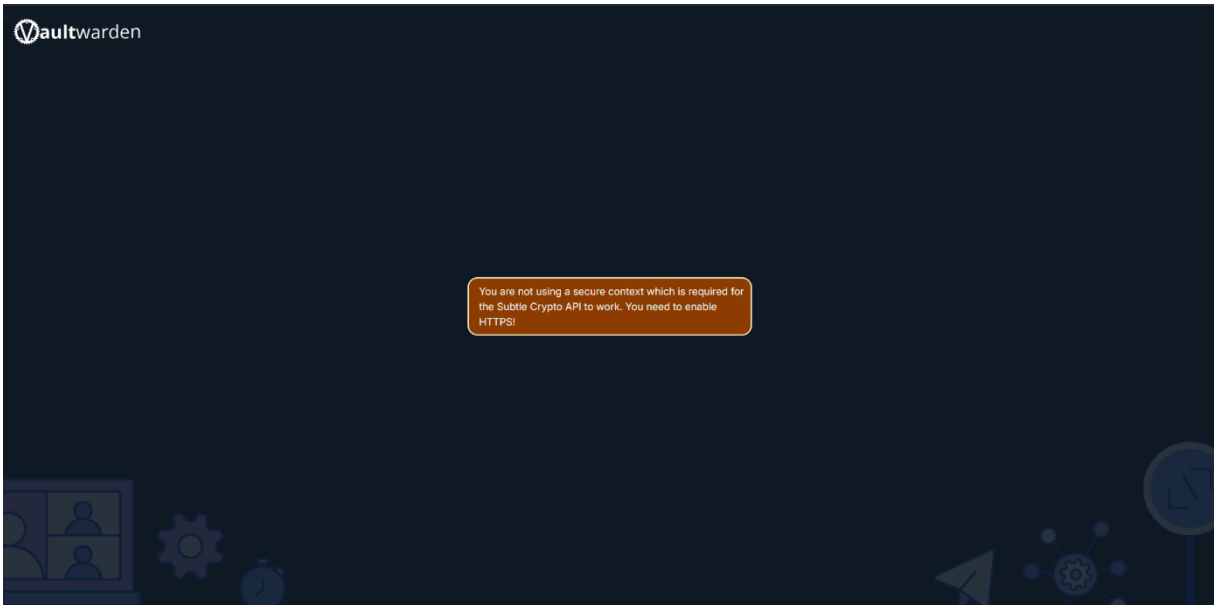
- **Dartess :** qui est une filiale de stockage pour les vins et spiritueux qui sont produits
- **Winetailors :** qui est une filiale qui développe une plateforme pour mettre en relation les fournisseurs et producteurs de vins avec les clients et les particuliers (exemple : vous prenez une bouteille de vin dans un restaurants que vous trouvez excellente et bien Winetailors vous permet de vous mettre en relation avec le producteur de la bouteille afin de pouvoir l'acheter en tant que particuliers
- **Les Salines :** qui est une filiale présente dans les marais salants des sables d'Olonne
- **Frinergy :** qui est la filiale principale en termes de stockage frigorifique (chambres froides etc.)
- **Innlog :** qui est la filiale du groupe s'occupant du développement d'une application Tesfri, qui est en gros une énorme base de données pour les bons de commande, pouvoir scanner les codes-barres et les répertorié dans l'application et qui s'occupe également de toute l'infra du groupe Tesson, la filiale Innlog ne fait maintenant plus parti du groupe Tesson à 100%, en effet une partie a été racheté par le groupe FBO à la Roche sur Yon et deux gros clients de Innlog en tant qu'actionnaire. Le groupe Tesson ne détient maintenant qu'un tiers d'Innlog.

Quand je suis arrivé chez Innlog j'ai tout de suite été accueilli et présenté au reste du groupe ainsi que dans l'environnement dans lequel j'allais travailler. Ma première mission était de me familiariser avec l'infrastructure du groupe et les outils qu'ils utilisent. On m'a tout de suite mis au boulot avec une migration d'un serveur d'hébergement de mot de passe TPM (TeamPasswordManager) vers un serveur Vaultwarden. Pour cette mission j'ai dû prendre en main l'outil de virtualisation PROXMOX car le groupe est en pleine migration de vSphere vers PROXMOX ce qui fera l'objet de mes futures missions également afin de créer une machine virtuelle de test avant de le balancer en prod. On m'avait au préalable cloné la VM qui accueille le « vrai » TPM pour que j'ai un environnement de test au moment de la migration.



L’idée étant de migrer cette base de données de mot de passe vers Vaultwarden il fallait l’installer sur la VM nouvellement créée. A ce stade j’ai accès à l'interface web mais pas l’interface d’administration car il faut absolument que Vaultwarden soit en HTTPS pour fonctionner.

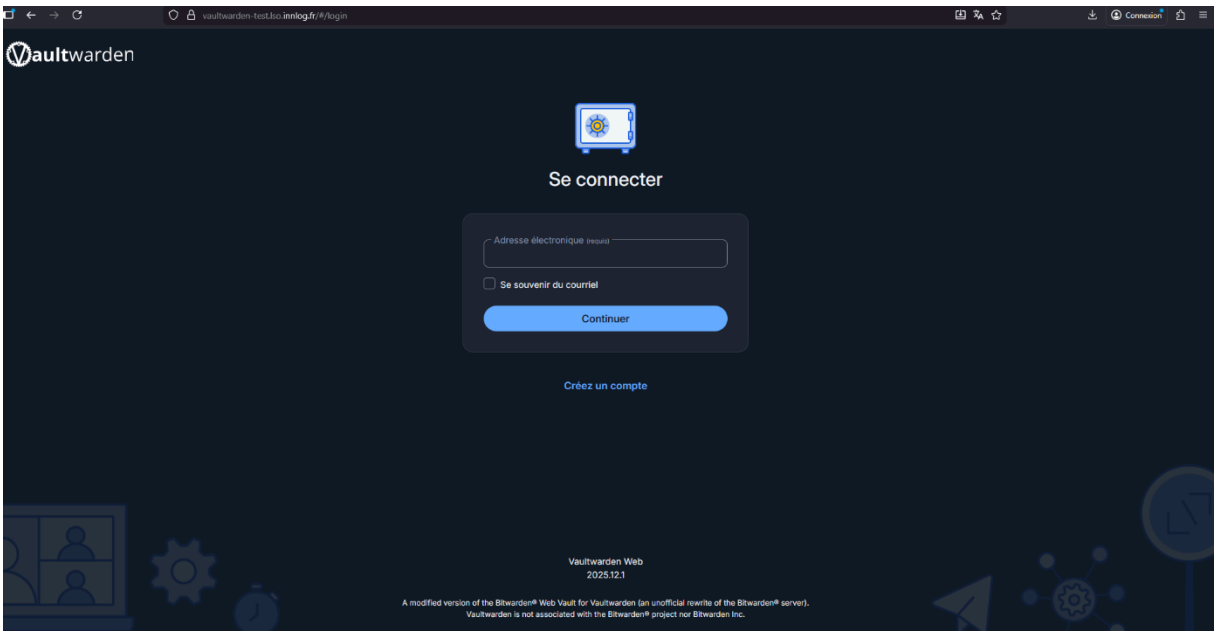
```
root@inl-lso-vaultwarden:/home/enzo# systemctl status vaultwarden
● vaultwarden.service - Bitwarden Server (Rust Edition)
     Loaded: loaded (/usr/lib/systemd/system/vaultwarden.service; enabled; preset: enabled)
     Active: active (running) since Mon 2026-01-05 15:40:34 CET; 55min ago
       Docs: https://github.com/dani-garcia/vaultwarden
   Main PID: 1493 (vaultwarden)
      Tasks: 9 (limit: 2265)
     Memory: 18.4M (peak: 19.1M)
        CPU: 755ms
     CGroup: /system.slice/vaultwarden.service
             └─1493 /usr/bin/vaultwarden

janv. 05 15:40:34 inl-lso-vaultwarden vaultwarden[1493]: |--------------------------------------------------------------------|
janv. 05 15:40:34 inl-lso-vaultwarden vaultwarden[1493]: | This is an *unofficial* Bitwarden implementation, DO NOT use the   |
janv. 05 15:40:34 inl-lso-vaultwarden vaultwarden[1493]: | official channels to report bugs/features, regardless of client.   |
janv. 05 15:40:34 inl-lso-vaultwarden vaultwarden[1493]: | Send usage/configuration questions or feature requests to:         |
janv. 05 15:40:34 inl-lso-vaultwarden vaultwarden[1493]: |   https://github.com/dani-garcia/vaultwarden/discussions or        |
janv. 05 15:40:34 inl-lso-vaultwarden vaultwarden[1493]: |   https://vaultwarden.discourse.group/                             |
janv. 05 15:40:34 inl-lso-vaultwarden vaultwarden[1493]: | Report suspected bugs/issues in the software itself at:            |
janv. 05 15:40:34 inl-lso-vaultwarden vaultwarden[1493]: |   https://github.com/dani-garcia/vaultwarden/issues/new            |
janv. 05 15:40:34 inl-lso-vaultwarden vaultwarden[1493]: \--------------------------------------------------------------------/
janv. 05 15:40:34 inl-lso-vaultwarden vaultwarden[1493]: [2026-01-05 15:40:34.988][start][INFO] Rocket has launched from http://0.0.0.0:8000
```

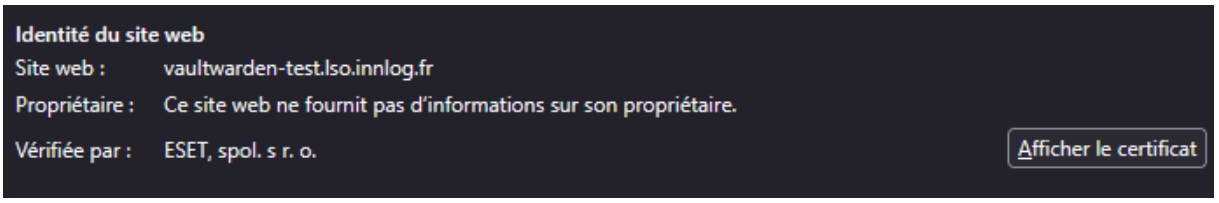


Mardi 6 janvier :

Dans la continuité de ce que j'ai fait lundi, j'ai dû trouver une solution pour utiliser Vaultwarden en HTTPS. Pour ça j'ai eu l'idée d'utiliser NGINX qui est un serveur web comme apache 2 pour me permettre de faire du HTTPS. Tout d'abord un collègue à dû assigner un nom de domaine à l'IP de la machine dans leur Active Directory pour pouvoir avoir un certificat SSL. Dans un premier temps j'ai essayé avec un certificat auto-signé ce qui me permettait d'avoir une connexion HTTPS mais en ayant le message d'erreur me disant que ma connexion n'est pas sécurisée étant donné que le certificat est auto-signé. Cependant chez Innlog ils hébergent leur nom de domaine chez OVH Cloud et on avec ça un outil pour générer des certificats SSL « officiel » avec certbot ce qui permet de ne plus avoir le message d'erreur.

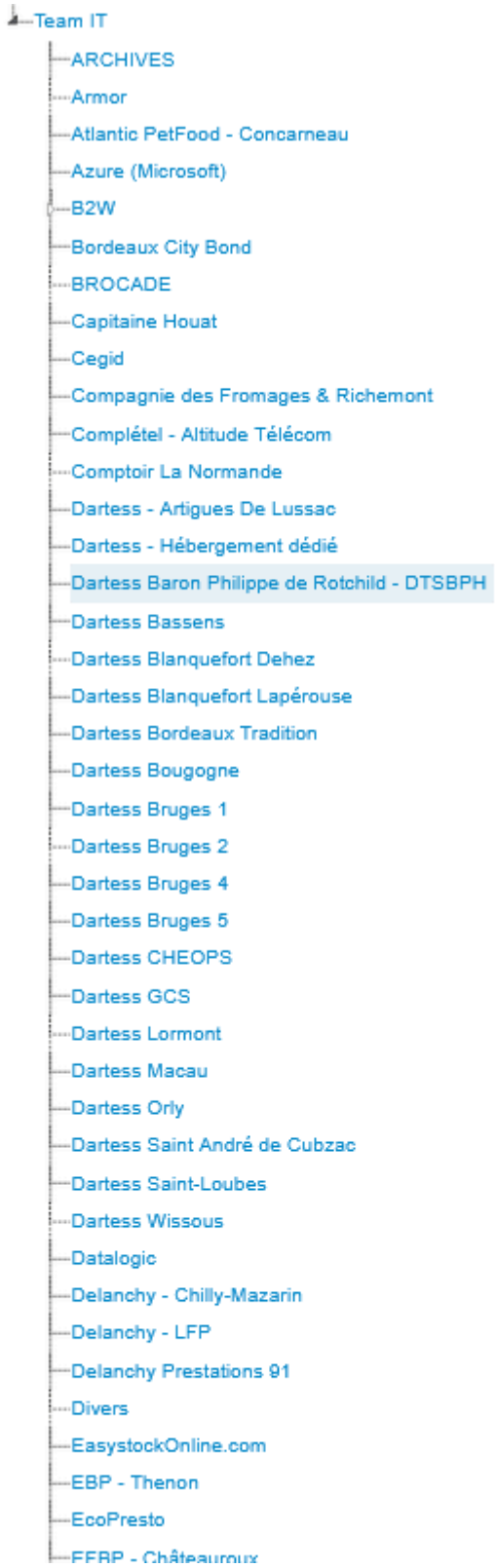
```
server {
    listen 443 ssl;
    server_name vaultwarden-test.lso.innlog.fr;

    ssl_certificate /etc/letsencrypt/live/vaultwarden-test.lso.innlog.fr/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/vaultwarden-test.lso.innlog.fr/privkey.pem;

    location / {
        proxy_pass http://127.0.0.1:8000;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
    }
}

#Redirection HTTP vers HTTPS
server {
    listen 80;
    server_name vaultwarden-test.lso.innlog.fr;

    return 301 https://$host$request_uri;
}
```



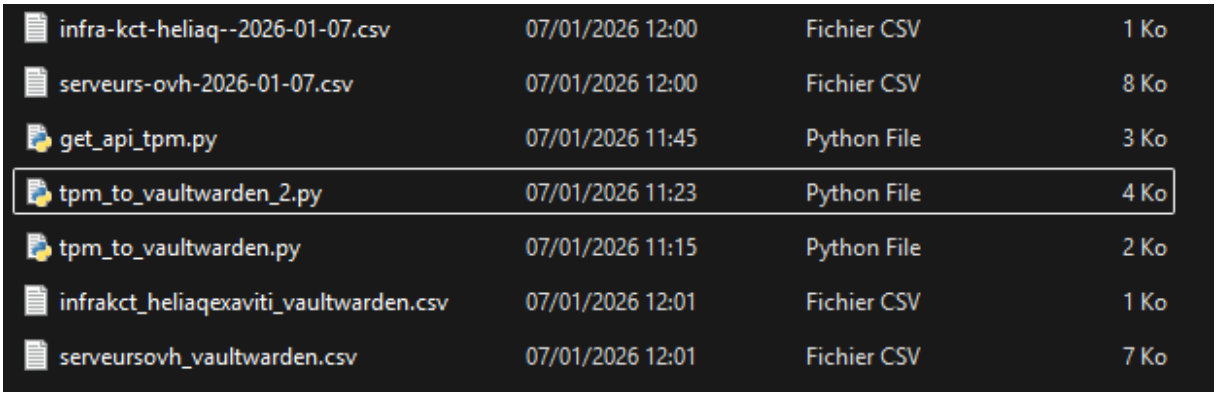


La prochaine étape est de comprendre comment importer la base de données de TPM (TeamPasswordManager) dans Vaultwarden. Le problème est que les champs dans le fichier CSV fourni par TPM ne correspondent pas avec les champs de Vaultwarden, j'avais donc pensé à importer d'abord le CSV de TPM dans Keepass pour ensuite qu'il soit plus compatible avec Vaultwarden mais ça ne fonctionne pas non plus car le fichier CSV de base de TPM n'a pas la notion de hiérarchie des groupes présents dans la base de données ce qui fait qu'il crache juste les mots de passe quand ils sont importés sans les classer dans leurs groupes respectif.

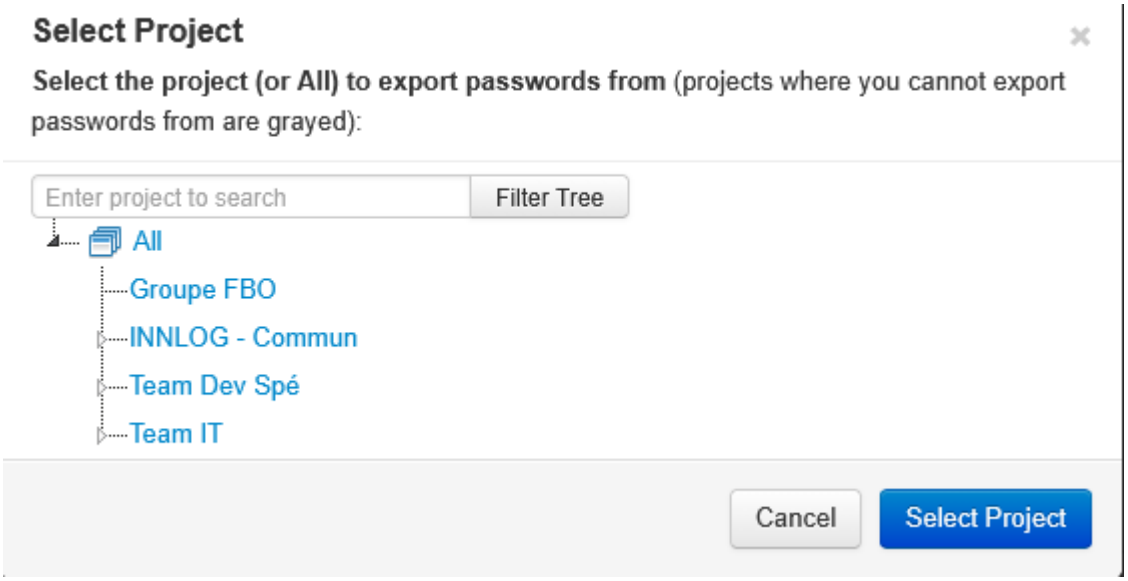
Mercredi 7 janvier :
Pour ce mercredi j'ai continué sur la migration de base de données TPM vers Vaultwarden, et je me suis penché sur ces fameux fichiers CSV. Pour télécharger les fichiers CSV de la base de données TPM, il faut les télécharger un par un à la main, sachant qu'il y a plus de 6500 entrées dans leur base de données la tâche aurait été infaisables à la main. J'ai donc réfléchi à plusieurs solutions, la première était de faire un script que ce soit en SQL ou en Python peu importe et la deuxième était de récupérer les mots de passe grâce à l'API de TPM vu que cet outil en possède une. Cependant, l'API ne renvoie rien, je suis donc passé au script et je suis parti du Python. Pour réaliser ce script je me suis aidé de l'IA.



Le premier souci que j'ai rencontré était que les champs CSV de TPM et de Vaultwarden ne sont pas du tout les mêmes j'ai dû donc réaliser un script à l'aide de l'IA qui prend tous les fichiers CSV qui sont présents au même endroit que le script et qui modifie ces champs et remet les données au bon endroit.



Une fois que je pouvais importer les fichiers CSV dans Vaultwarden dans le bon format, il fallait que je me procure ces fichiers CSV. La base de données contenant plus de 6500 entrées je n'allais pas le faire à la main. J'ai donc utilisé la librairie **Playwright** qui est une librairie Python permettant l'automatisation de navigateurs web, elle peut lancer un navigateur, ouvrir des pages, attendre certains éléments ou encore récupérer des données dans notre cas des fichiers CSV, tout ça en se basant sur les attributs HTML comme les balises « li », « a » etc. ou encore des chemin CSS



```python
import os
import time
from playwright.sync_api import sync_playwright

# --------------------------
# CONFIGURATION
# --------------------------
TPM_URL = ""
USERNAME = ""
PASSWORD = ""

DOWNLOADS_FOLDER = os.path.join(os.environ["USERPROFILE"], "Downloads")


# --------------------------
# FONCTIONS UTILES
# --------------------------

def return_to_export_menu(page):
    """Retourne au menu Export après un export."""
    page.wait_for_selector('a.btn[href*="settings/view/export_import"]', timeout=10000)
    page.click('a.btn[href*="settings/view/export_import"]')
    page.wait_for_timeout(1500)

    page.wait_for_selector('a.btn.btn-primary[href*="settings/export"]', timeout=10000)
    page.click('a.btn.btn-primary[href*="settings/export"]')
    page.wait_for_timeout(1500)


def open_and_expand_team_it(page):
    """
    Ouvre la modale Select Project, attend qu'elle soit visible,
    attend que Team IT soit présent, puis déroule Team IT.
    """
    page.click('#tpm_select_project_export')
    page.wait_for_timeout(500)

    page.wait_for_selector("#tree_container_modal", timeout=10000)
    page.wait_for_selector("#pwd_prj_1", timeout=10000)

    # Dérouler Team IT
    page.click("#pwd_prj_1 > .jstree-ocl")
    page.wait_for_timeout(800)

    # Attendre que les sous-dossiers apparaissent
    page.wait_for_selector("#pwd_prj_1 ul li a.jstree-anchor", timeout=5000)


def get_team_it_children(page):
    """Retourne les sous-dossiers de Team IT."""
    return page.query_selector_all("#pwd_prj_1 ul li a.jstree-anchor")


def export_current_project(page, name):
    """Select Project, Export, attend le téléchargement, retourne au menu."""
    print(f"➡ Export du projet : {name}")

    page.click('#tpm_select_project_modal_submit_button')
    page.wait_for_timeout(800)
```

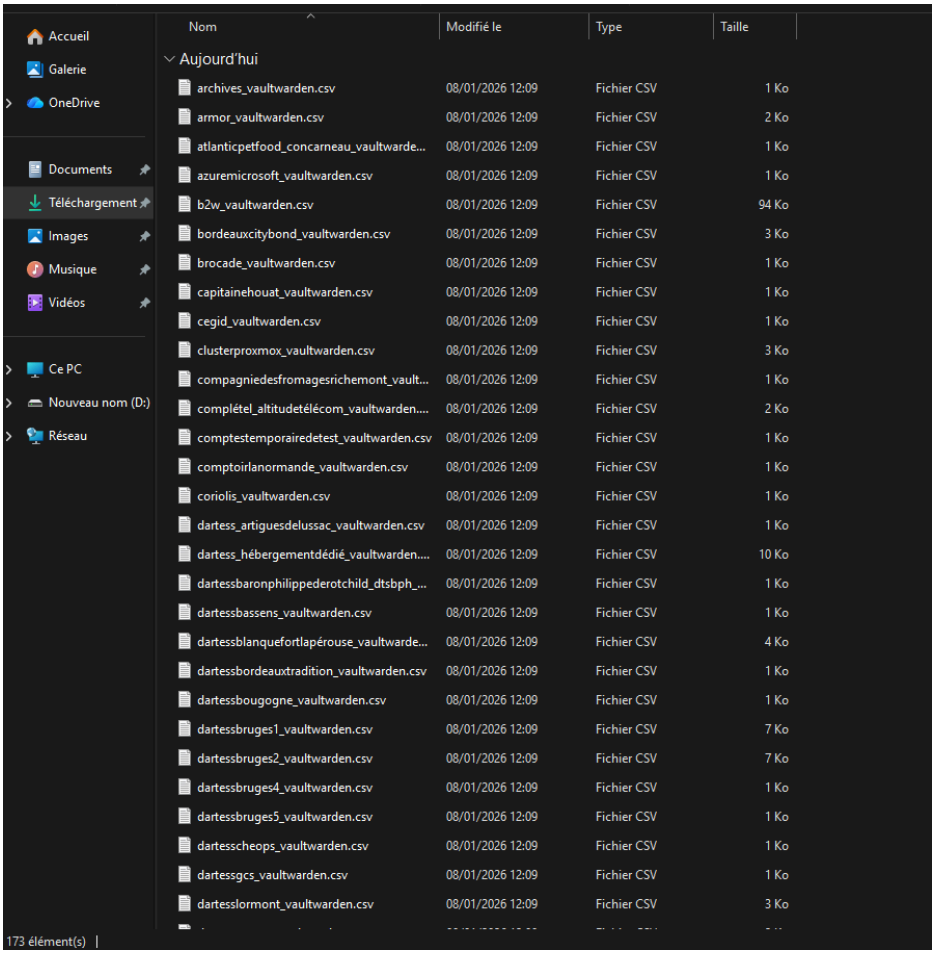
```
# --------------------------
# SCRIPT PRINCIPAL
# --------------------------
with sync_playwright() as p:
    browser = p.chromium.launch(headless=False)
    page = browser.new_page(accept_downloads=True)

    print("🌐 Connexion à TPM…")
    page.goto(TPM_URL)
    page.fill('input[name="username"]', USERNAME)
    page.fill('input[name="password"]', PASSWORD)
    page.click('input[name="sign_in"]')
    page.wait_for_timeout(5000)

    print("➡ Ouverture de Settings → Export / Import…")
    page.click('a[href*="settings"]')
    page.click('a[href*="export"]')
    page.wait_for_timeout(2000)

    print("➡ Ouverture du module d'export…")
    page.click('a.btn.btn-primary[href*="export"]')
    page.wait_for_timeout(2000)

    # --------------------------
    # 1) EXPORT DU DOSSIER TEAM IT
    # --------------------------
    print("\n📂 Ouverture du menu des projets…")
    open_and_expand_team_it(page)

    team_it_anchor = page.query_selector("#pwd_prj_1 > a.jstree-anchor")
    team_it_anchor.click()
    page.wait_for_timeout(300)

    export_current_project(page, "Team IT")

    # --------------------------
    # 2) EXPORT DES SOUS-DOSSIERS DE TEAM IT (+ LEURS SOUS-DOSSIERS)
    # --------------------------
    done = []  # sous-dossiers de Team IT déjà exportés (niveau 1)

    while True:
        print("\n📂 Réouverture du menu des projets…")
        open_and_expand_team_it(page)

        children = get_team_it_children(page)

        # Filtrer les sous-dossiers de Team IT déjà exportés
        remaining = [c for c in children if c.inner_text().strip() not in done]

        if not remaining:
            print("🎉 Tous les sous-dossiers de Team IT ont été exportés.")
            break

        # On prend le premier sous-dossier restant
        sub = remaining[0]
        name = sub.inner_text().strip()

        print(f"➡ Dossier de Team IT : {name}")
```

Là où j'en suis rendu actuellement est que le script fait toutes les manipulations pour atteindre le menu d'exportation des fichiers CSV qu'il déroule les sous menus de la base de données mais il ne détecte pas si un sous dossier a lui-même un sous dossier. Exemple : avec une arborescence comme celle-ci ;

Le script va prendre **Team IT** et télécharger son CSV ensuite il rouvrir le menu et passer à **ARCHIVES**, de même avec **Armor** et **B2W** mais il ne va pas passer dans **B2W** pour voir s'il y a OVH donc il ne pas télécharger OVH mais passer au dossier après **B2W.**

Jeudi 8 janvier :

Pour ce jeudi, j'ai donc réussi à télécharger tous les fichiers csv de Team IT, je les ai convertis pour qu'ils aient un format compatible avec Vaultwarden au niveau de champs CSV.



J'ai donc dû m'attaquer au dossier Innlog - Commun qui lui contient énormément de sous dossiers, de sous dossiers de sous dossiers etc. Pour ça j'ai refait un script Python à l'aide de l'IA que vraiment le script agisse récursivement sur les dossiers et qu'il me les télécharge un par un dans un dossier Migration TPM. J'ai également rajouté la création d'un fichier de log pour avoir une preuve des sorties du script Python. De plus il y avait des groupes ou il n'y avait aucun mot de passe et donc ça faisait crasher le script étant donné qu'il attend un fichier CSV et que la ba il y en a pas. On a donc eu l'idée de faire un manifest en .json qui stock les dossiers déjà traité et téléchargé.

```
2026-01-08 16:43:05,786 - INFO - 🌐 Connexion à TPM...
2026-01-08 16:43:09,961 - INFO - 🚀 Cible actuelle : INNLOG - Commun
2026-01-08 16:43:17,440 - INFO -    ✨ Nouveau sous-dossier détecté : TESFRI
2026-01-08 16:43:17,440 - INFO -    ✨ Nouveau sous-dossier détecté : NEXT
2026-01-08 16:43:17,440 - INFO -    ✨ Nouveau sous-dossier détecté : Internet
2026-01-08 16:43:17,441 - INFO -    ✨ Nouveau sous-dossier détecté : INNPULSE
2026-01-08 16:43:17,441 - INFO -    ✨ Nouveau sous-dossier détecté : INNLOG
2026-01-08 16:43:17,441 - INFO -    ✨ Nouveau sous-dossier détecté : HARMATTAN
2026-01-08 16:43:17,441 - INFO -    ✨ Nouveau sous-dossier détecté : GESTCOM
2026-01-08 16:43:17,442 - INFO -    ✨ Nouveau sous-dossier détecté : Frigo Connecté
2026-01-08 16:43:17,442 - INFO -    ✨ Nouveau sous-dossier détecté : Environnements PROD Clients
2026-01-08 16:43:17,442 - INFO -    ✨ Nouveau sous-dossier détecté : Environnement Sofrilog Temporaire
2026-01-08 16:43:17,442 - INFO -    ✨ Nouveau sous-dossier détecté : Environnement RECETTE
2026-01-08 16:43:17,442 - INFO -    ✨ Nouveau sous-dossier détecté : Environnement INTERNE
2026-01-08 16:43:17,443 - INFO -    ✨ Nouveau sous-dossier détecté : Environnement DEV
2026-01-08 16:43:17,443 - INFO -    ✨ Nouveau sous-dossier détecté : Environnement DEMO
2026-01-08 16:43:17,443 - INFO -    ✨ Nouveau sous-dossier détecté : DARTESS
2026-01-08 16:43:17,443 - INFO -    ✨ Nouveau sous-dossier détecté : Applications
2026-01-08 16:43:17,993 - INFO -    📩 Exportation de : INNLOG - Commun...
2026-01-08 16:43:20,215 - INFO -    ✅ Téléchargé : INNLOG - Commun.csv
2026-01-08 16:43:20,217 - INFO - 🚀 Cible actuelle : INNLOG - Commun > Applications
2026-01-08 16:43:25,440 - INFO -    ✨ Nouveau sous-dossier détecté : Logins
2026-01-08 16:43:25,440 - INFO -    ✨ Nouveau sous-dossier détecté : Licences
2026-01-08 16:43:25,972 - INFO -    📩 Exportation de : Applications...
2026-01-08 16:43:26,319 - INFO -    ✅ Téléchargé : Applications.csv
2026-01-08 16:43:26,320 - INFO - 🚀 Cible actuelle : INNLOG - Commun > Applications > Licences
2026-01-08 16:43:31,980 - INFO -    📩 Exportation de : Licences...
2026-01-08 16:43:32,337 - INFO -    ✅ Téléchargé : Licences.csv
2026-01-08 16:43:32,338 - INFO - 🚀 Cible actuelle : INNLOG - Commun > Applications > Logins
2026-01-08 16:43:37,946 - INFO -    📩 Exportation de : Logins...
2026-01-08 16:43:38,439 - INFO -    ✅ Téléchargé : Logins.csv
2026-01-08 16:43:38,440 - INFO - 🚀 Cible actuelle : INNLOG - Commun > DARTESS
2026-01-08 16:43:43,519 - INFO -    ✨ Nouveau sous-dossier détecté : Flux EDI Dartess
2026-01-08 16:43:43,520 - INFO -    ✨ Nouveau sous-dossier détecté : Connexion RDP
```

JSON Données brutes En-têtes

Enregistrer Copier Tout réduire Tout développer Filtrer le JSON

```
stack:        []
▼ done_paths:
    0:        "INNLOG - Commun"
    1:        "INNLOG - Commun > Applications"
    2:        "INNLOG - Commun > Applications > Licences"
    3:        "INNLOG - Commun > Applications > Logins"
    4:        "INNLOG - Commun > DARTESS"
    5:        "INNLOG - Commun > DARTESS > Connexion RDP"
```

Ce qui fait que maintenant je vais avoir tous les fichiers CSV de toutes les entrées de mot de passe de la base de données. Il faut maintenant que je trouve un moyen d'importer tous ces fichiers dans Vaultwarden.

Vendredi 9 janvier :

Arrivé au vendredi mon script fonctionne très bien, cependant une autre problématique est arrivée, le script lui parcourt tous les dossiers à partir de Innlog - Commun et va rencontrer par exemple un premier dossier qui s'appelle Applications et va le télécharger et ensuite il va continuer sa boucle récursivement, cependant si il rencontre un autre dossier qui s'appelle Applications qui se trouve dans une autre

branche il va reprendre sa boucle au premier dossier Applications c'est à dire qu'il tourne en boucle.
Arrivé au milieu de la journée je me suis penché sur l'API que possède TPM pour essayer de récupérer les mots de passe en plus grande quantité et plus facilement.
J'ai d'abord essayé d'interroger l'API via **curl** sous linux, dans la doc la commande c'était

**curl -u username:password **
**-H 'Content-Type: application/json; charset=utf-8' **
-i https://tpm.mydomain.com/index.php/api/v6/passwords.json

j'ai donc adapté la commande et au départ elle ne fonctionnait pas, déjà la v6 de l'API n'existe pas j'ai donc dû me rabattre sur la v5 ce qui donne une URL comme ça

curl -u <login TPM>:"**mdp pour se connecter à TPM**" -H 'Content-Type: application/json; charset=utf-8' -i https://**url_du_tpm**/index.php/api/v5/passwords/page/1.json

En faisant cette commande je tombe sur un fichier .json que j'ai importé dans VScode pour le mettre en forme afin que ce soit plus visible. Et j'ai récupéré des entrées comme ça :


```
{
    "id": 8980,
    "name": "",
    "project": {
        "id": 1418,
        "name": "FTP"
    },
    "notes_snippet": "",
    "tags": "",
    "access_info": "",
    "username": "",
    "email": "",
    "has_password": true,
    "expiry_date": null,
    "expiry_status": 0,
    "archived": false,
    "project_archived": false,
    "favorite": false,
    "num_files": 0,
    "locked": false,
    "locking_type": 0,
    "external_sharing": false,
    "linked": false,
    "updated_on": "2026-01-02 12:18:40"
},
```

les doubles cotes étant vide pour un souci de confidentialité. Donc dans ces champs je n'ai pas le mot de passe en clair, cependant j'ai des **id** qui vont me permettre de refaire un curl et d'ajouter cet **id** à l'url avec ça `GET /passwords/ID.json`
si je reprend mon **curl** ça va donner quelque chose comme ça avec l'id du fichier ci dessus **curl -u <login tpm>:"mdp pour se connecter à tpm" -H 'Content-Type: application/json; charset=utf-8' -i https://url_tpm/index.php/api/v5/passwords/8980.json**
ce qui va me donner un fichier json comme ça :


```
{
   "id": 8980,
   "name": "",
   "project": {
      "id": 1418,
      "name": "FTP"
   },
   "tags": "",
   "access_info": "",
   "username": "",
   "email": "",
   "password": "",
   "expiry_date": null,
   "expiry_status": 0,
   "notes": "",
   "custom_field1": null,
   "custom_field2": null,
   "custom_field3": null,
   "custom_field4": null,
   "custom_field5": null,
   "custom_field6": null,
   "custom_field7": null,
   "custom_field8": null,
   "custom_field9": null,
   "custom_field10": null,
   "users_permissions": null,
   "groups_permissions": null,
   "parents": [
      1180,
      1217,
      1218,
      1412,
      1418
   ],
   "user_permission": {
      "id": 30,
      "label": "Manage"
   },
   "archived": false,
```

```
    "project_archived": false,
    "favorite": false,
    "num_files": 0,
    "locked": false,
    "locking_type": 0,
    "locking_request_notify": 0,
    "external_sharing": false,
    "external_url": null,
    "linked": false,
    "source_password_id": 0,
    "managed_by": {
        "id": 2,
        "username": "",
        "email_address": "",
        "name": "",
        "role": "IT"
    },
    "created_on": "2025-12-31 10:53:29",
    "created_by": {
        "id": 2,
        "username": "",
        "email_address": "",
        "name": "",
        "role": "IT"
    },
    "updated_on": "2026-01-02 12:18:40",
    "updated_by": {
        "id": 2,
        "username": "",
        "email_address": "",
        "name": "",
        "role": "IT"
    }
}
```


cette fois ci avec un champs **password** qui contient bien le mot de passe en clair
Donc pour l’instant j’en suis rendu là, il faudra que je montre mes avancements au responsable IT qui revient lundi pour voir la meilleure solution possible car c’est une manipulation beaucoup plus compliquée que prévu.

Compte rendu hebdomadaire deuxième semaine

Lundi 12 janvier :
Ce lundi 12 janvier, le responsable étant revenu je lui ai exposé mon travail de la semaine dernière avec mes réussites et mes échecs et la question première était de savoir si je continuais sur ça de peur d'y passer une bonne partie de mon stage, cependant d'autres missions m'ont été proposées comme le **déploiement d'une infrastructure wifi sur une infrastructure vieillissante d'un des gros clients de Innlog** et aussi **le déménagements physique d'une bonne partie de leur infra qui est basé dans un data center à la Roche sur Yon et que la boîte à décider de migrer vers un datacenter à Nantes.** Le responsable Anthony et un autre collègue, Maxime m'ont donc proposé d'approfondir les call API ce qui a été une bonne chose et d'ensuite couper mon stage en plusieurs missions.

Suite de la migration de TPM vers vaultwarden, je me suis définitivement penché sur les call API pour arriver à mes fins, les scripts crées dans la matinée ont fonctionnés, c'est à dire que la j'ai toutes les entrées de mots de passe dans un fichier json qu'ensuite j'ai passé dans un autre script afin de récupérer les mots de passe en clair avec l'ID : rappel de la manipulation.

Une première commande **curl** me permettait d'arriver à ce résultat :

```
{
"id": 8980,
"name": "",
"project": {
"id": 1418,
"name": "FTP"
},
"notes_snippet": "",
"tags": "",
"access_info": "",
"username": "",
"email": "",
"has_password": true,
"expiry_date": null,
"expiry_status": 0,
"archived": false,
"project_archived": false,
"favorite": false,
"num_files": 0,
"locked": false,
"locking_type": 0,
"external_sharing": false,
"linked": false,
"updated_on": "2026-01-02 12:18:40"
},
```


Sans mot de passe mais avec un ID, celui qui m'intéresse est le premier qui est l'ID de l'entrée et l'autre étant celui du projet dans lequel est le mot de passe.

Cependant dans la doc une autre commande **curl** était précisé pour avoir le mot de passe lié à cet ID justement ce qui me donnait une sortie json comme ça :


```
{
"id": 8980,
"name": "",
"project": {
"id": 1418,
"name": "FTP"
},
"tags": "",
"access_info": "",
"username": "",
"email": "",
"password": "",
```

```
"expiry_date": null,
"expiry_status": 0,
"notes": "",
"custom_field1": null,
"custom_field2": null,
"custom_field3": null,
"custom_field4": null,
"custom_field5": null,
"custom_field6": null,
"custom_field7": null,
"custom_field8": null,
"custom_field9": null,
"custom_field10": null,
"users_permissions": null,
"groups_permissions": null,
"parents": [
1180,
1217,
1218,
1412,
1418
],
"user_permission": {
"id": 30,
"label": "Manage"
},
"archived": false,

"project_archived": false,
"favorite": false,
"num_files": 0,
"locked": false,
"locking_type": 0,
"locking_request_notify": 0,
"external_sharing": false,
"external_url": null,
"linked": false,
"source_password_id": 0,
"managed_by": {
"id": 2,
"username": "",
"email_address": "",
```

```
"name": "",
"role": "IT"
},
"created_on": "2025-12-31 10:53:29",
"created_by": {
"id": 2,
"username": "",
"email_address": "",
"name": "",
"role": "IT"
},
"updated_on": "2026-01-02 12:18:40",
"updated_by": {
"id": 2,
"username": "",
"email_address": "",
"name": "",
"role": "IT"
}
}
```


Cette fois-ci avec un champ mot de passe en clair.

Il m'a simplement suffit d'automatiser tout ça avec **deux scripts Python** (toujours à l'aide de l'IA)

- le premier qui s'occupait de faire une itération avec la commande curl étant de donné que dans l'URL ce sont des numéros de page il a simplement à faire une boucle jusqu'à ce qu'il n'y est plus de page et de mettre les sorties dans un fichier json ce qui donnait quelque chose comme ça dans les logs.

```
...
2026-01-12 11:26:05,611 - INFO - Page 326 récupérée.
2026-01-12 11:26:06,231 - INFO - Page 327 récupérée.
2026-01-12 11:26:06,666 - INFO - Page 328 récupérée.
2026-01-12 11:26:07,136 - INFO - Page 329 récupérée.
2026-01-12 11:26:07,206 - INFO - Page 332 est vide. Signal d'arrêt
envoyé.
2026-01-12 11:26:07,212 - INFO - Page 330 récupérée.
2026-01-12 11:26:07,251 - INFO - Page 331 récupérée.
2026-01-12 11:26:07,651 - INFO - Page 333 est vide. Signal d'arrêt
envoyé.
2026-01-12 11:26:07,814 - INFO - Page 334 est vide. Signal d'arrêt
envoyé.
2026-01-12 11:26:07,819 - INFO - Tri des données par numéro de
```

```
page...
2026-01-12 11:26:08,007 - INFO - Succès ! 6613 éléments
sauvegardés dans l'ordre croissant.
```

- et un deuxième qui reprenait ce fichier json, qui prenait l'ID contenu dans l'entrée et qui refaisait la commande curl permettant d'avoir cette fois ci les mots de passe en clair dans le fichier json ce qui donnait quelque chose comme ça.

```
2026-01-12 14:25:51,766 - INFO - SUCCÈS : ID 20 | Nom: ''
2026-01-12 14:25:52,232 - INFO - SUCCÈS : ID 16 | Nom: ''
2026-01-12 14:25:52,255 - INFO - SUCCÈS : ID 17 | Nom: ''
2026-01-12 14:25:52,537 - INFO - SUCCÈS : ID 14 | Nom: ''
2026-01-12 14:25:52,614 - INFO - SUCCÈS : ID 13 | Nom: ''
2026-01-12 14:25:52,830 - INFO - SUCCÈS : ID 12 | Nom: ''
2026-01-12 14:25:53,574 - INFO - SUCCÈS : ID 9 | Nom: ''
2026-01-12 14:25:53,601 - INFO - SUCCÈS : ID 7 | Nom: ''
2026-01-12 14:25:53,649 - INFO - SUCCÈS : ID 11 | Nom: ''
2026-01-12 14:25:53,671 - INFO - SUCCÈS : ID 8 | Nom: ''
2026-01-12 14:25:53,838 - INFO - SUCCÈS : ID 1 | Nom: ''
2026-01-12 14:25:53,843 - INFO - Reconstruction du fichier final
selon l'ordre d'origine...
2026-01-12 14:25:54,389 - INFO - Terminé. Fichier généré :
C:\Users\INL090\Downloads\méthode_API\tpm_final_complet.json
```

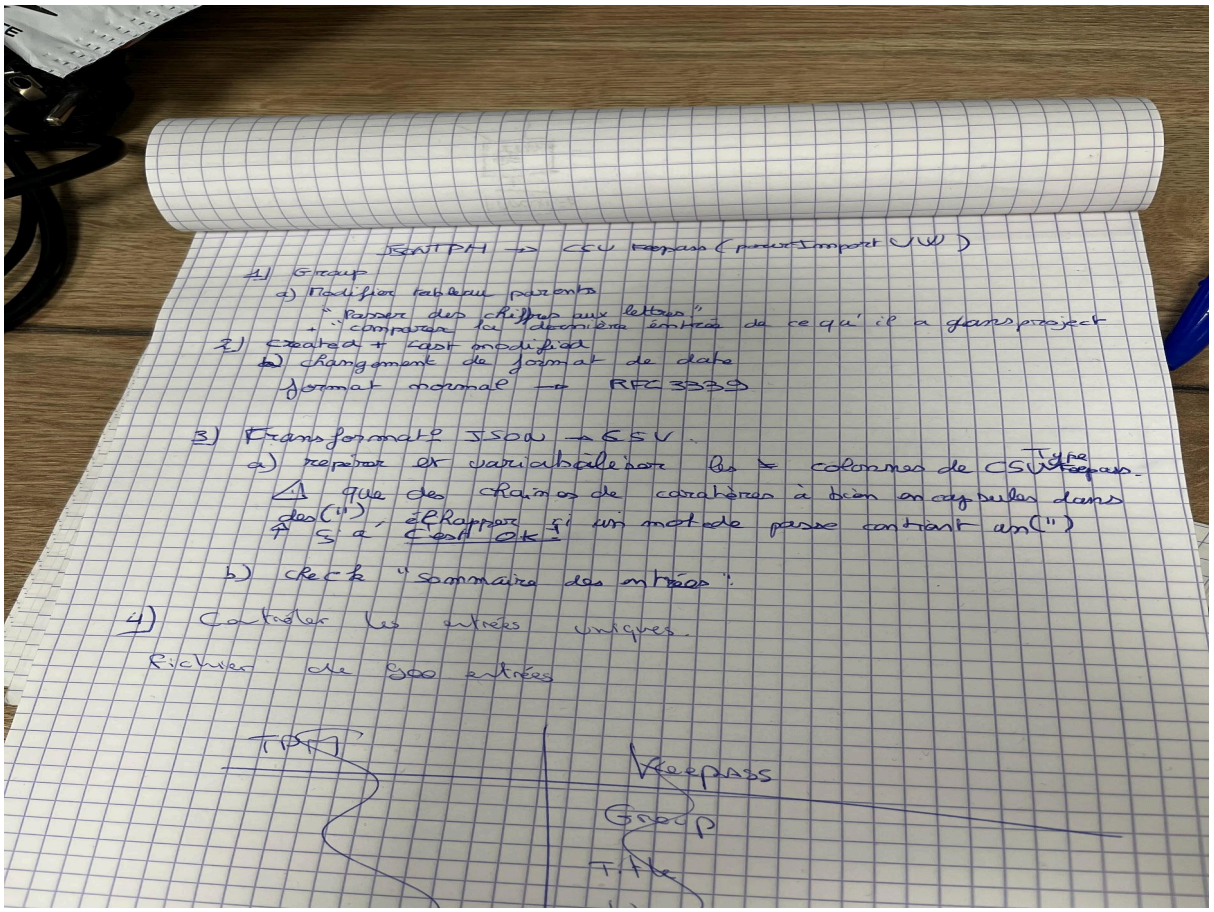
***les noms étant vides pour soucis de confidentialité**

Maintenant il faut comprendre comment déjà convertir les json TPM en json Vaultwarden car les champs sont complétements différents comme pour les CSV. Ensuite je me suis renseigné sur comment l'API de Vaultwarden fonctionne et en gros elle ne supporte pas l'API publique de Bitwarden sur laquelle on peut faire un simple **POST** avec une URL en endpoint comme pour l'export de TPM par exemple ou on faisait un curl avec une url bien précise, en fait il faut utiliser un client Bitwarden en CLI afin de déployer un serveur en local qui communique directement avec l'API de Vaultwarden et en faisant ça il y a une documentation des différents commandes que l'on peut faire pour communiquer avec l'API de Vaultwarden directement. Donc il faudra approfondir le sujet demain.

Mardi 13 janvier :

Après avoir approfondi le sujet de l'importation des mots de passe par la CLI de Bitwarden pour communiquer avec l'API de Vaultwarden nous avons laissé tomber l'idée car trop compliqué pour rien et ça représentait une perte de temps considérable. En amont nous avions déjà essayé d'importer des fichiers CSV au format KeepassXC ce qui marchait dans Vaultwarden. Du coup nous avons fait ça, c'était une dernière solution dont on pensait qu'elle allait marcher et elle a marché.

Donc nous sommes partis du fichier .json avec toutes les entrées de mot de passe et nous avons fait une checklist des choses que les différents scripts devaient faire.



On a d'abord commencé par faire un script qui prend évidemment toutes les entrées json dans le fichier :


```
    {
        "id": 8479,
        "name": "",
        "project": {
            "id": 1718,
            "name": ""
        },
        "tags": "",
        "access_info": "",
        "username": "",
        "email": "",
        "password": "",
        "expiry_date": null,
        "expiry_status": 0,
        "notes": "",
        "custom_field1": null,
```

```
        "custom_field2": null,
        "custom_field3": null,
        "custom_field4": null,
        "custom_field5": null,
        "custom_field6": null,
        "custom_field7": null,
        "custom_field8": null,
        "custom_field9": null,
        "custom_field10": null,
        "users_permissions": null,
        "groups_permissions": null,
        "parents": [
            1180,
            1217,
            1218,
            1718
        ],
        "user_permission": {
            "id": 30,
            "label": "Manage"
        },
        "archived": false,
        "project_archived": false,
        "favorite": false,
        "num_files": 0,
        "locked": false,
        "locking_type": 0,
        "locking_request_notify": 0,
        "external_sharing": false,
        "external_url": null,
        "linked": false,
        "source_password_id": 0,
        "managed_by": {
            "id": 2,
            "username": "",
            "email_address": "",
            "name": "",
            "role": ""
        },
        "created_on": "2024-08-14 09:47:09",
        "created_by": {
            "id": 2,
```

```
            "username": "",
            "email_address": "",
            "name": "",
            "role": ""
        },
        "updated_on": "2024-08-14 14:12:56",
        "updated_by": {
            "id": 2,
            "username": "",
            "email_address": "",
            "name": "",
            "role": ""
        }
    },
```


et qui devait modifier le tableau parents pour ne plus que ce soit des ID mais le nom qui correspondaient à ces ID pour qu'après on puisse les entrés dans les champs du fichier CSV de type Keepass. Ensuite premier problème, dans les sorties que j'avais certains ID n'ont pas de nom étant donnée qu'ils n'ont pas de mot de passe dans TPM, ils sont vides ce qui fait que leur ID n'est pas associez à un nom j'avais donc des sorties qui ressemblait à ça :


```
    {
        "id": 8954,
        "name": "",
        "project": {
            "id": 1459,
            "name": "FTP"
        },
        "tags": "",
        "access_info": "",
        "username": "",
        "email": "",
        "password": "",
        "expiry_date": null,
        "expiry_status": 0,
        "notes": "",
        "custom_field1": null,
        "custom_field2": null,
        "custom_field3": null,
        "custom_field4": null,
        "custom_field5": null,
        "custom_field6": null,
        "custom_field7": null,
```

```
        "custom_field8": null,
        "custom_field9": null,
        "custom_field10": null,
        "users_permissions": null,
        "groups_permissions": null,
        "parents": [
            "INNLOG - Commun",
            "TESFRI",
            "1218",
            "1458",
            "FTP"
        ],
        "user_permission": {
            "id": 30,
            "label": "Manage"
        },
        "archived": false,
        "project_archived": false,
        "favorite": false,
        "num_files": 0,
        "locked": false,
        "locking_type": 0,
        "locking_request_notify": 0,
        "external_sharing": false,
        "external_url": null,
        "linked": false,
        "source_password_id": 0,
        "managed_by": {
            "id": 2,
            "username": "",
            "email_address": "",
            "name": "",
            "role": ""
        },
        "created_on": "2025-12-01T14:43:23Z",
        "created_by": {
            "id": 2,
            "username": "",
            "email_address": "",
            "name": "",
            "role": ""
        },
```

```
        "updated_on": "2025-12-01T14:52:37Z",
        "updated_by": {
            "id": 2,
            "username": "",
            "email_address": "",
            "name": "",
            "role": ""
        }
    },
```


J’ai donc eu l’idée de récupérer tous les projets ainsi que leurs noms et ID dans un autre fichier .json et cette fois ci le script avait la condition de “si je ne trouve pas un nom correspondant à l’ID par exemple 1218 je vais voir dans l’autre fichier .json si je le trouve et si je le trouve je viens le remettre dans le fichier .json de base. Ce qui a marché j’avais donc après des entrées .json totalement compatible avec les champs d’un fichier CSV de type Keepass. Maintenant il fallait que le fichier .json se transforme en CSV dans un autre script python qui était prévu à cet effet et je savais exactement quoi demander, mon prompt ressemble donc à cela :

Alors maintenant il me faut un script pour passer du fichier json sortie.json la en fichier CSV au format Keepass, la structure d'un CSV keepass est la suivante :

"Group","Title","Username","Password","URL","Notes","TOTP","Icon","Last Modified","Created"

Les champs qui m'intéresse sont les suivants :
parents => Group
name => Title
username => Username
access_info => URL
notes => Notes
updated_on => Last Modified
created_on => Created

à gauche ce sont les champs du fichier json qui doivent être transférés dans les champs du fichier CSV keepass à droite.

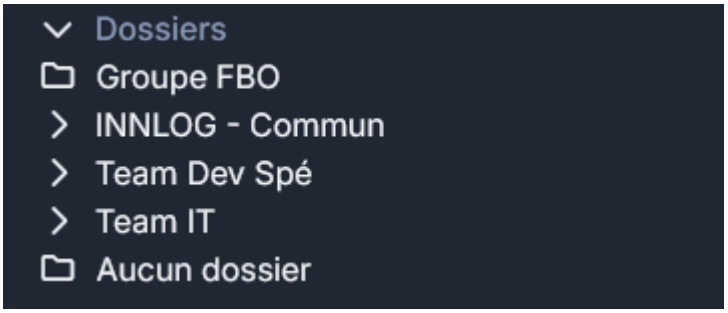
Petite subtilité ! Dans le tableau parent il peut y en avoir plusieurs. Exemple :

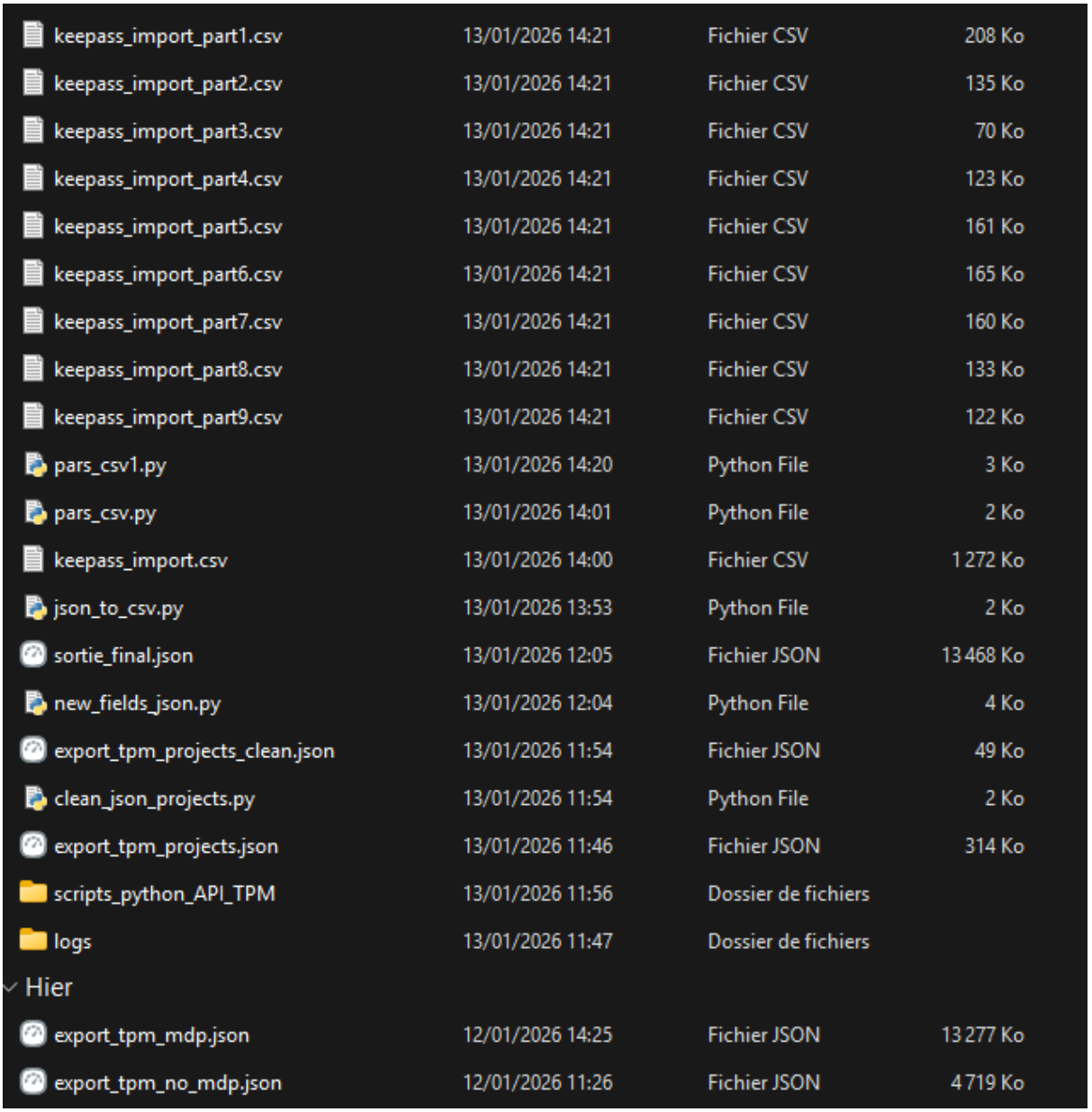
```
    "parents": [
      "INNLOG - Commun",
      "TESFRI",
      "Clients hébergés chez INNLOG",
      "Transports CRAS",
      "FTP"
    ],
```

il faut que dans le champs Group du fichier CSV keepass, il y a une hiérarchie de sorte à ce que le premier parents soit la racine et le dernier le dossier dans lequel se trouve l'entrée ce qui donnerait quelque chose comme ça pour te le tableau que je t'ai donné :

"INNLOG - Commun/TESFRI/Clients hébergés chez INNLOG/Transports CRAS/FTP c'est ça qui doit se trouver dans le champs Group du fichier CSV final keepass

et à partir de ce prompt Gemini m'a pondu un script parfait qui mettait les données dans les bon champs et qui faisait la hiérarchie dans le champ Group exactement comme je lui avait demandé. Ce qui faisait que j'avais enfin mon fichier CSV de type Keepass qui pouvait être importé dans Vaultwarden mais encore une fois ce n'était pas si simple. Vaultwarden a une sécurité anti-DoS qui fait que je ne peux pas balancer un fichier CSV avec des milliers de lignes à l'intérieur ce qui fait que j'ai coupé le fichier en 9 fichiers distincts que j'ai dû ensuite importer dans Vaultwarden. Ce qui fait que je me suis retrouvée avec tous ces fichiers et ces scripts à la fin de la matinée. Mais au moins ça a fonctionné et **tous les mots de passe étaient enfin importés dans Vaultwarden**. Ce qui donne quelque chose comme ça :

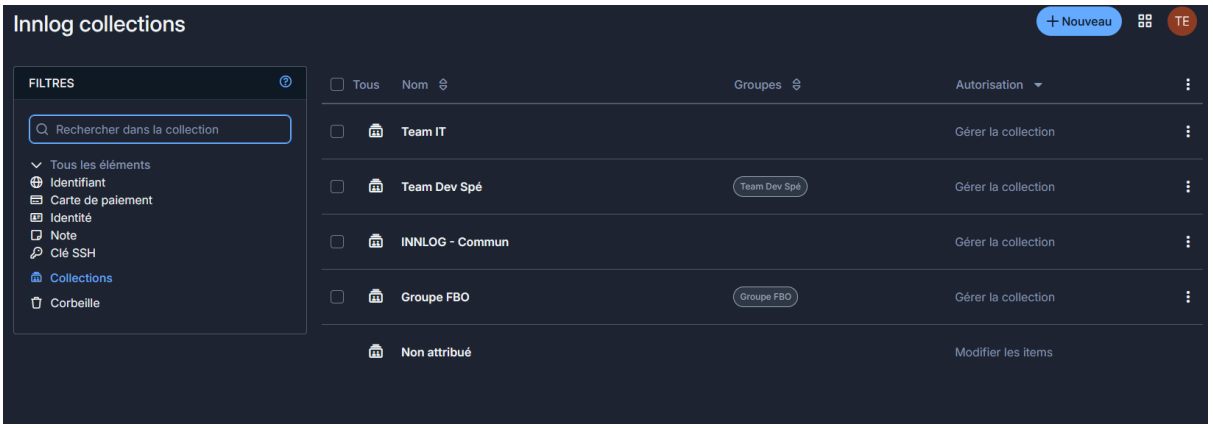




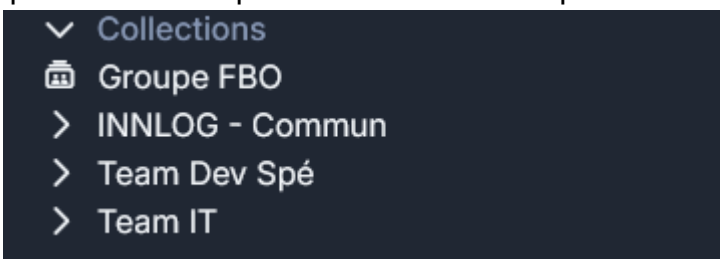
Après la migration de tous les mots de passe il a fallu trouver un moyen de créer et gérer des utilisateurs, n'ayant pas de fonctionnalités de liaison LDAP, nous devions créer une organisation cependant, la fonctionnalités front-end de création d'organisation ne marchant pas, nous avons été voir sur les releases github pour voir et le bug a été corrigé il y a 4 jours mais vaultwarden a été installé avec APT et le package d'où avait été installé vaultwarden n'avait pas tenu à jour cette fonctionnalité du coup, nous sommes restés sur la version 1.35.1 de vaultwarden et la fonctionnalités fonctionne à partir de la 1.35.2.

Maxime étant familier avec le rust (langage avec lequel est écrit vaultwarden), il a recompiler vaultwarden avec le bon front-end qui fait que la fonctionnalité de création d'organisation fonctionne. Cependant quand il me redonne le binaire ça ne fonctionne pas.

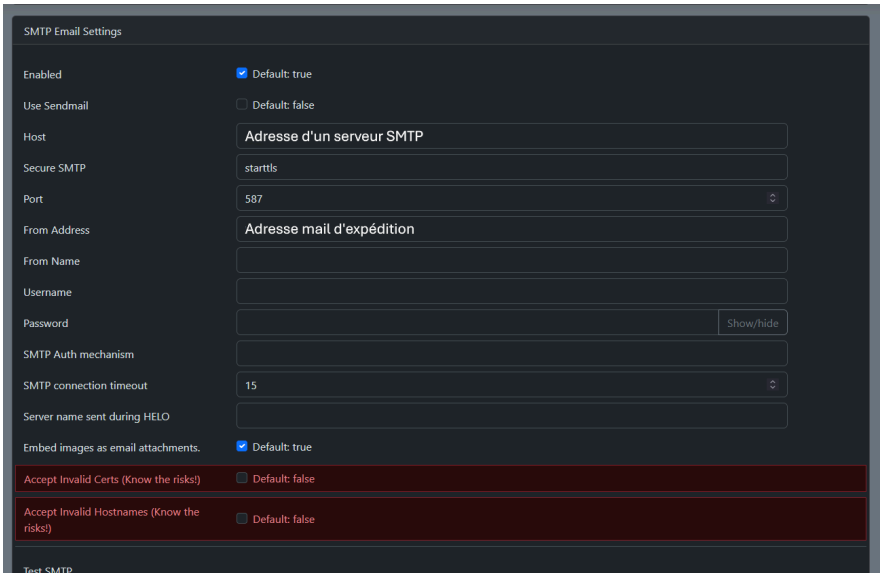
Mercredi 14 janvier :
Le création de l'organisation a marché, Maxime a rétrogradé la version Vaultwarden **serveur** pour que la version web du front-end prenne en compte la création de l'organisation donc j'ai pu créer l'organisation **Innlog** qui va ensuite nous permettre d'inviter des utilisateurs via leur adresse mail @innlog.fr et qu'ils pourront avoir accès aux mots de passe du Vaultwarden.



Par contre, il y a une nuance entre les "dossiers" et les "collections", les dossiers représentent le coffre personnel et les collections sont les dossiers de l'organisation ce qui fait que les dossiers n'ont pas la même structure CSV que les collections de l'organisation, ce qui ouvre un autre problème, je ne pouvais pas simplement exporté en csv ou un json les dossiers du coffre personnel car les champs ne correspondait pas, j'ai du donc refaire un script python qui modifie les champs pour qu'ils soient compatibles avec les champs des collections de l'organisation.



Après ça, il fallait trouver un moyen d'envoyer des mails aux personnes qu'ont invitent donc il fallait configurer le serveur SMTP avec Vaultwarden.

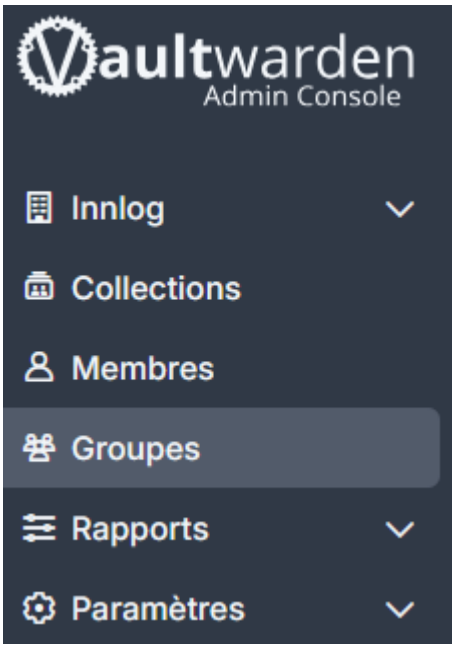


Les mails pouvant être envoyés de cette manière, il faudra trouver un moyen d'importer massivement des utilisateurs plus tard.

Un autre problème s'était présenté dans la fin de la matinée, en fait on se demandait si on ne pouvait pas faire de groupe pour gérer les permissions car les permissions dans Vaultwarden son bizarrement gérer donc on aurait voulu faire des groupes, dans la doc, il est question d'une fonctionnalité de groupe en bêta qu'il faut activer dans le fichier des variables d'environnement.

```
## BETA FEATURE: Groups
## Controls whether group support is enabled for organizations
## This setting applies to organizations.
## Disabled by default because this is a beta feature, it contains known issues!
## KNOW WHAT YOU ARE DOING!
ORG_GROUPS_ENABLED=true
```

Ce qui a marché et nous a fait apparaître la rubrique Groupes dans l'interface web



J'ai assigné les collections aux groupes manuellement.

Ensuite il fallait trouver un moyen d'importer massivement des utilisateurs par invitation via leurs emails.

Dans Vaultwarden il y a une page d'administration ou l'on peut inviter des personnes via leurs emails à rejoindre l'organisation mais on ne peut en inviter qu'un seul à la fois. Le groupe Tesson comptant beaucoup d'employés en tout cas trop pour tous les envoyés un mail manuellement un par un, j'ai eu l'idée de concaténer plusieurs adresses mails à la suite avec un séparateur , ou {} par exemple ce qui n'a pas marché, j'ai donc demander à Gemini s'il y avait une syntaxe particulière à mettre entre deux adresse mails pour qu'elles soient prise en compte et en même temps j'ai eu l'idée pour tester de lui envoyer le <input> en html pour voir ce qu'il allait en faire :

```
<input id="emails" type="text" appautofocus="" bitinput=""
formcontrolname="emails" class="ng-pristine ng-invalid
[&amp;:is(input,textarea):disabled]:tw-bg-secondary-100
focus:tw-outline-none tw-bg-background tw-block tw-border-none
tw-h-full tw-placeholder-text-muted tw-px-1 tw-text-main tw-w-full
ng-touched" required="" aria-describedby="bit-error-0"
aria-invalid="true">
```

et le bouton pour ouvrir la fenêtre qui permet d'entrer un email car c'est un modal js (une fenêtre pop-up)

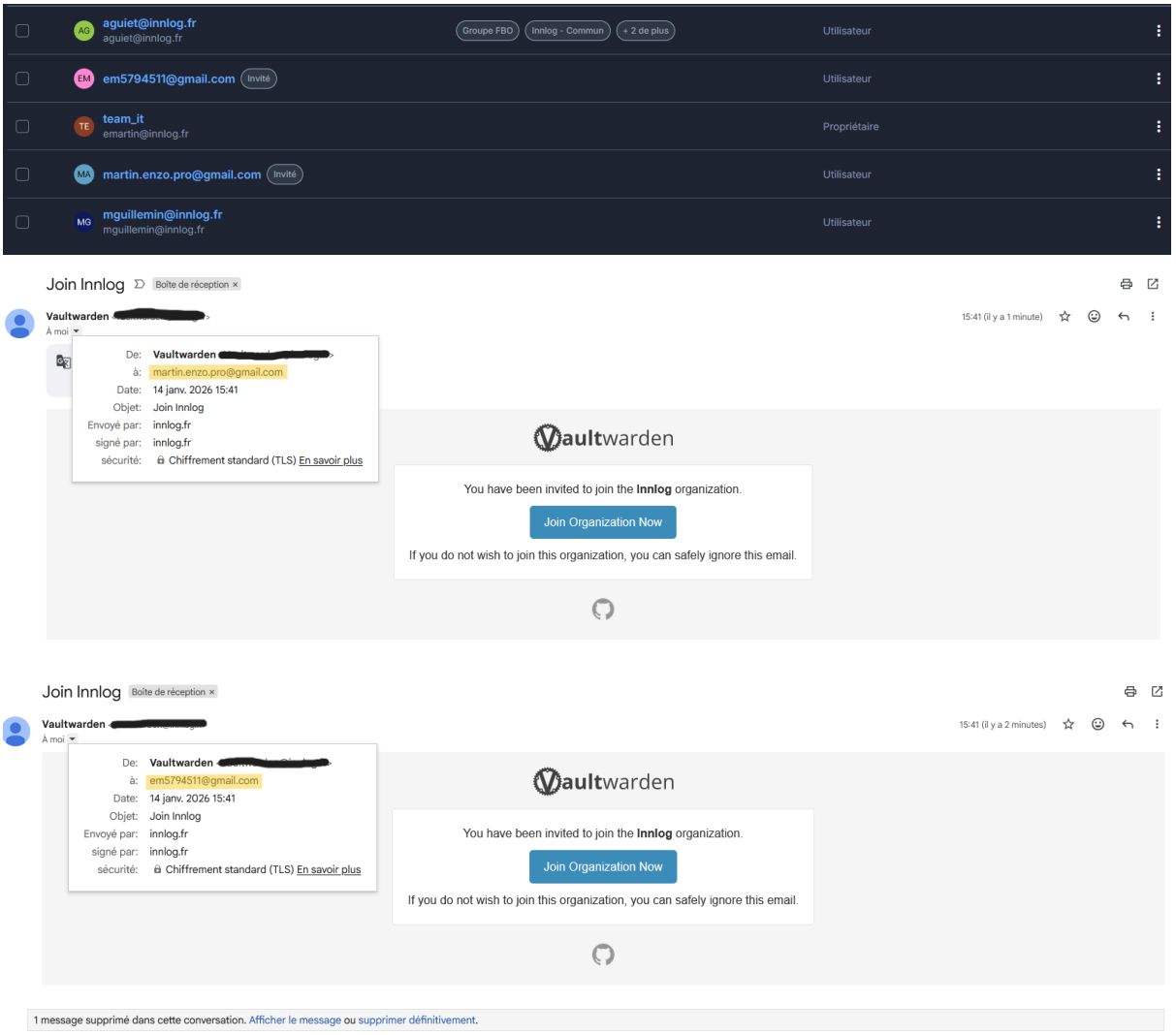
```
<button type="button" bitbutton="" buttontype="primary"
bit-aria-disable="true" class="!tw-text-contrast
focus-visible:tw-ring-2 focus-visible:tw-ring-offset-2
focus-visible:tw-ring-primary-600 focus-visible:tw-z-10
focus:tw-outline-none hover:tw-bg-primary-700
hover:tw-border-primary-700 hover:tw-no-underline
tw-bg-primary-600 tw-border-2 tw-border-primary-600
tw-border-solid tw-font-medium tw-inline-block tw-no-underline
tw-px-3 tw-py-1.5 tw-rounded-full tw-text-center tw-transition
ng-star-inserted"><span class="tw-relative"><span><i
aria-hidden="true" class="bwi bwi-plus bwi-fw"></i> Inviter un
membre </span><!----></span></button>
```

et il m’a pondu un morceau de JavaScript directement à balancer dans la console quand on fait F12 pour automatiser ça

```
{
    const emails = [
        "test1@gmail.com",
        "test2@gmail.com"
        // Ajoutez vos centaines d'emails ici
    ];

    const delay = (ms) => new Promise(resolve => setTimeout(resolve, ms));

    async function processAll() {
        for (const email of emails) {
            console.log("--- Début du processus pour : " + email);

            // 1. Trouver et cliquer sur le bouton "Inviter un membre"
            const openModalBtn =
Array.from(document.querySelectorAll('button'))
                                        .find(el =>
el.textContent.includes("Inviter un membre"));

            if (!openModalBtn) {
                console.error("Impossible de trouver le bouton 'Inviter un
membre'");
                break;
            }
            openModalBtn.click();

            // 2. Attendre que la modale s'ouvre (1 seconde)
            await delay(1000);

            // 3. Trouver l'input et le remplir
            const emailInput = document.getElementById('emails');
            if (emailInput) {
                emailInput.value = email;
                emailInput.dispatchEvent(new Event('input', { bubbles: true
}));
                emailInput.dispatchEvent(new Event('change', { bubbles: true
}));
                console.log("Champ rempli.");
```

```
            } else {
                console.error("Champ email introuvable dans la modale.");
                continue;
            }

            // 4. Trouver le bouton "Envoyer" dans la modale et cliquer
            // On cherche un bouton de type submit OU un bouton qui contient
"Envoyer" ou "Inviter"
            const submitBtn = document.querySelector('button[type="submit"]')
||

Array.from(document.querySelectorAll('button')).find(el =>
el.innerText.includes("Inviter"));

            if (submitBtn) {
                submitBtn.click();
                console.log("Bouton d'envoi cliqué.");
            }

            // 5. Attendre que la modale se ferme et que le site enregistre
(2.5 secondes)
            // Augmentez ce temps si votre site est lent à réagir
            await delay(2500);
        }
        alert("Toutes les invitations ont été traitées !");
    }

    processAll();
}
```

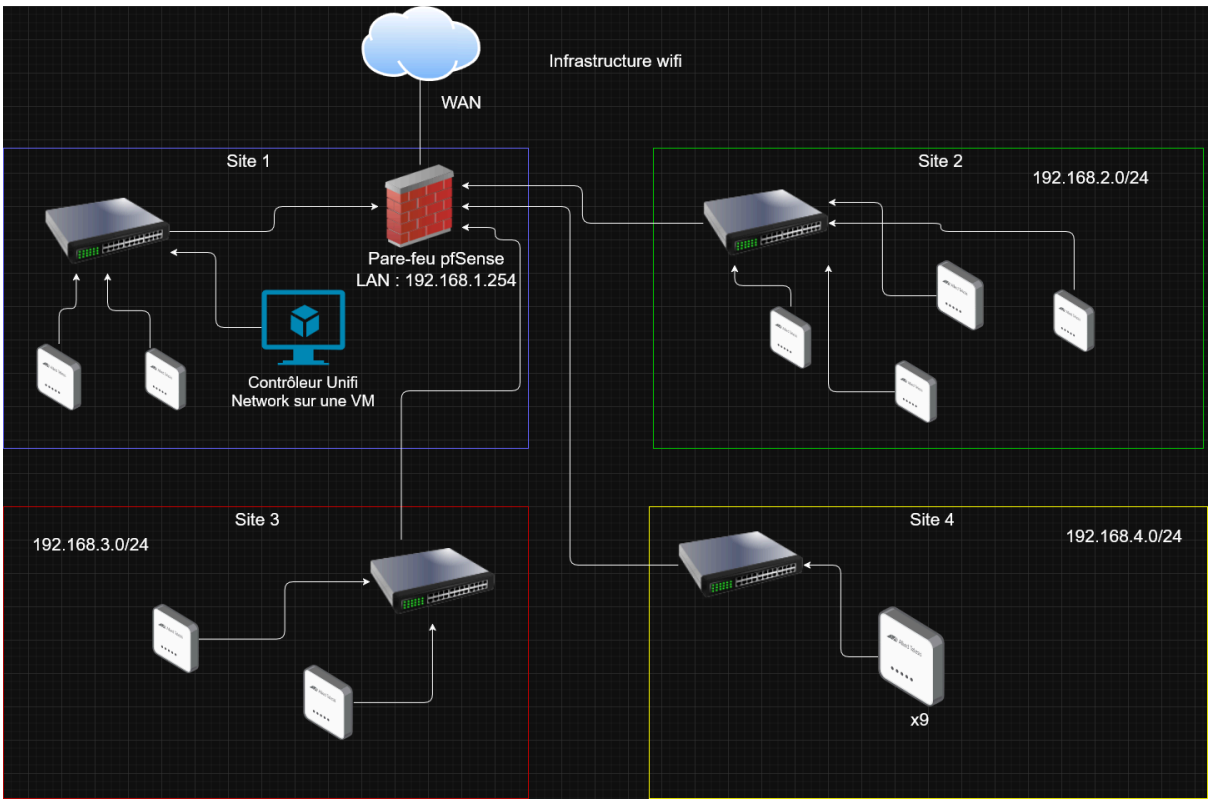
```
{
    const emails = [
        "martin.enzo.pro@gmail.com",
        "em5794511@gmail.com"
        // Ajoutez vos centaines d'emails ici…
--- Début du processus pour : martin.enzo.pro@gmail.com
Promise { <state>: "pending" }
Champ rempli.
Bouton d'envoi cliqué.
--- Début du processus pour : em5794511@gmail.com
Champ rempli.
Bouton d'envoi cliqué.
```



Donc au stade ou j'en suis rendu, mon responsable, Anthony a proposé qu'on soit en phase de test pendant une petite semaine afin de tester le produit et se familiariser avec l'interface et qu'en suite on verra pour le mettre en prod.

Jeudi 15 janvier :

Arrivé le jeudi et la migration de Vaultwarden étant finie, le responsable m'a donné une autre mission, refaire l'infrastructure wifi d'un de leur clients, en effet ce groupe qui fait de l'entreposage et du transport frigorifique avait une infrastructure wifi très vieille et obsolète qui diffusait encore sous le protocole WEP, il fallait donc changer l'infrastructure car les nouveaux terminaux qu'ils avaient n'étaient plus compatible avec cette technologie, avec l'aide d'Antoine on a donc réfléchi à une infrastructure pour les 4 sites que possède l'entreprise. La mission pour l'instant est de mettre en place une sorte de LAB dans le bureau avec les switch cisco et les bornes wifi Unifi qui seront mis en place sur les sites.



***fausse adresses IP et sites anonymiser**

ça c'est le schéma que j'ai fait à l'aide de draw.io qui représente le lab élaborer dans le bureau, dans la réalité chaque site aura son pare-feu pfSense attribuer.

Vendredi 16 janvier :

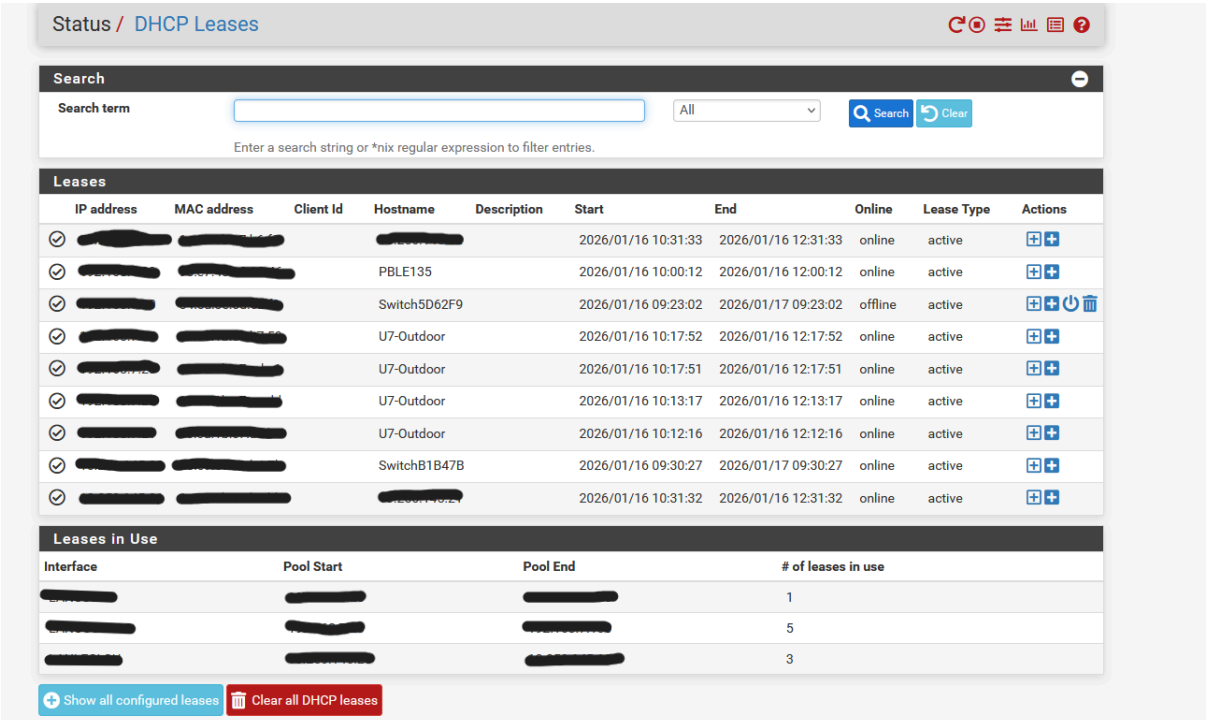
Arrivé le vendredi j'ai commencé à appliquer le schéma que j'avais fait en amont en commençant par les bornes du premier site qui était les plus simples à mettre en place étant donné qu'elles sont dans le même réseau que le pare-feu.

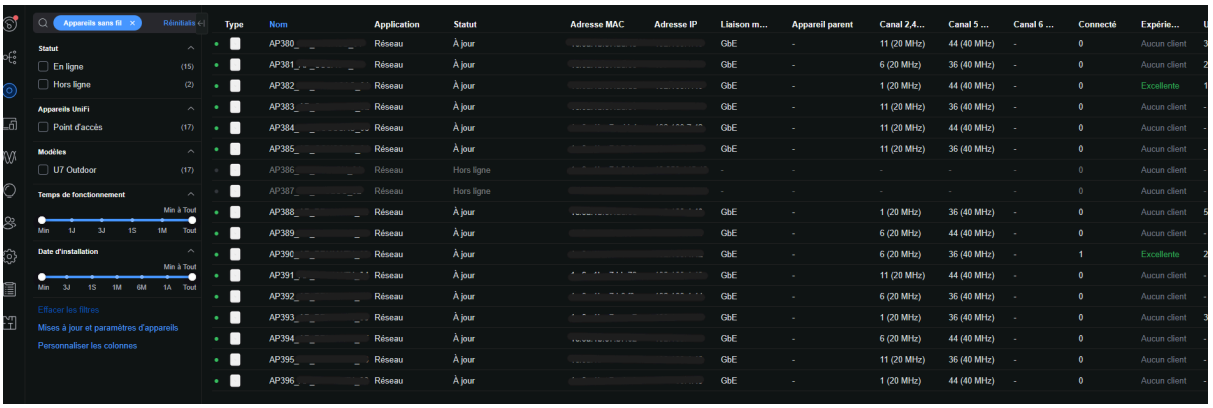
Pour les bornes n'étant pas dans le même réseau que le contrôleur Unifi il faut se connecter en SSH sur les bornes et taper cette commande

set-inform http://<ip_du_contrôleur>/inform

Je récupère les adresses IP des bornes dans les bails que distribue le pfSense

Et j'ai évidemment renommé les switch cisco correctement.





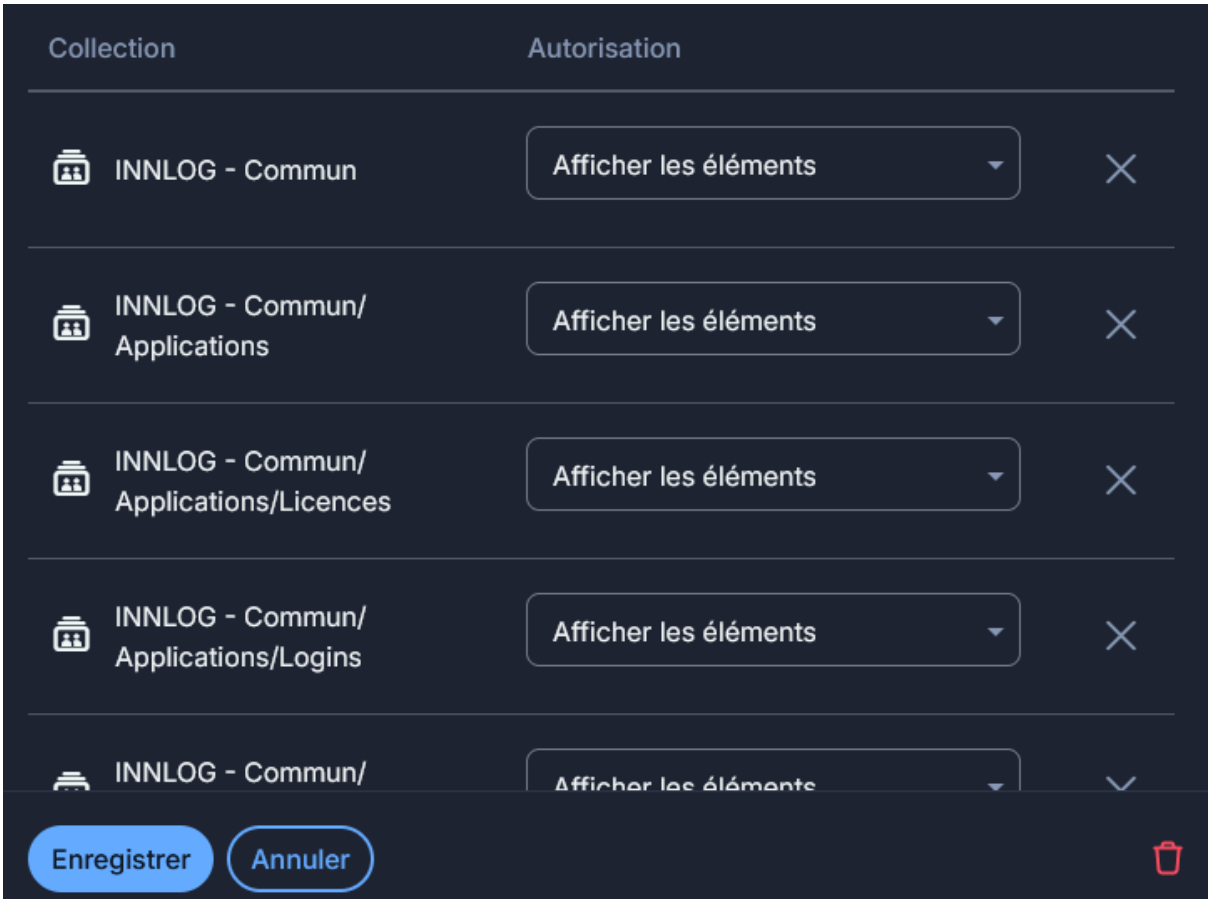
Ça donne ça à la fin, deux bornes étant hors ligne car le pfSense n’a pas assez de ports pour connecter tous les switchs.


cisco
UniFi

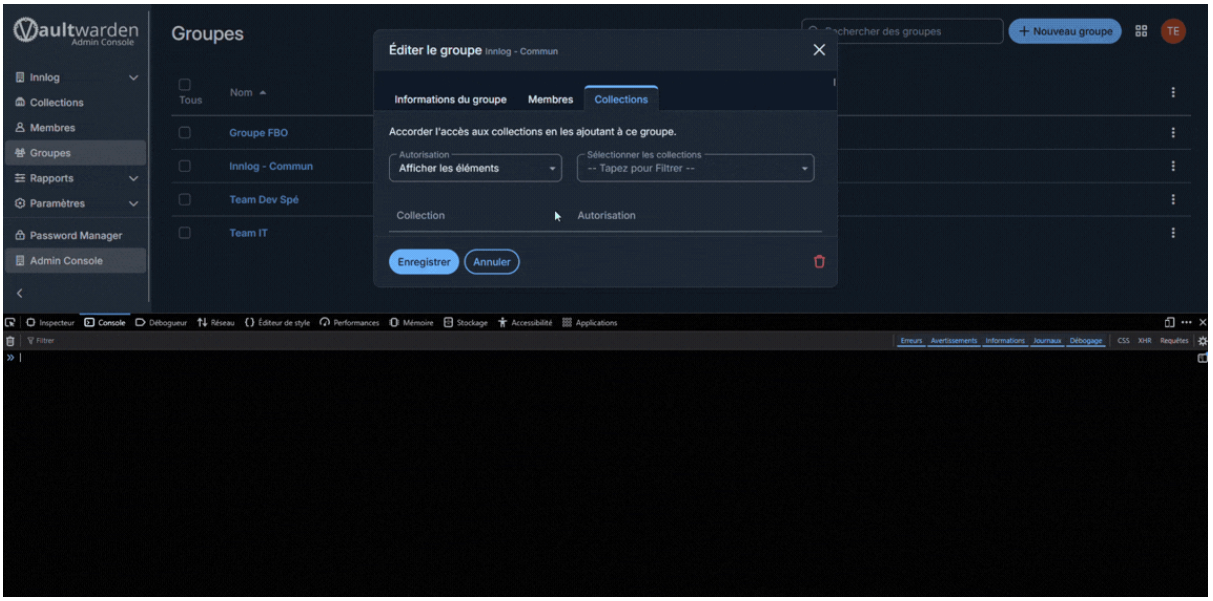

UniFi
U7 Outdoor
U7 Outdoor
UniFi
U7 Outdoor
U7 Outdoor
UniFi
U7 Outdoor
WAN
LAN

Compte rendu hebdomadaire troisième semaine

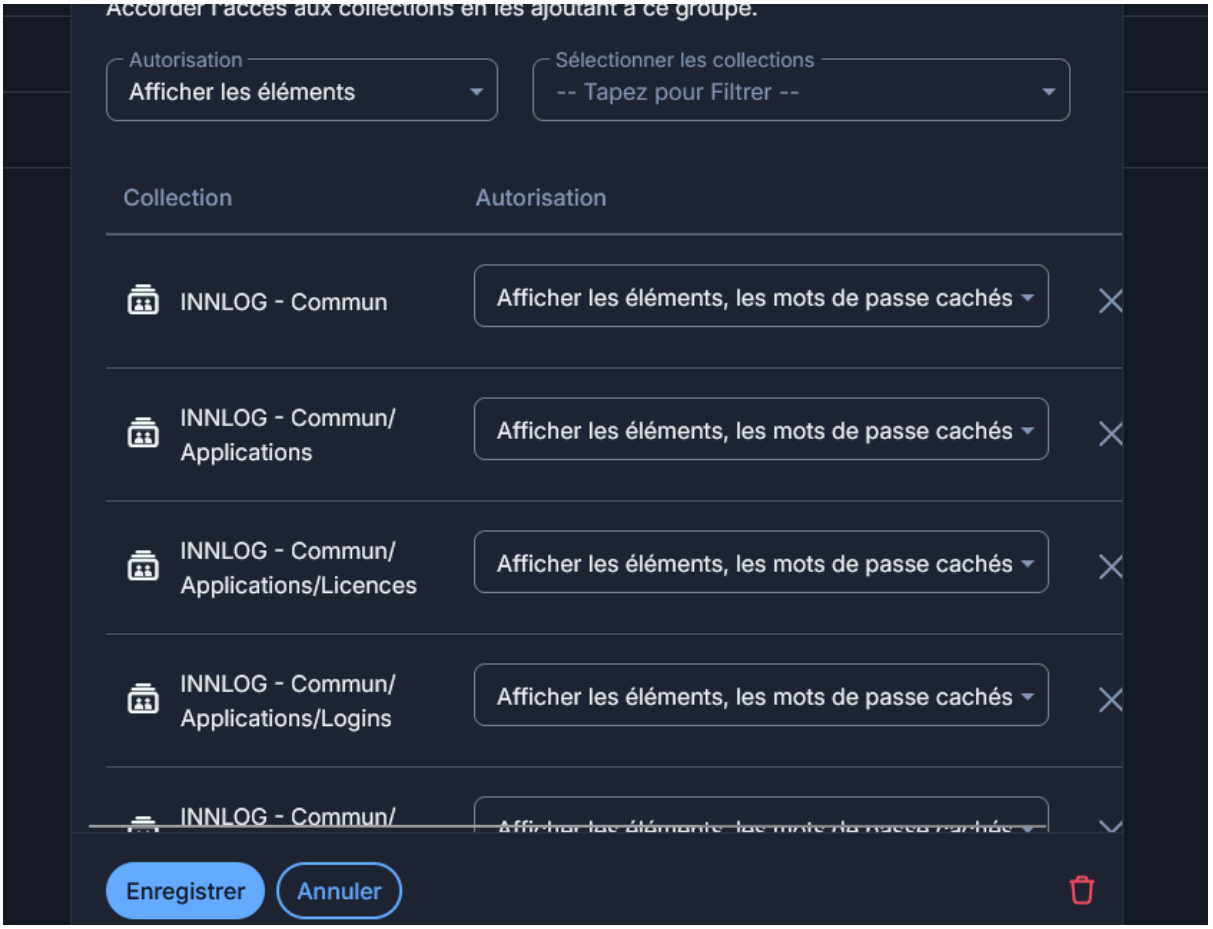
Lundi 19 janvier :
Ce lundi 19 janvier, Anthony m'a soumis un problème dans Vaultwarden car comme je l'ai précisé précédemment nous sommes en phase de test de ce nouvel outil, en effet j'ai affecté Anthony en tant qu'utilisateur avec l'attribut "Afficher les éléments"



Hors quand il va sur une entrée de mot de passe, il ne voit pas le champ site web ce qui sera gênant sur le long terme si l'on installe l'extension d'auto-complétion des mots de passe sur les sites web. Et j'ai découvert qu'il fallait que les autorisations soient en "Afficher les éléments, les mots de passe cachés" donc il fallait changer tous ces champs à la main, sachant qu'il y a plus de 700 champs dans cette base de données, j'allais forcément automatiser la chose.



Après cette modification automatisée, les permissions ressemblent maintenant à ça



Ce qui est bien, maintenant Anthony peut voir les sites web dans les entrées de mots de passe.

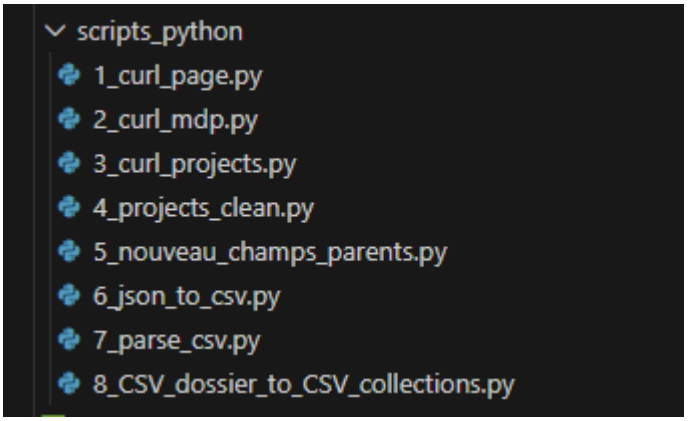
Mardi 20 janvier :
Aujourd'hui je n'avais plus grand chose à faire je me suis penché sur la fonctionnalité de HeatMap que propose le logiciel Unifi Network et qui permet d'importer les plans du bâtiments où se trouvent les bornes et de pouvoir les placer pour voir la couverture qu'elles ont. J'ai donc demandé à Antoine d'envoyer un mail au client en question pour avoir les plans des 4 sites et je les ai eu mais les plans n'étaient pas clair, il n'y avait pas de mesures et je ne savais la composition des murs ce qui est essentiel pour utiliser la HeatMap, ça s'est donc soldé par un échec.

J'ai donc avancé sur mon portfolio afin de commencer mon article de stage en amont.

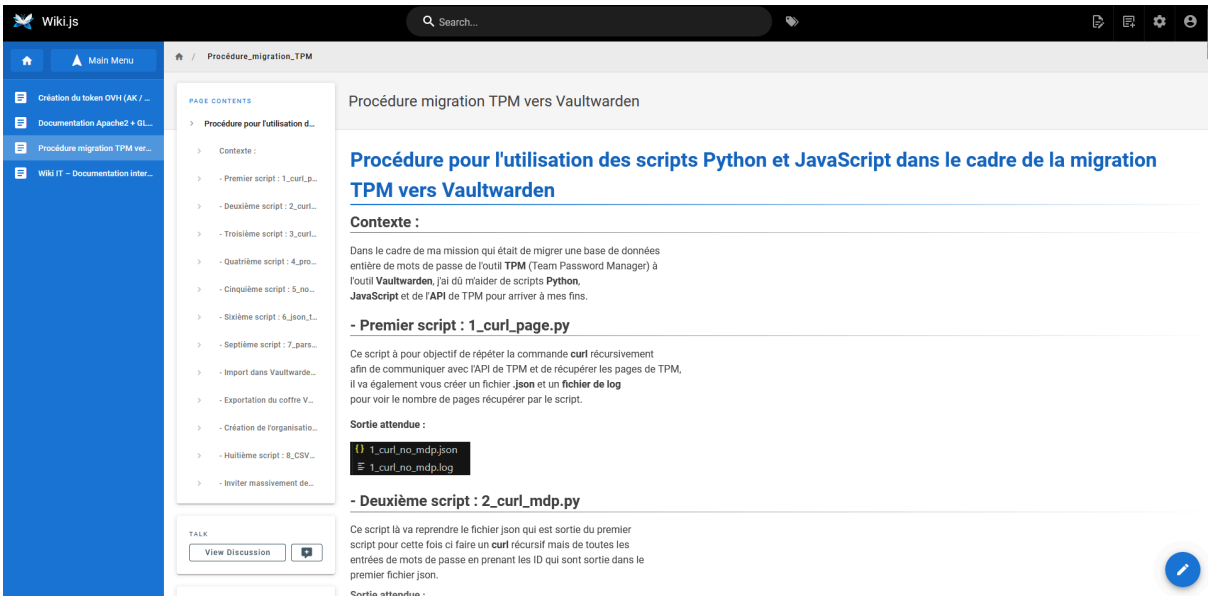
***image d'illustration**

Mercredi 21 janvier :

Ce matin j'ai eu l'idée de faire une documentation sur l'utilisation des scripts Python et JavaScript pour la migration de TPM vers Vaultwarden car il faudra en refaire étant donné que le Vaultwarden n'est pas encore en production les employés utilisent encore TPM et crée encore des mots de passe dans TPM donc il faudra faire une migration finale vers Vaultwarden le jour de la mise en production, j'ai donc trié les scripts Python pour que ce soit plus lisible.

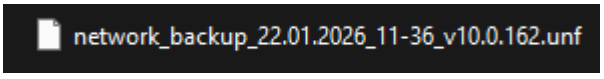


J'ai passé la fin de ma matinée et une bonne partie de mon après-midi à faire la documentation sur l'utilisation des scripts dans le cadre de la migration et l'entreprise utilise Wiki JS pour auto-héberger leurs procédures donc je l'ai fais en .md (Markdown) et je l'ai mise dans leur Wiki.



Jeudi 22 janvier :
Je suis revenu sur les plans du client pour les importer dans la HeatMap avec Antoine qui a renvoyé un mail au client pour avoir des meilleurs plans d'un de leur site ce qu'on a réussi à avoir j'ai donc pu l'importer dans le logiciel Unifi pour pouvoir placer les bornes wifi et avoir un aperçu de la couverture wifi.

Ensuite j'ai exporté le fichier de configuration pour l'envoyer au client car eux vont mettre en place une machine virtuelle sous Debian (moi je l'ai fais sur Ubuntu), il n'auront plus qu'à installer le contrôleur avec le script d'installation et de réimporter le fichier de configuration avec toutes les bornes (leurs noms, leurs IP etc)



Fin d'après midi nous avons remballer tout le matériel du client pour leur envoyer afin qu'ils puissent tout installer sur leurs différents sites avec les config des bornes et du contrôleur, les plans des bornes etc.

Vendredi 23 janvier :
Je suis arrivé le matin et Maxime voyait bien que je n'avais plus grand chose à faire il m'a donc donné pour mission de faire un serveur **GRAYLOG** pour centraliser les logs des machines. J'ai donc commencé à m'intéresser au sujet en parcourant la doc officielle et en commençant les manipulations.

J'ai d'abord commencé par créer une VM dans proxmox et j'ai commencé à installer les premiers paquets mais j'ai vite rencontré un problème, comme moteur de base de données GRAYLOG utilise mongodb, hors la VM que j'avais créé était sous trixie (debian 13) et le dernier repo de mongodb est pour bookworm (debian 12), j'ai donc refais une VM sous debian 12. De plus, les hyperviseurs que l'on utilise on est processeur sous architecture **Nehalem** et c'est une architecture qui n'est pas capable de traiter des instructions **AVX** et donc MongoDB 5.0 et version supérieure ne sont pas compatible avec ce genre de processeur, la seule version compatible est la 4.4. J'ai essayé la méthode Docker mais l'architecture des processeurs physique bloque vraiment le processus c'est donc un échec et maxime va me mettre sur autre chose.

Il m'a donc proposé de déployer une stack **Grafana, Alloy et Loki.**

GRAFANA :

sudo apt-get install -y apt-transport-https wget

sudo apt-get install gpg

sudo mkdir -p /etc/apt/keyrings/
wget -q -O - https://apt.grafana.com/gpg.key | gpg --dearmor | sudo tee /etc/apt/keyrings/grafana.gpg > /dev/null

echo "deb [signed-by=/etc/apt/keyrings/grafana.gpg] https://apt.grafana.com stable main" | sudo tee -a /etc/apt/sources.list.d/grafana.list

sudo apt-get update

sudo apt-get install grafana

http://<ip>:3000

ALLOY :

sudo wget -O /etc/apt/keyrings/grafana.asc https://apt.grafana.com/gpg-full.key

chmod 644 /etc/apt/keyrings/grafana.asc

echo "deb [signed-by=/etc/apt/keyrings/grafana.asc] https://apt.grafana.com stable main" | sudo tee /etc/apt/sources.list.d/grafana.list

sudo apt-get update
sudo apt-get install alloy

sudo systemctl start alloy

sudo systemctl status alloy

sudo systemctl enable alloy.service

LOKI :

sudo apt install loki promtail

sudo systemctl status loki

Détails de la stack :

Grafana Alloy : Grafana Alloy est le collecteur de télémétrie qui récupère les logs mais aussi les métriques qu'on lui fournit.

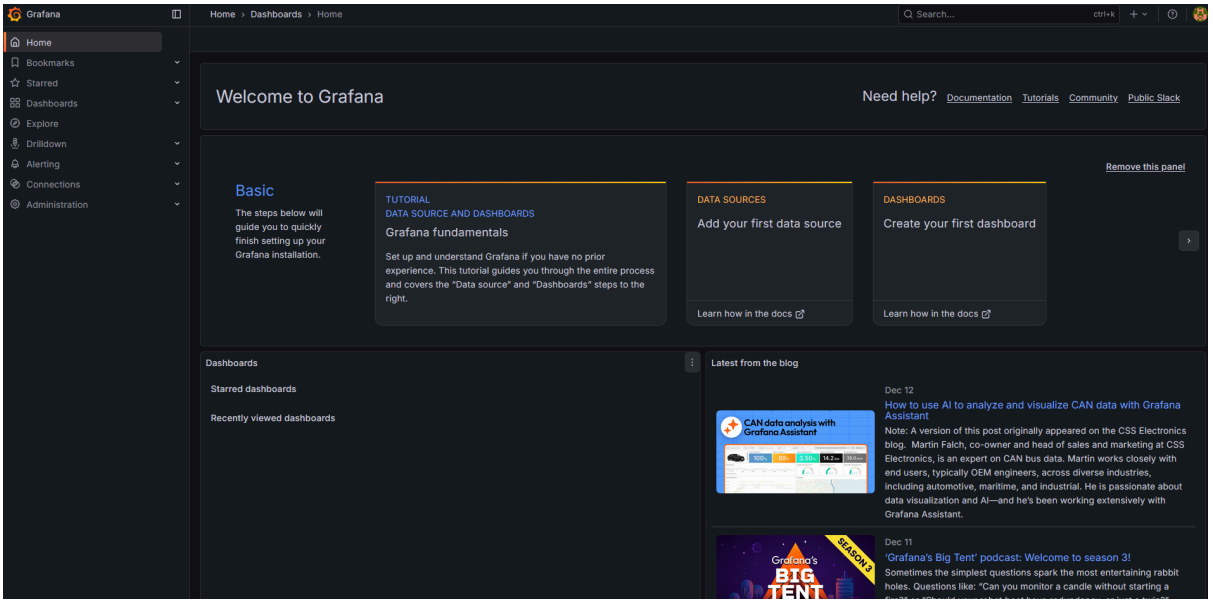
Grafana Loki : Grafana Loki a pour travail de recevoir les logs envoyer par Alloy, c'est la que les logs sont stockés

Grafana : Grafana en tant que tel est l'interface de toutes les données recueillies précédemment.

Il faut installer tous les éléments de la stack individuellement pour ensuite les faire communiquer.

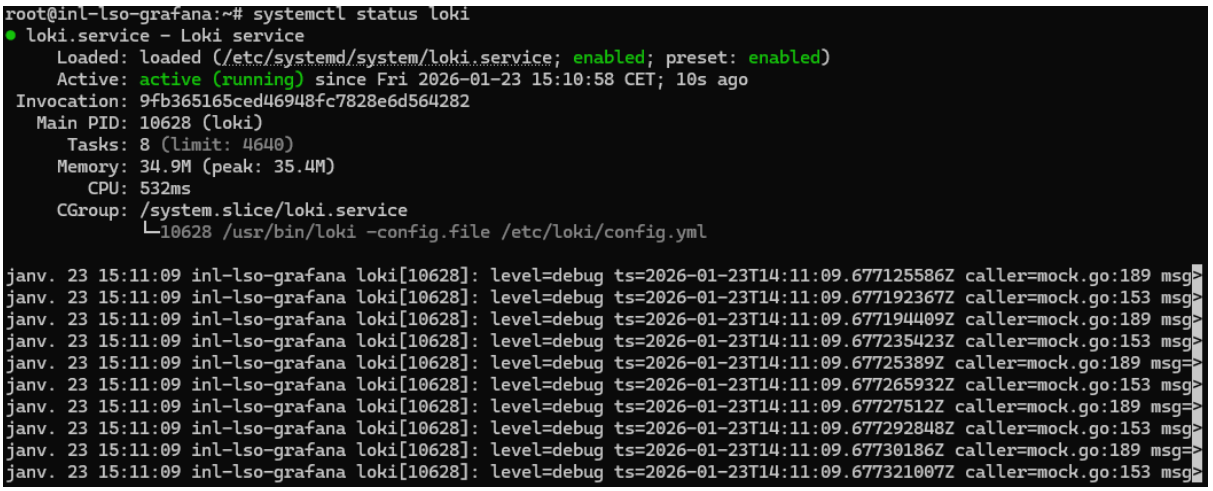
Tout d'abord j'ai installé Grafana, l'interface via le gestionnaire de paquets

```
root@inl-lso-grafana:~# systemctl restart alloy
root@inl-lso-grafana:~# systemctl status grafana-server.service
● grafana-server.service - Grafana instance
     Loaded: loaded (/usr/lib/systemd/system/grafana-server.service; enabled; preset: enabled)
     Active: active (running) since Fri 2026-01-23 14:39:09 CET; 14min ago
 Invocation: 1d7095a19f7f401cb5e9952af07d8d7b
       Docs: http://docs.grafana.org
   Main PID: 9883 (grafana)
      Tasks: 15 (limit: 4640)
     Memory: 144.9M (peak: 169.3M)
        CPU: 17.074s
     CGroup: /system.slice/grafana-server.service
             └─9883 /usr/share/grafana/bin/grafana server --config=/etc/grafana/grafana.ini --pidfile=/run/grafana/>

janv. 23 14:39:25 inl-lso-grafana grafana[9883]: logger=plugin.installer t=2026-01-23T14:39:25.930144139+01:00 leve>
janv. 23 14:39:26 inl-lso-grafana grafana[9883]: logger=installer.fs t=2026-01-23T14:39:26.232621015+01:00 level=in>
janv. 23 14:39:26 inl-lso-grafana grafana[9883]: logger=plugins.registration t=2026-01-23T14:39:26.511363025+01:00 >
janv. 23 14:39:26 inl-lso-grafana grafana[9883]: logger=plugin.backgroundinstaller t=2026-01-23T14:39:26.511511494+>
janv. 23 14:39:45 inl-lso-grafana grafana[9883]: logger=context userId=0 orgId=0 uname= t=2026-01-23T14:39:45.19775>
janv. 23 14:40:21 inl-lso-grafana grafana[9883]: logger=infra.usagestats t=2026-01-23T14:40:21.234806941+01:00 leve>
janv. 23 14:42:38 inl-lso-grafana grafana[9883]: logger=context userId=1 orgId=1 uname=admin t=2026-01-23T14:42:38.>
janv. 23 14:49:23 inl-lso-grafana grafana[9883]: logger=cleanup t=2026-01-23T14:49:23.217421982+01:00 level=info ms>
janv. 23 14:49:23 inl-lso-grafana grafana[9883]: logger=sqlstore.transactions t=2026-01-23T14:49:23.275992432+01:00>
janv. 23 14:49:23 inl-lso-grafana grafana[9883]: logger=plugins.update.checker t=2026-01-23T14:49:23.707776145+01:0>
```



Il faut maintenant installer **Alloy** qui va récupérer les logs et les métriques de la machine hôte pour l'instant et voir comment le faire communiquer avec l'interface de Grafana.

```
root@inl-lso-grafana:~# systemctl status alloy
● alloy.service - Vendor-agnostic OpenTelemetry Collector distribution with programmable pipelines
     Loaded: loaded (/usr/lib/systemd/system/alloy.service; enabled; preset: enabled)
     Active: active (running) since Fri 2026-01-23 14:53:05 CET; 1min 18s ago
 Invocation: 7f18afe9b1744f57aaee01fddd510a44
       Docs: https://grafana.com/docs/alloy
   Main PID: 10071 (alloy)
      Tasks: 8 (limit: 4640)
     Memory: 36.2M (peak: 36.7M)
        CPU: 572ms
     CGroup: /system.slice/alloy.service
             └─10071 /usr/bin/alloy run --storage.path=/var/lib/alloy/data /etc/alloy/config.alloy

janv. 23 14:53:05 inl-lso-grafana systemd[1]: Started alloy.service - Vendor-agnostic OpenTelemetry Collector distr>
```



Maintenant il fallait faire communiquer Alloy avec Loki pour que les logs et les métriques que récupère Alloy soient stockés dans Loki et que Loki puisse devenir une source de données pour Grafana.

le fichier de configuration d'Alloy ressemble à ça :

```
logging {
  level = "info"
}

/* ===== JOURNALD ===== */
loki.source.journal "systemd" {
  forward_to = [loki.write.default.receiver]

  labels = {
    job  = "journald",
    host = "<ip de l'hôte>",
  }
}

/* ===== DESTINATION LOKI ===== */
loki.write "default" {
  endpoint {
    url = "http://127.0.0.1:3100/loki/api/v1/push"
  }
}
```

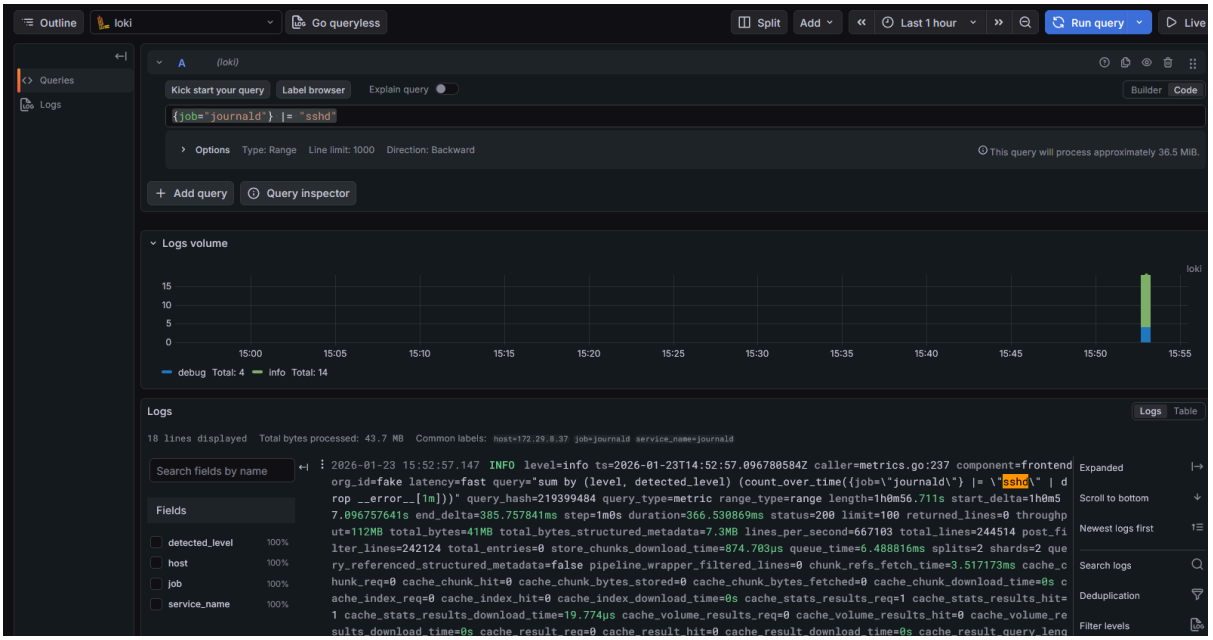
Ensuite j'ai commencé à comprendre comment tout cela fonctionnait et que le fichier de configuration Alloy est très personnalisable par exemple ajouter un bloc pour les logs SSH :

```
loki.relabel "journal_filter" {
  forward_to = [loki.write.default.receiver]
```

```
  // Si l'unité systemd est ssh.service, on change le label job en "ssh"
  rule {
    source_labels = ["__journal__systemd_unit"]
    regex         = "ssh.service"
    target_label  = "job"
    replacement   = "ssh"
  }
}
```

Et ensuite aller vérifier dans Grafana si j'ai des logs en lançant ce qu'on appel des **LogQL** qui sont des requête pour aller chercher des logs précis par exemple pour les logs SSH :

```
{job="journald"} |= "sshd"
```



Maintenant il faut mettre ça en forme et faire des dashboards. Pour faciliter la création de dashboard on peut en importer avec des LoqQL prédéfinies, j'ai également dû installer Prometheus pour faire un dashboard avec des métriques systèmes.

Maintenant il faut vraiment que je trouve les bons LogQL pour afficher les logs SSH de connexions échouées et de voir de quelle ip provient la connexion, quel port, quel utilisateur etc, c’est ça le but final.

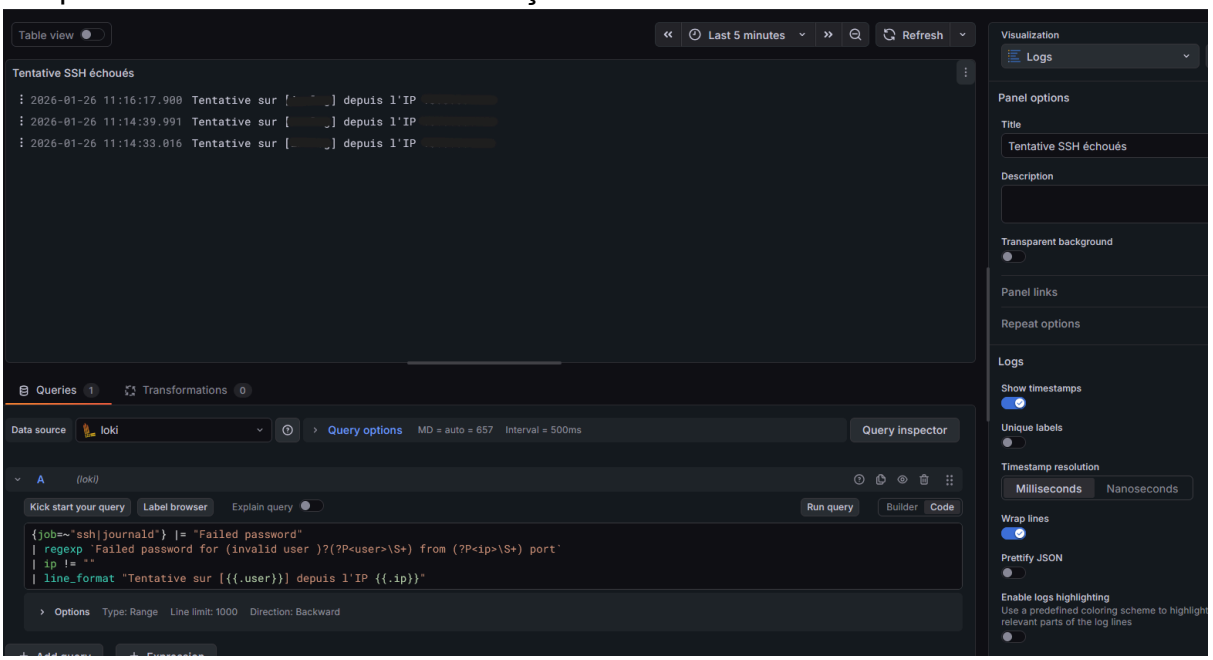
Compte rendu hebdomadaire quatrième semaine

Lundi 26 janvier :

En arrivant lundi je me suis remis sur ma stack de la semaine dernière Grafana , Alloy et Loki pour, je le rappelle avoir des logs SSH et des métriques systèmes, la semaine dernière les métriques marchaient avec Prometheus et remontaient bien dans Grafana. Donc en arrivant ce matin je me suis mis sur les logs SSH afin de les faire marchés et qu'ils remontent bien dans Grafana par Loki en étant bien mis en page, ce que j'ai réussi à faire avec le bon **LogQL :**

```
{job=~"ssh|journald"} |= "Failed password"
| regexp `Failed password for (invalid user )?(?P<user>\S+)
from (?P<ip>\S+) port`
| ip != ""
| line_format "Tentative sur [{{.user}}] depuis l'IP {{.ip}}"
```

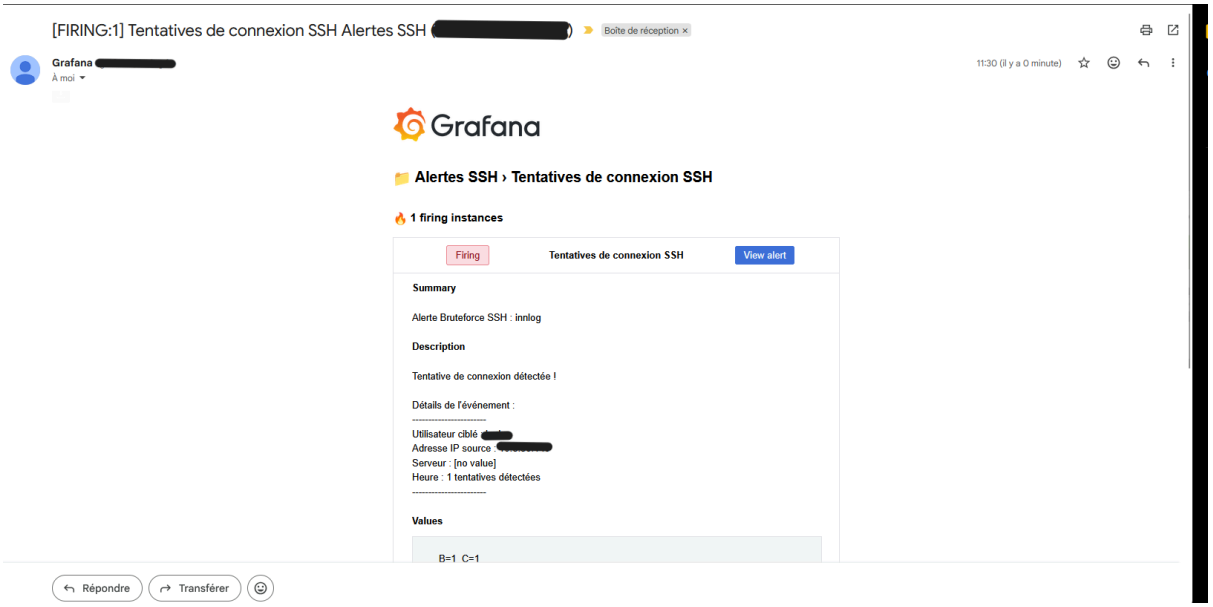
Ce qui me donne un résultat comme ça :



Ensuite le meilleur à faire avec ce genre de résultats c'est de se les envoyer par mail ce que j'ai fais en configurant le serveur SMTP sur Grafana dans son fichier de configuration :

```
[smtp]
enabled = true
host =
user =
# If the password contains # or ; you have to wrap it with triple quotes. Ex """#password;"""
password =
cert_file =
key_file =
skip_verify = false
from_address =
from_name = Grafana
# EHLO identity in SMTP dialog (defaults to instance_name)
ehlo_identity = dashboard.example.com
# SMTP startTLS policy (defaults to 'OpportunisticStartTLS')
startTLS_policy =
# Enable trace propagation in e-mail headers, using the 'traceparent', 'tracestate' and (optionally) 'baggage' fields (defaults to false)
enable_tracing = false
```

Ce qui fait qu'à chaque tentatives échouées de connexions je reçois un mail comme ça :



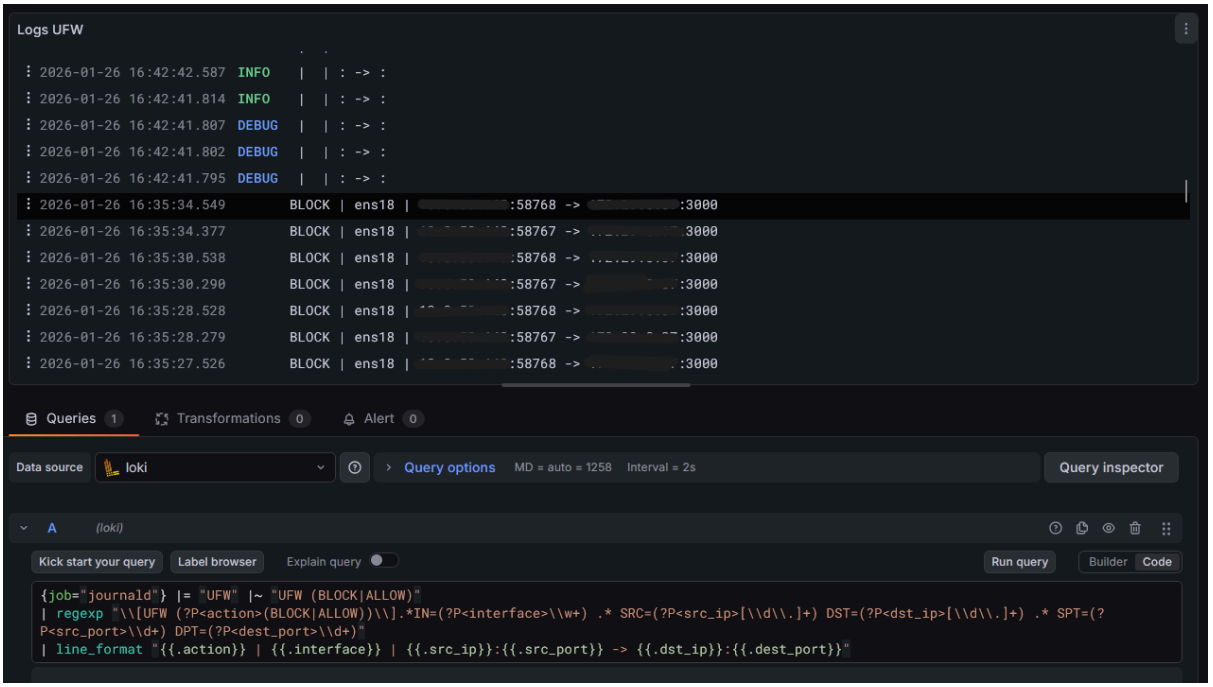
Ensuite Maxime ne me l'avait pas demandé encore mais j'ai pris l'initiative de mettre le serveur Grafana en HTTPS avec un reverse proxy Nginx

```
map $http_upgrade $connection_upgrade {
  default upgrade;
  ''      close;
}

server {
    listen 443 ssl;
    server_name [redacted];

    ssl_certificate [redacted];
    ssl_certificate_key [redacted];

    # optionnel mais utile pour les pièces jointes
    client_max_body_size 128m;

    # Le reste (UI/API)
    location / {
        proxy_pass http://127.0.0.1:3000;

        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
    }
}

server {
    listen 80;
    server_name [redacted];
    return 301 https://$host$request_uri;
}
```

Au début du projet, Maxime m'avait fait part que l'on pourrait mettre un UFW plus tard sur cette machine ce qui ne me gênait pas étant donné que j'utilise déjà cet outil sur mon NAS à la maison donc je le connais très bien j'ai donc commencé à faire quelques règles :

```
root@inl-lso-grafana:/etc/alloy# ufw status verbose
Status: active
Logging: on (medium)
Default: deny (incoming), allow (outgoing), disabled (routed)
New profiles: skip

To                         Action      From
--                         ------      ----
22/tcp                     ALLOW IN    Anywhere                   # Accès SSH
443/tcp                    ALLOW IN    Anywhere                   # HTTPS
9100                       ALLOW IN    Anywhere                   # Prometheus
```

et après je me suis dis que je ferai remonter les logs d'UFW dans Grafana en ajoutant un bloc dans mon fichier de configuration Alloy



Pour l'instant j'en suis là dans la remonter de logs, j'ai encore des lignes vides mais c'est à cause du LogQL qui n'est pas encore optimisé à 100%.

Mardi 27 janvier :

Ce que j'ai fait en arrivant ce matin c'est me remettre sur les logs UFW, j'ai repris les logQL et je les ai refait à ma manière, j'ai donc fait deux requêtes dans mon grafana, une pour les UFW ALLOW et l'autre pour les UFW BLOCK :

```
{job="journald"}
|~ "UFW ALLOW"
| regexp "SRC=(?P<src>[\\d\\.]+)"
| regexp "DST=(?P<dst>[\\d\\.]+)"
| regexp "SPT=(?P<spt>\\d+)"
| regexp "DPT=(?P<dpt>\\d+)"
| regexp "PROTO=(?P<proto>\\w+)"
| src != ""
| line_format "ALLOW [{{.proto}}] | SRC: {{.src}}:{{.spt}} | DST:
{{.dst}}:{{.dpt}}"
```

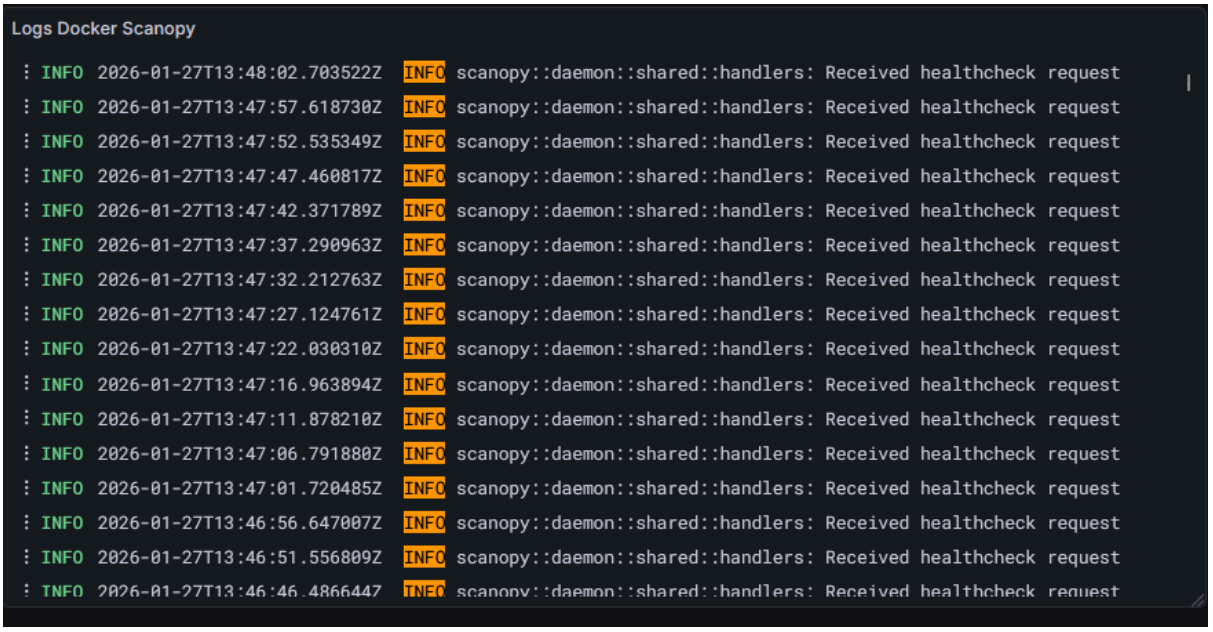
Ce LogQL va chercher dans les logs de UFW tous les blocs ALLOW, il va ensuite prendre l'IP source `| regexp "SRC=(?P<src>[\\d\\.]+)"`, l'IP destination `| regexp "DST=(?P<dst>[\\d\\.]+)"`, le port source et de destination `| regexp "SPT=(?P<spt>\\d+)", | regexp "DPT=(?P<dpt>\\d+)"` et enfin le protocole `| regexp "PROTO=(?P<proto>\\w+)"`. Ensuite cette ligne là permet de ne pas avoir de ligne vide dans le parsing des logs `| src != ""` et enfin cette ligne sert à mettre en page les précédentes "variables" `| line_format "ALLOW [{{.proto}}] | SRC: {{.src}}:{{.spt}} | DST: {{.dst}}:{{.dpt}}"`
Ce qui va donner des lignes de logs comme ça :

et pareil pour les blocs BLOCK :

```
{job="journald"}
|~ "UFW BLOCK"
| regexp "SRC=(?P<src>[\\d\\.]+)"
| regexp "DST=(?P<dst>[\\d\\.]+)"
| regexp "SPT=(?P<spt>\\d+)"
| regexp "DPT=(?P<dpt>\\d+)"
| regexp "PROTO=(?P<proto>\\w+)"
| src != ""
| line_format "BLOCK [{{.proto}}] | SRC: {{.src}}:{{.spt}} | DST:
{{.dst}}:{{.dpt}}"
```

La prochaine étape la serait d'installer un agent Alloy sur une autre machine et de la faire communiquer avec la machine où est installé grafana, j'ai décider de prendre une machine où on a installé Scanopy, c'est un logiciel Open Source qui scan le réseau afin de faire une topologie automatique de l'infrastructure, l'outil est installé avec Docker, il y a donc des logs qui peuvent être intéressant.

J’ai réussi à faire remonter les logs de Scanopy dans le grafana, le “problème” c’est que ce sont des logs docker qui remontent étant donné que Scanopy a été installé en docker donc les logs ne sont pas trop parlant étant donné qu’il n’y a pas d’erreur je n’ai que des logs INFO.

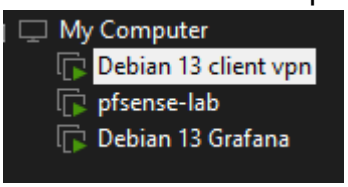


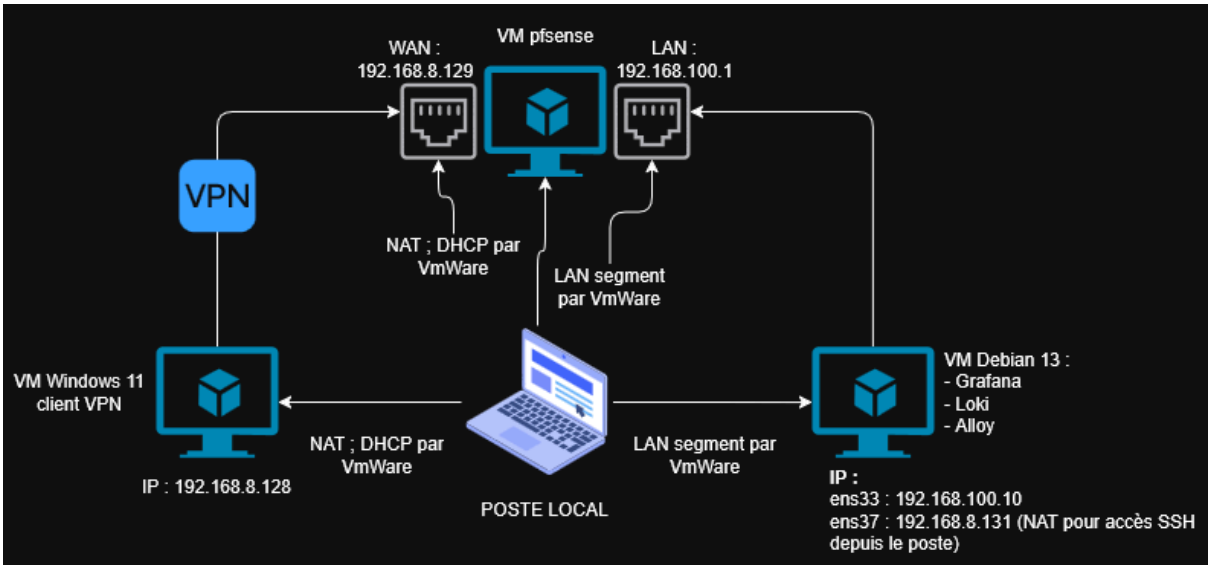
La prochaine étape serait d’installer ce genre de système sur un pfSense pour vraiment avoir beaucoup de logs et parser le maximum de logs pour que ce soit plus lisible.

Mercredi 28 janvier :

En arrivant ce mercredi, il y a eu une infiltration d’eau dans la salle des baies et en fait l’infrastructure ProxMox sur laquelle je bossais depuis le début n’est pas brassée et la multiprise à laquelle sont branchés les hyperviseur était par terre et donc était dans l’eau, ils ont donc éteint toute l’infra donc je ne pouvais plus travailler sur mes VM. J’ai donc dû faire un LAB avec des VM locales dans VirtualBox, j’ai fais un LAB avec un Windows 11 qui sera connecté en VPN à une VM pfSense et avec une Debian 13 version CLI qui elle jouera le rôle de Grafana + alloy + Loki.

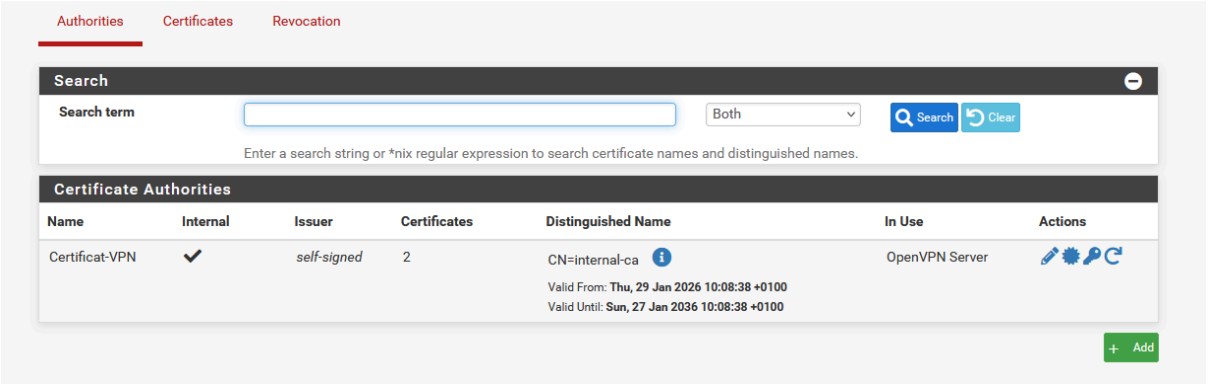
J’ai passé une bonne partie de ma journée à debugger l’aspect réseau de mes VM pour avoir mes plusieurs interfaces sur mon pfSense, mettre une VM dans le LAN et dans le WAN de mon pare-feu.



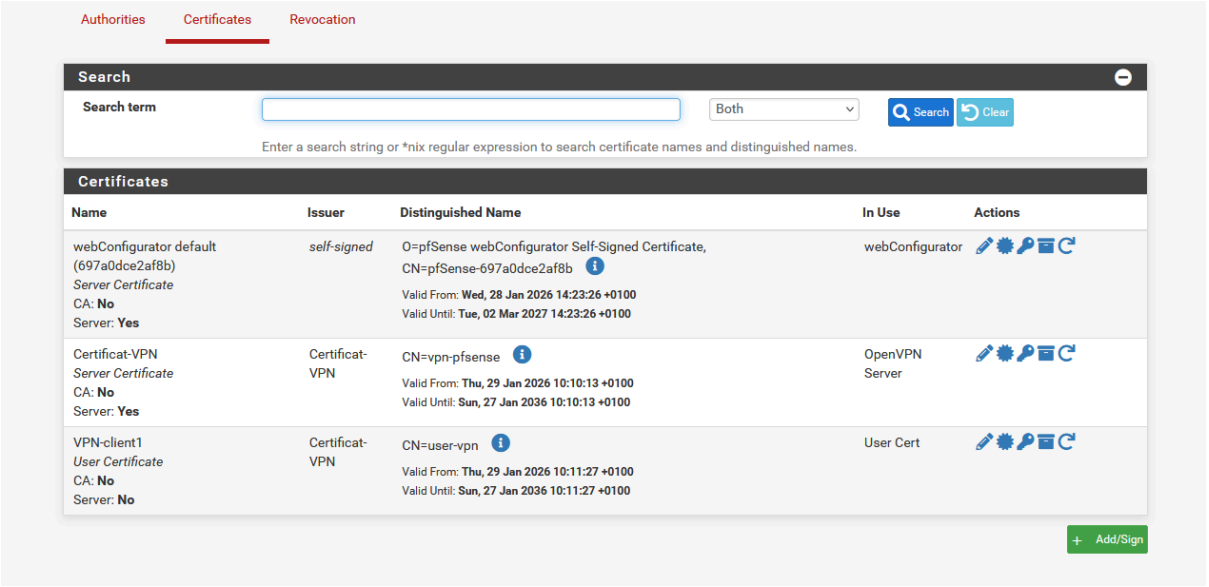


Maintenant que les deux VM Debian ping leur passerelle je vais commencer à me renseigner sur comment faire le tunnel VPN sur le WAN.

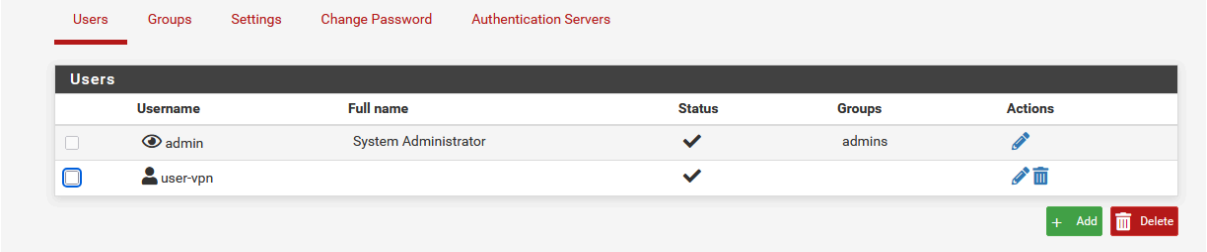
On peut d'abord créer une autorité de certification auto-signée.



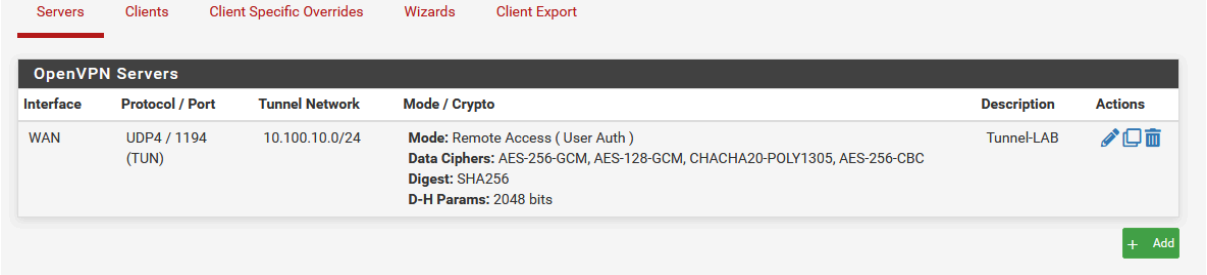
Créer le certificat avec.



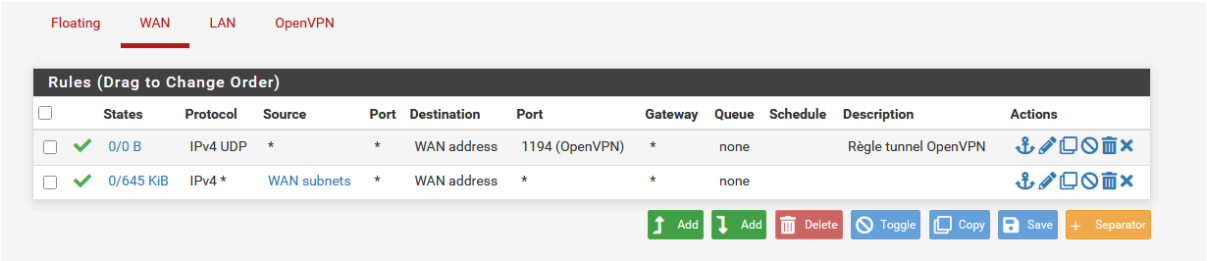
Créer l'utilisateur qui pourra se connecter avec son fichier de configuration.

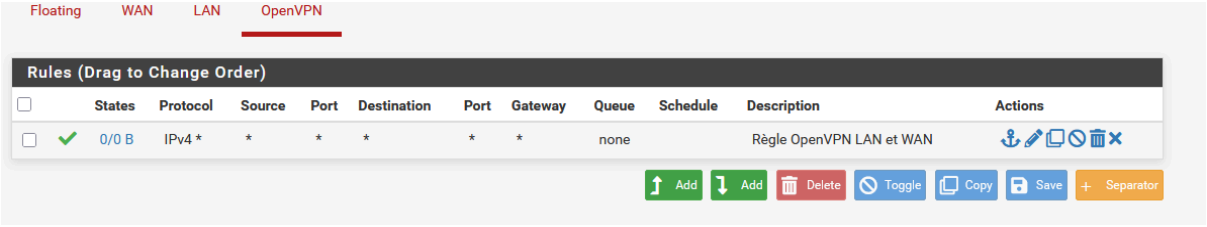


Ensuite il faut créer le tunnel VPN OpenVPN sur le pfsense.

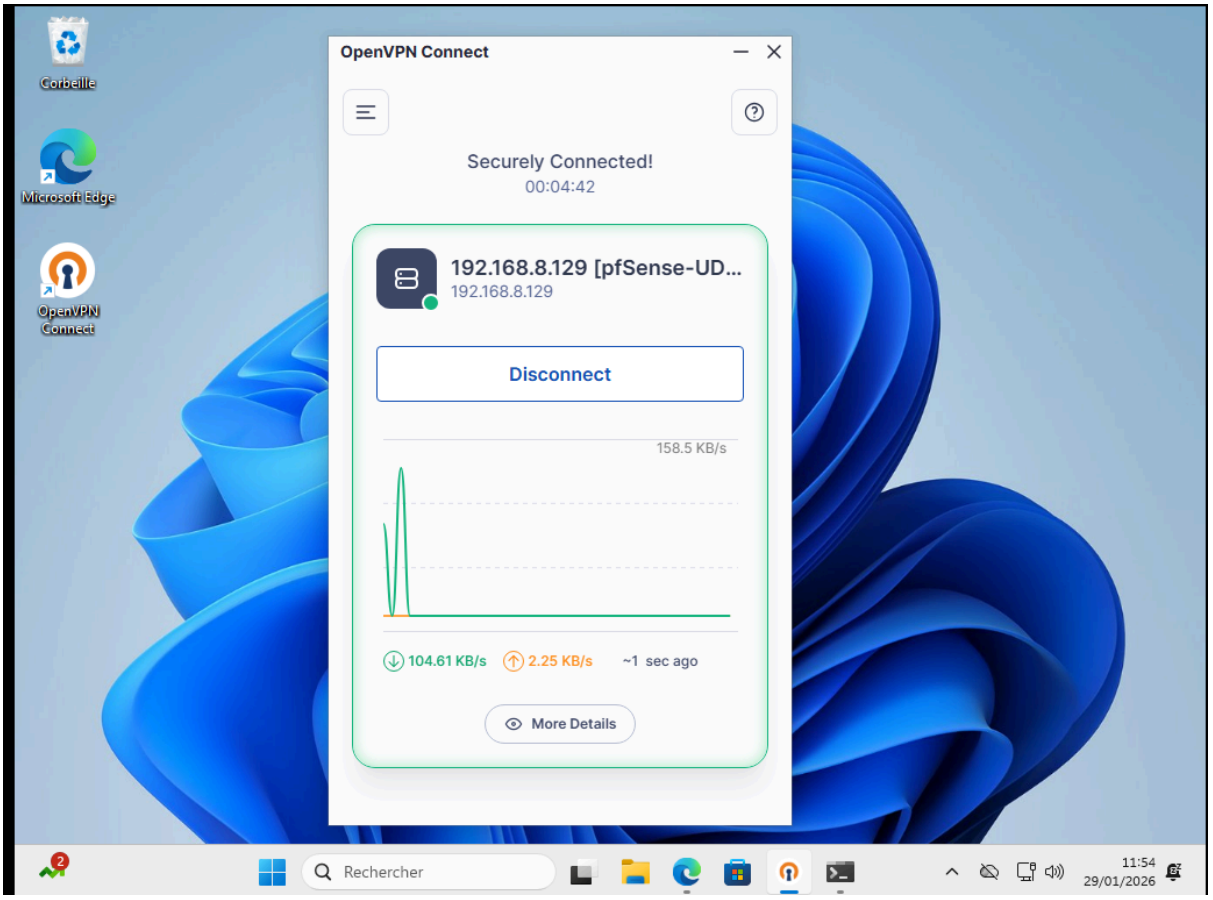


Et enfin créer les bonnes règles de pare-feu pour le WAN et l'interface OpenVPN.





Se connecter avec le client Windows une fois qu'il a son fichier de configuration.



Et aller voir ses logs qui faudra faire remonter dans le Grafana que je vais refaire sur ma Debian 13 locale

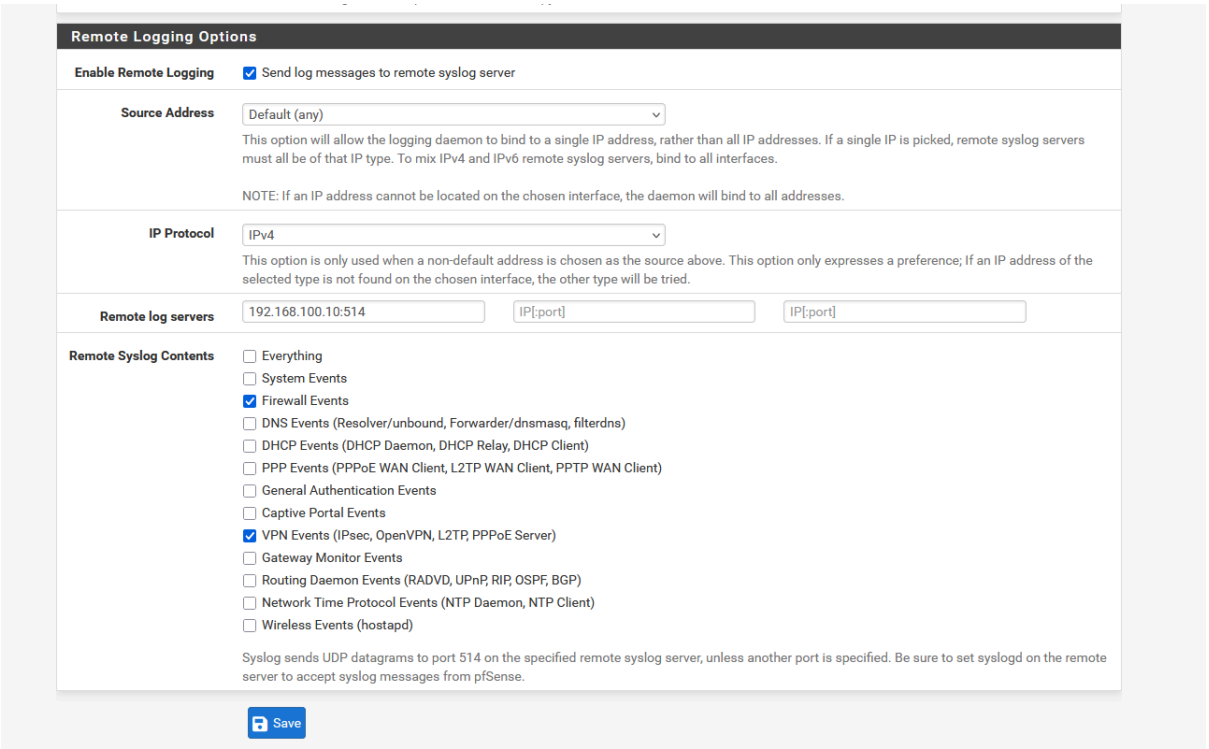
Last 80 OpenVPN Log Entries. (Maximum 500)

Time	Process	PID	Message
Jan 29 11:49:28	openvpn	35378	openvpn server 'ovpns1' user 'user-vpn' address '192.168.8.130:57165' - connected
Jan 29 11:49:28	openvpn	30680	openvpn server 'ovpns1' user 'user-vpn' address '192.168.8.130:57165' - connecting
Jan 29 11:49:28	openvpn	56242	192.168.8.130:57165 MULTI_sva: pool returned IPv4=10.100.10.2, IPv6=(Not enabled)
Jan 29 11:49:27	openvpn	76622	user 'user-vpn' authenticated

Ce genre de logs là

En fin de journée j'ai réussi à faire remonter les logs dans rsyslog sur ma machine Debian ce qui est un bon début.

Il faut d'abord configurer l'envoi de log sur une machine distante dans le pfsense :



Ensuite dans la machine Debian j'ai fait un fichier de configuration dans /etc/rsyslog.d qui détecte que si le host est le pfsense il met les logs qu'il reçoit dans un fichier de log à part entière dans /var/log/pfsense.log :

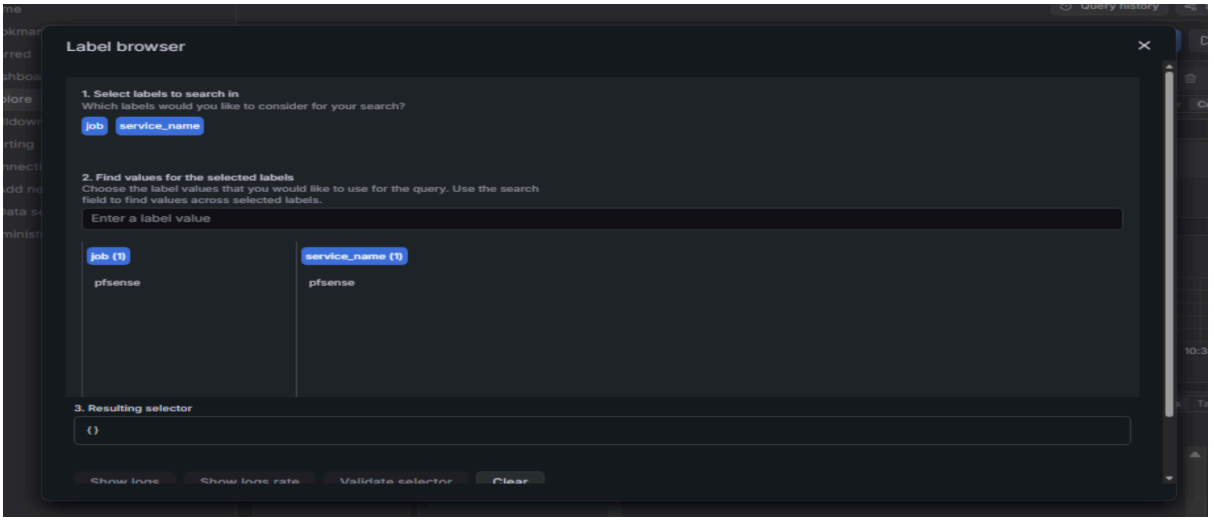
```
root@debian:~# cat /etc/rsyslog.d/pfsense.conf
if $fromhost-ip == '192.168.100.1' then /var/log/pfsense.log
& stop
```

```
2026-01-29T15:56:55+01:00 192.168.100.1 openvpn[64136]: 192.168.8.130:58205 MULTI_sva: pool returned IPv4=10.100.10.6, IPv6=(Not enabled)
2026-01-29T15:56:55+01:00 192.168.100.1 openvpn[8722]: openvpn server 'ovpns1' user 'user-vpn' address '192.168.8.130:58205' - connecting
2026-01-29T15:56:55+01:00 192.168.100.1 openvpn[8722]: openvpn server 'ovpns1' user 'user-vpn' address '192.168.8.130:58205' - connecting
2026-01-29T15:56:55+01:00 192.168.100.1 openvpn[13543]: openvpn server 'ovpns1' user 'user-vpn' address '192.168.8.130:58205' - connected
2026-01-29T15:56:55+01:00 192.168.100.1 openvpn[13543]: openvpn server 'ovpns1' user 'user-vpn' address '192.168.8.130:58205' - connected
```

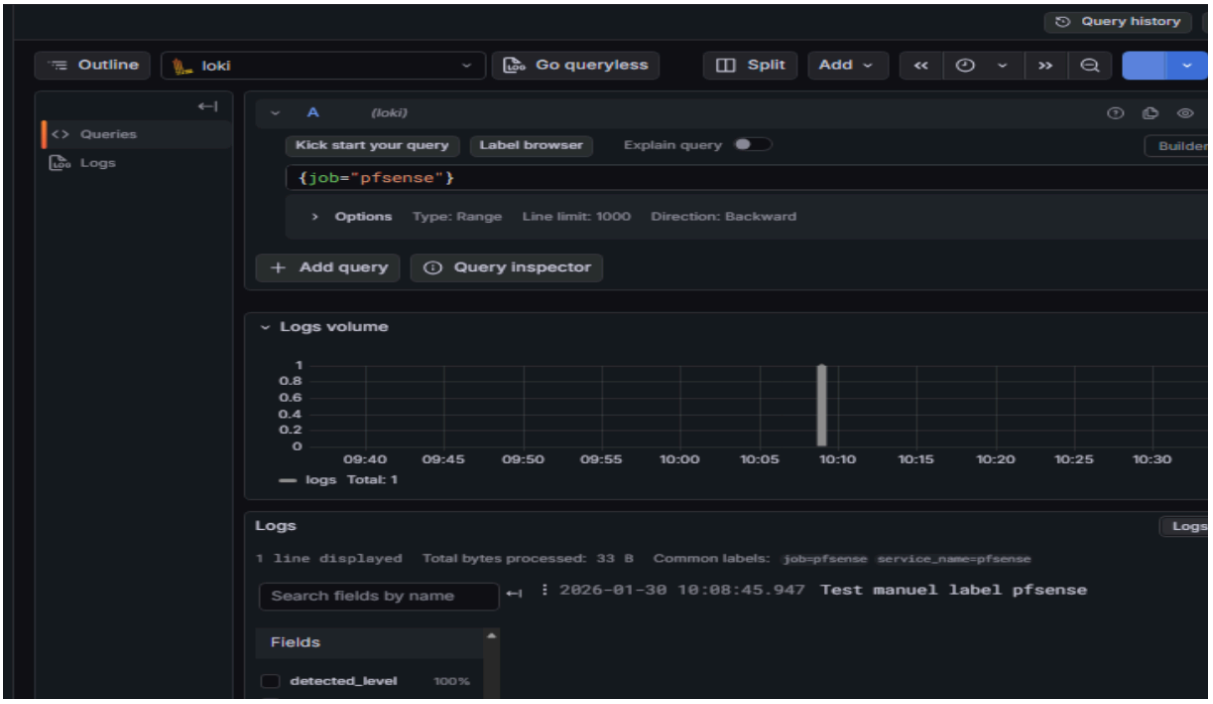
ce qui me donne des logs comme ça. Ensuite il faut les faire remonter dans Alloy pour qu'il puisse les push dans Loki.

Vendredi 30 janvier :

Le vendredi j'étais en télétravail mais j'ai quand même pu avancer sur mon LAB étant donné que ce sont des VM locales ça ne posait pas de problème, cependant j'ai passé une bonne partie de ma matinée à debugger ces logs qui ne remontaient pas dans grafana, en fait si, ils remontent mais pas les bons, le label pfsense remonte bien dans grafana.



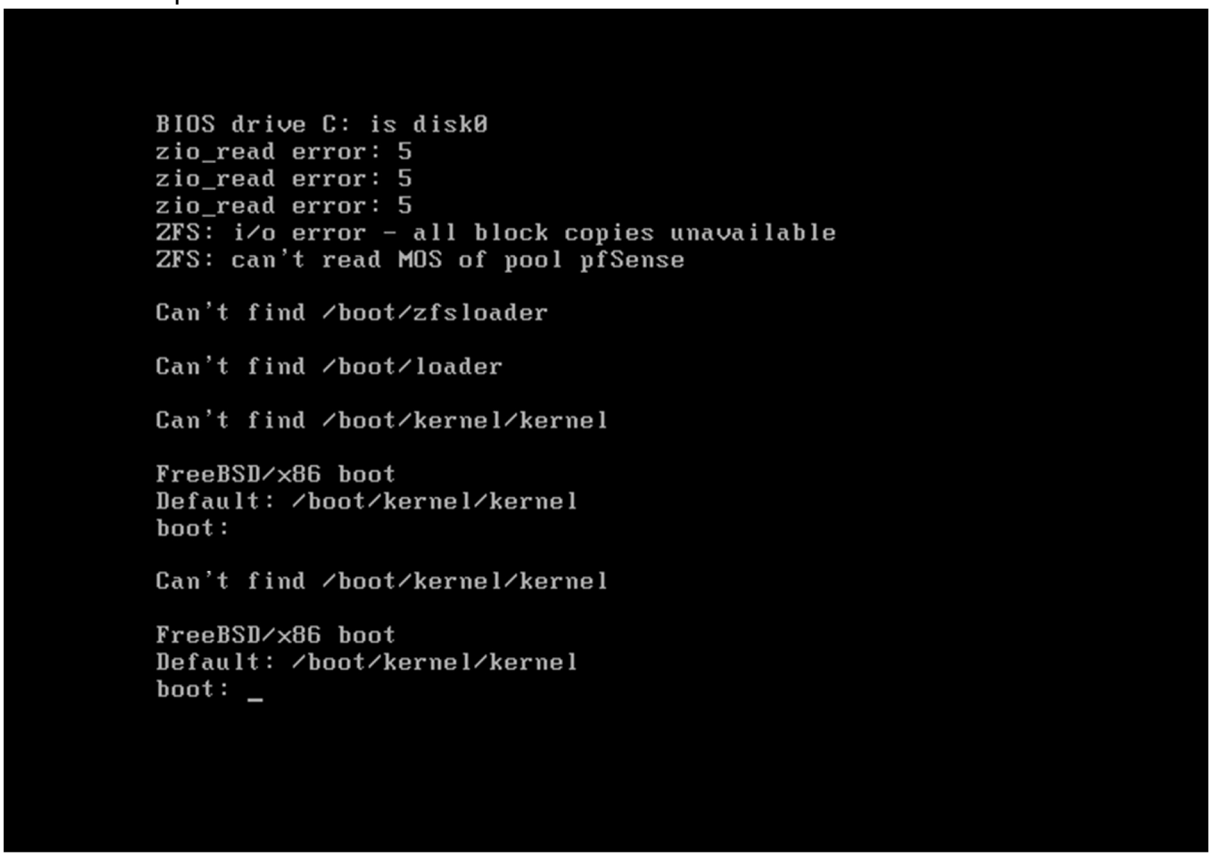
et quand je fais un log manuel, c'est à dire que je force un log dans le fichier pfsense.log, il remonte bien dans grafana comme on peut le voir ici :



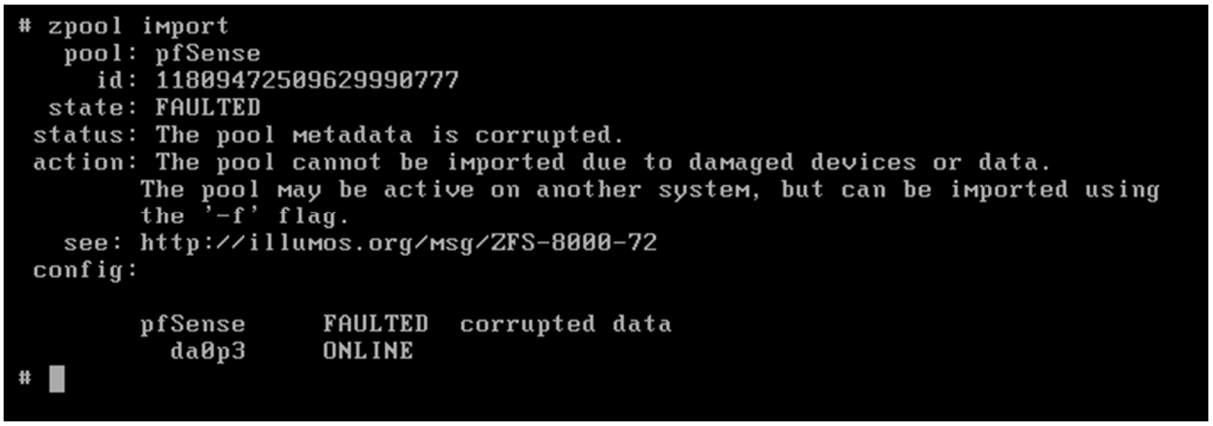
Ce que je suspecte c'est le format des logs du pfsense, en effet, l'horodatage des logs qui remontent dans le fichier de log sont en format **RFC3339 2026-01-30T10:29:41+01:00 192.168.100.1 openvpn[55413]: user 'user-vpn' authenticated** ce qui pourrait être mal interprété par Alloy et donc par l'API de Loki, comme le log forcé à été envoyé avec un format d'horodatage basique il a réussi à l'interpréter et donc le faire remonter dans grafana.

Compte rendu hebdomadaire cinquième semaine

Lundi 2 février :
En arrivant le lundi, ma machine virtuelle pfSense était HS. En effet le disque virtuel était corrompu et les données avec.

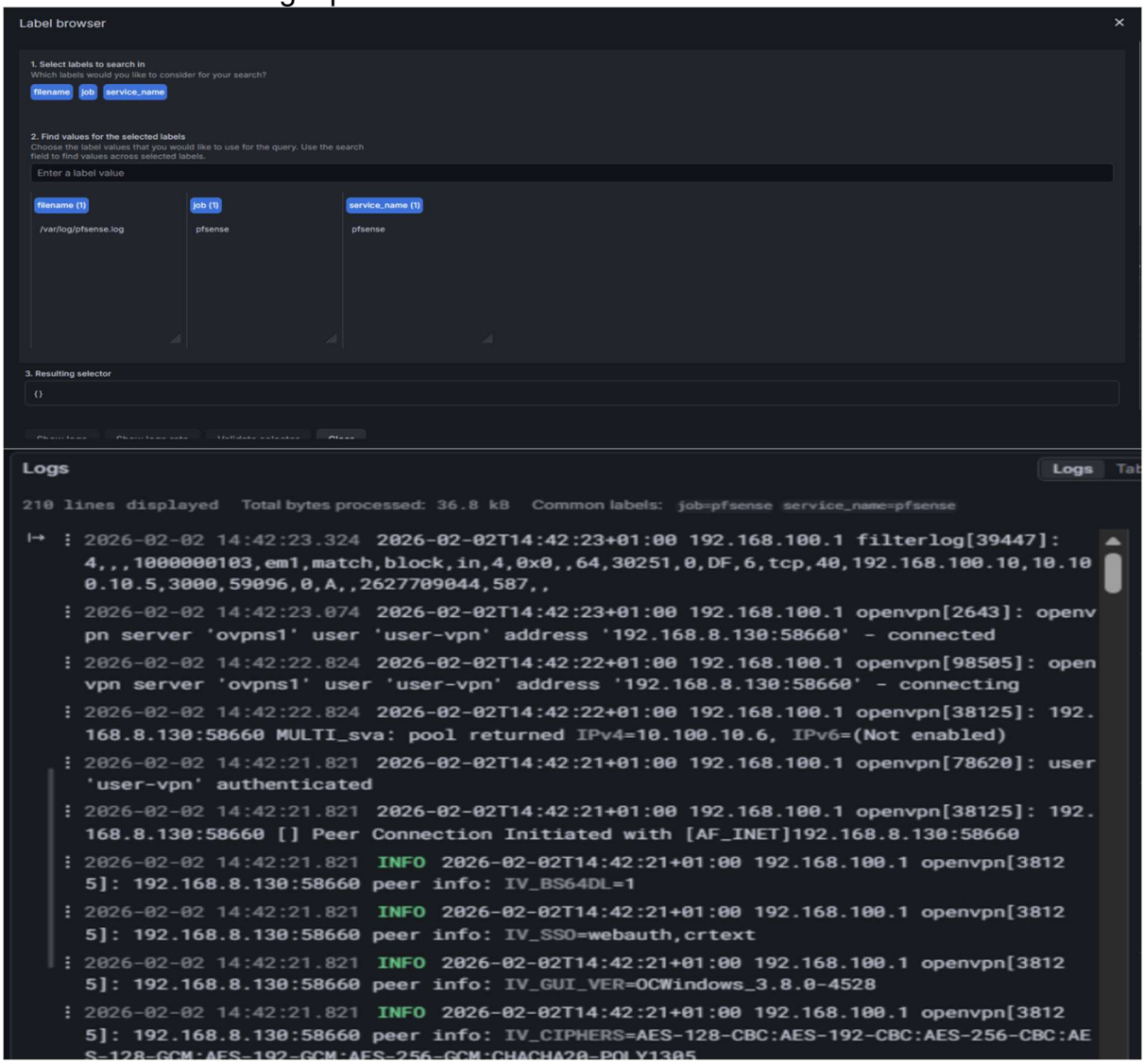


En démarrant la VM j'arrivais sur quelque chose comme ça. J'ai vite compris qu'il y avait un problème avec le disque, j'ai donc démarré la VM sur l'ISO (changer l'ordre de boot dans le BIOS) et j'ai fait le menu Rescue Shell disponible dans le firmware de pfsense.
Et en faisant la commande **zpool import** j'ai vite compris que j'allais devoir refaire ma VM.



Là c'est simple les métadonnées du disque virtuel sont **corrompues.**

Tout le reste de ma matinée je l'ai passé à refaire ma VM et à reconfigurer le tunnel VPN, les interfaces du pare-feu etc.
Une fois que tout était revenu à la normal, j'ai enfin réussi à faire remonter les logs dans Grafana et ce n'était pas une erreur de parsing de l'horodatage des logs, j'ai juste bidouiller le fichier de configuration d'Alloy jusqu'à avoir le bon label (pfsense) et enfin d'avoir les logs qui remontent.



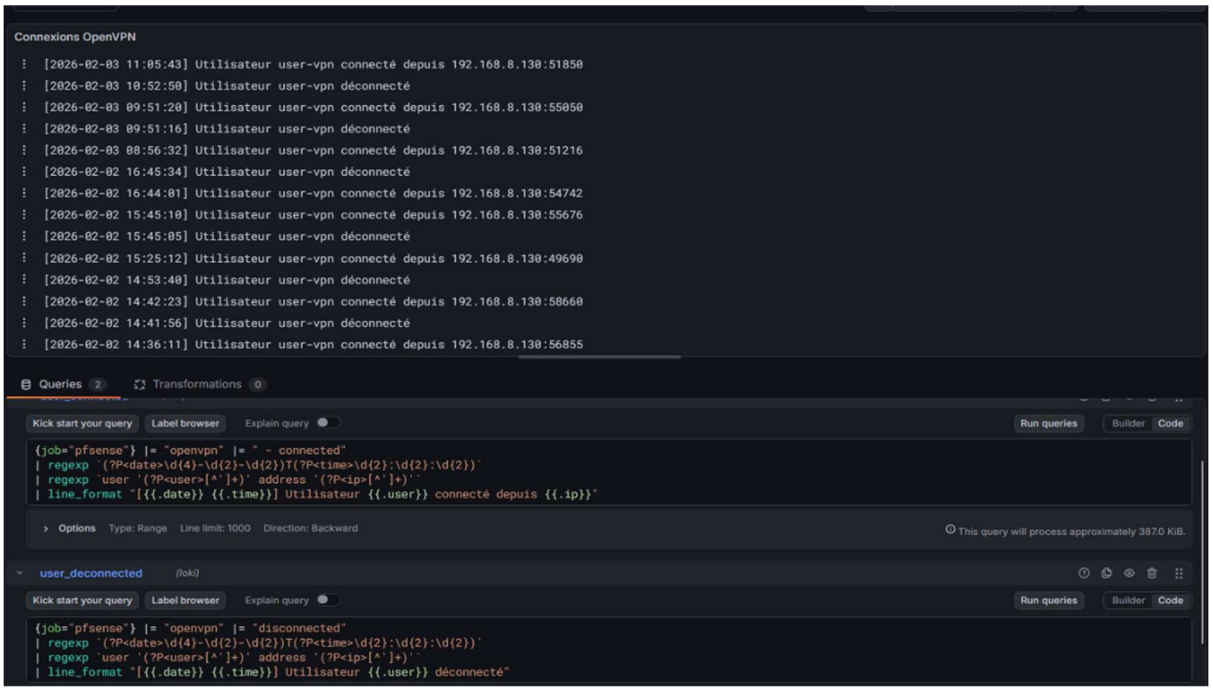
Maintenant il y a plus qu'à mettre les bon logQL pour parser ce que l'on veut c'est à dire les block du pare-feu et les connexions VPN.

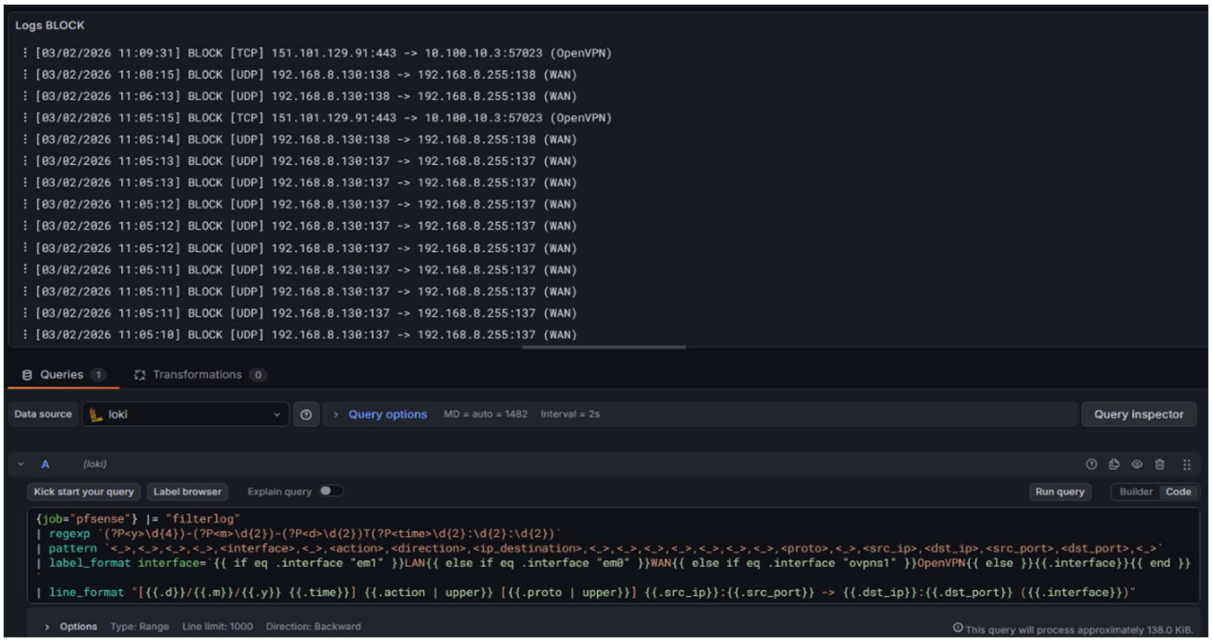
Fichier Alloy final :

```
local.file_match "pfsense_actual" {
    path_targets = [
        { "job" = "pfsense", "__path__" = "/var/log/pfsense.log" },
    ]
}

loki.source.file "pfsense_read" {
    targets    = local.file_match.pfsense_actual.targets
    forward_to = [loki.write.local_loki.receiver]
}

loki.write "local_loki" {
    endpoint {
        url = "http://127.0.0.1:3100/loki/api/v1/push"
    }
}
```

Il faut aussi faire ça dans le fichier pfsense.conf qui crée en fait le fichier pfsense.log si les logs proviennent de 192.168.100.1 c'est-à-dire l'adresse LAN du pfsense.

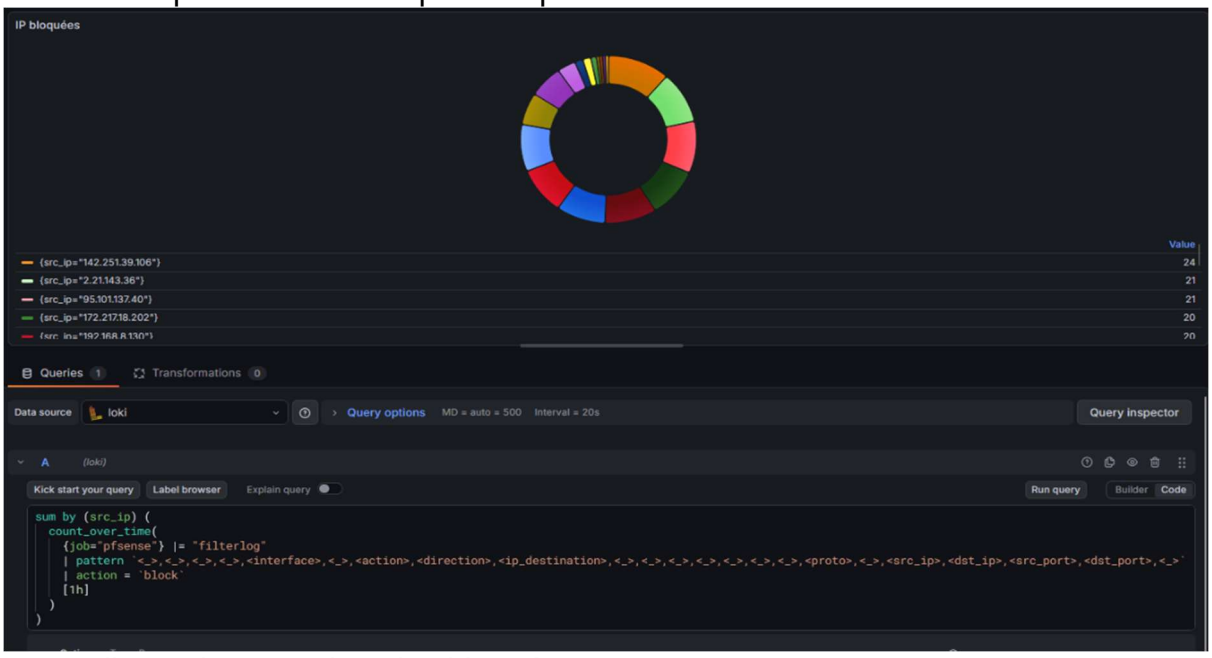
```
root@debian:/etc/rsyslog.d# cat pfsense.conf
if $fromhost-ip == '192.168.100.1' then /var/log/pfsense.log
& stop
```



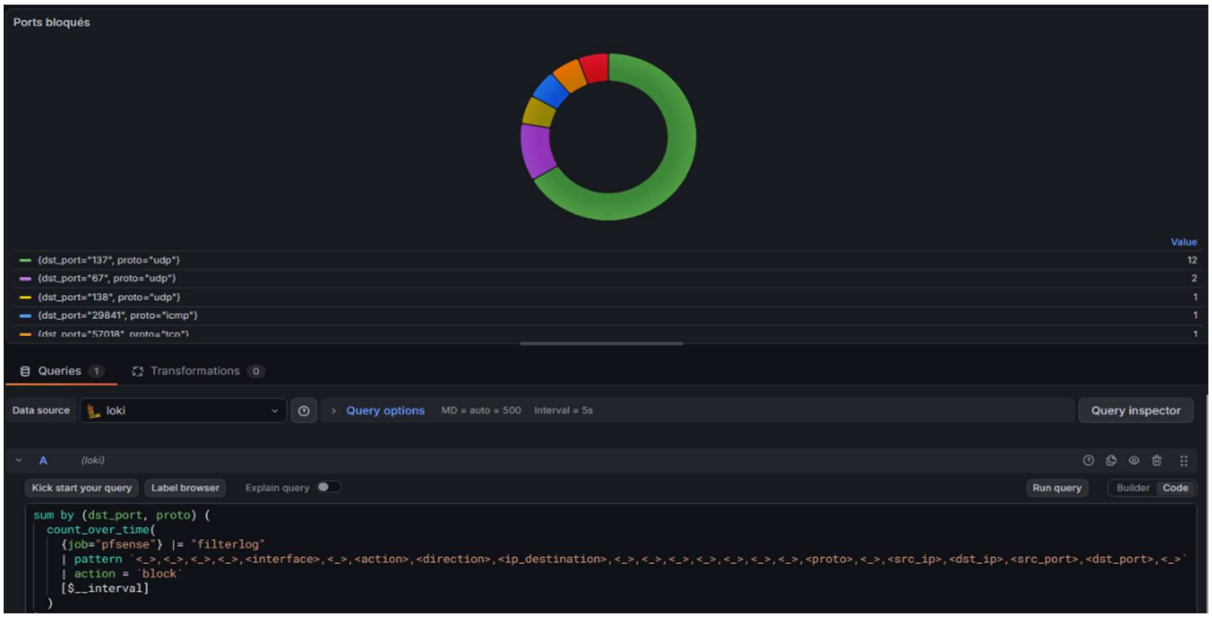


Ensuite je pense que je vais m'attaquer à faire des graphiques en camembert pour faire ressortir les IP les plus bloquées ou les ports les plus bloqués etc.

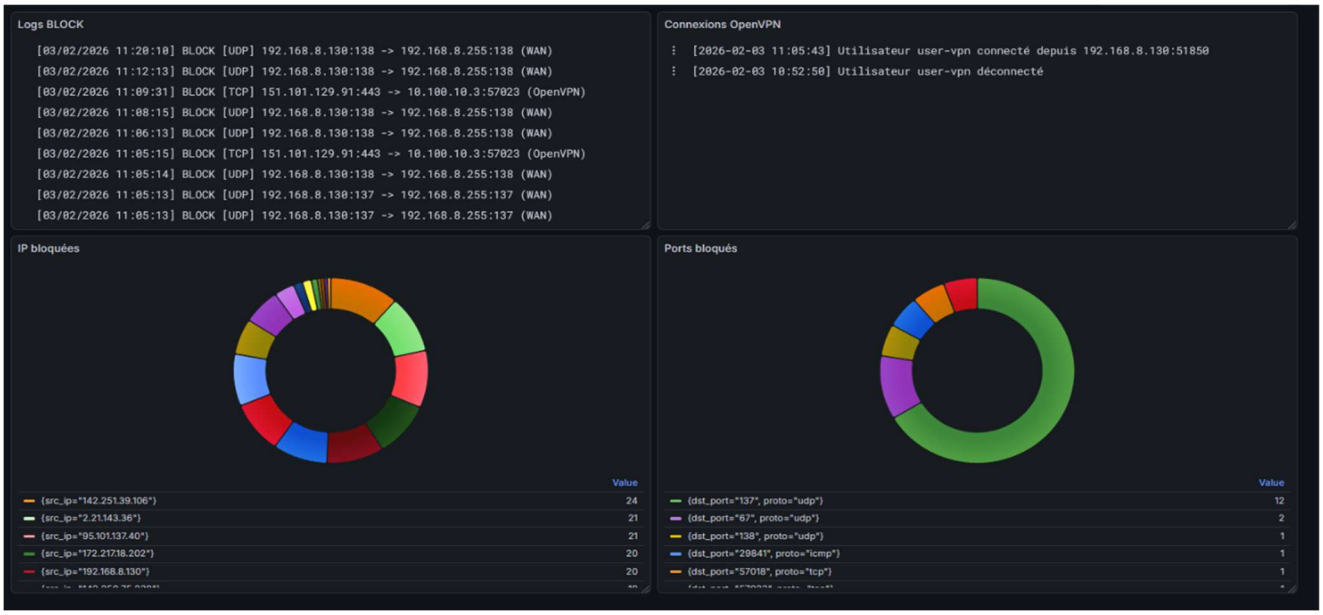
Mardi 3 février :
En arrivant le mardi je me suis penché sur le graphique en Pie chart c'est à dire en camembert pour le les IP les plus bloquées :



et pareil pour les ports les plus bloqués :

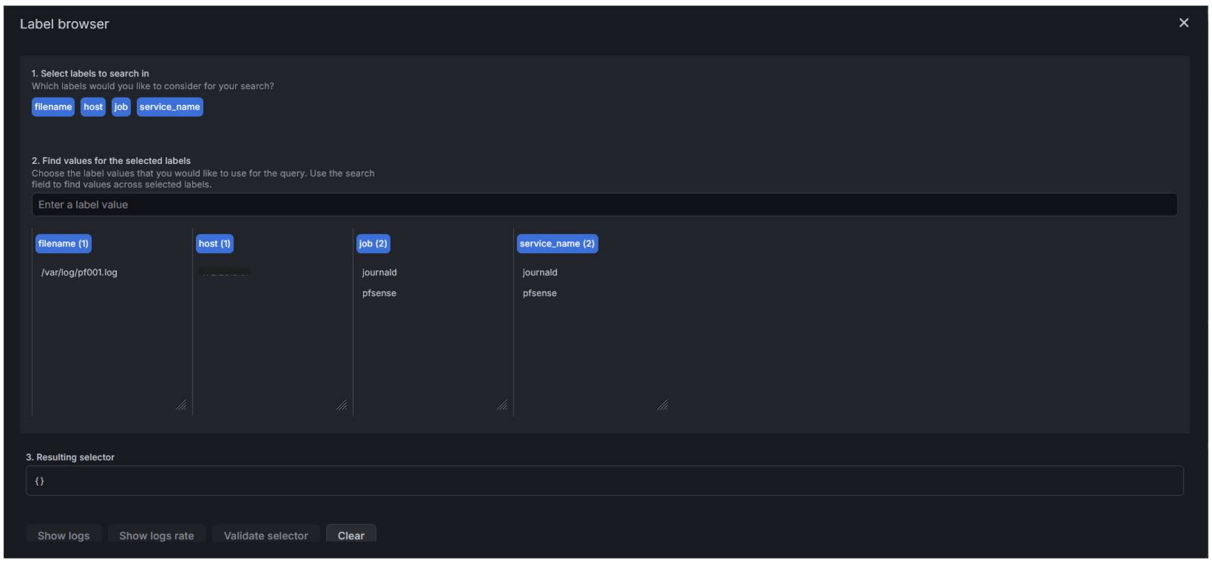


ce qui à la fin nous donne un dashboard complet comme ça :

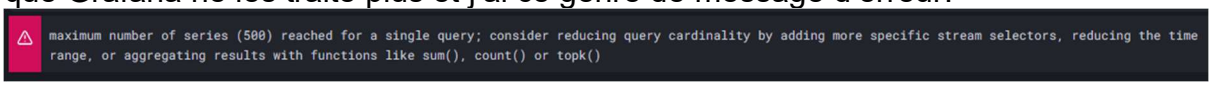


Mercredi 4 février :
Maintenant je pense que le but final est d'appliquer ces dashboard sur le vrai pfsense du service IT et sur une vraie VM grafana, la première que j'ai faite qui tourne sur l'infrastructure ProxMox.

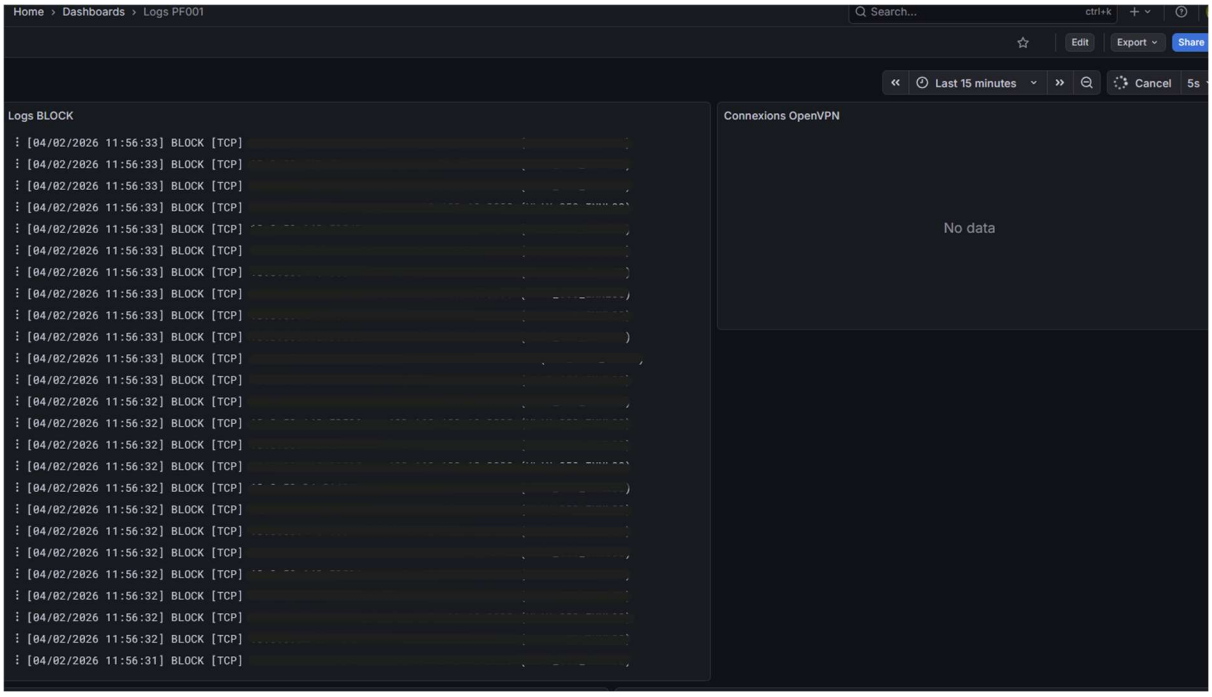
J'ai donc tout configuré comme sur mes VM locales et le label pfsense remonte bien dans grafana.



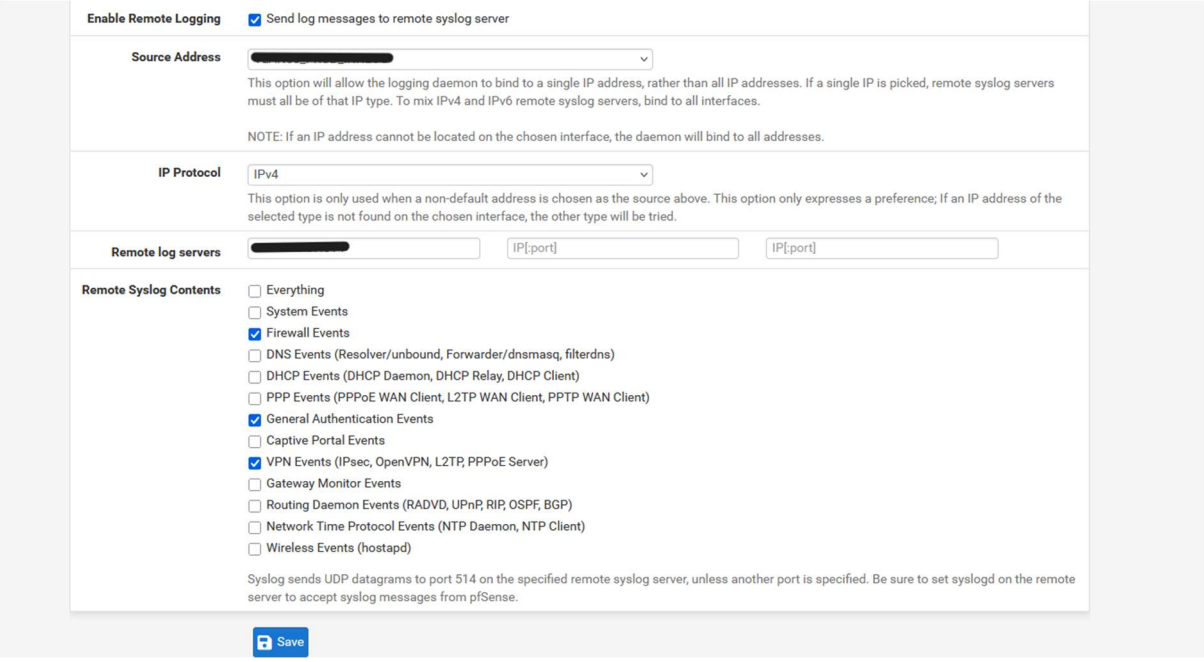
Sauf que j'ai un problème, là j'ai repris les même logQL que pour mes VM locales et pour le graphique des IP bloquées et des ports bloqués, il y a tellement de requête que Grafana ne les traite plus et j'ai ce genre de message d'erreur.



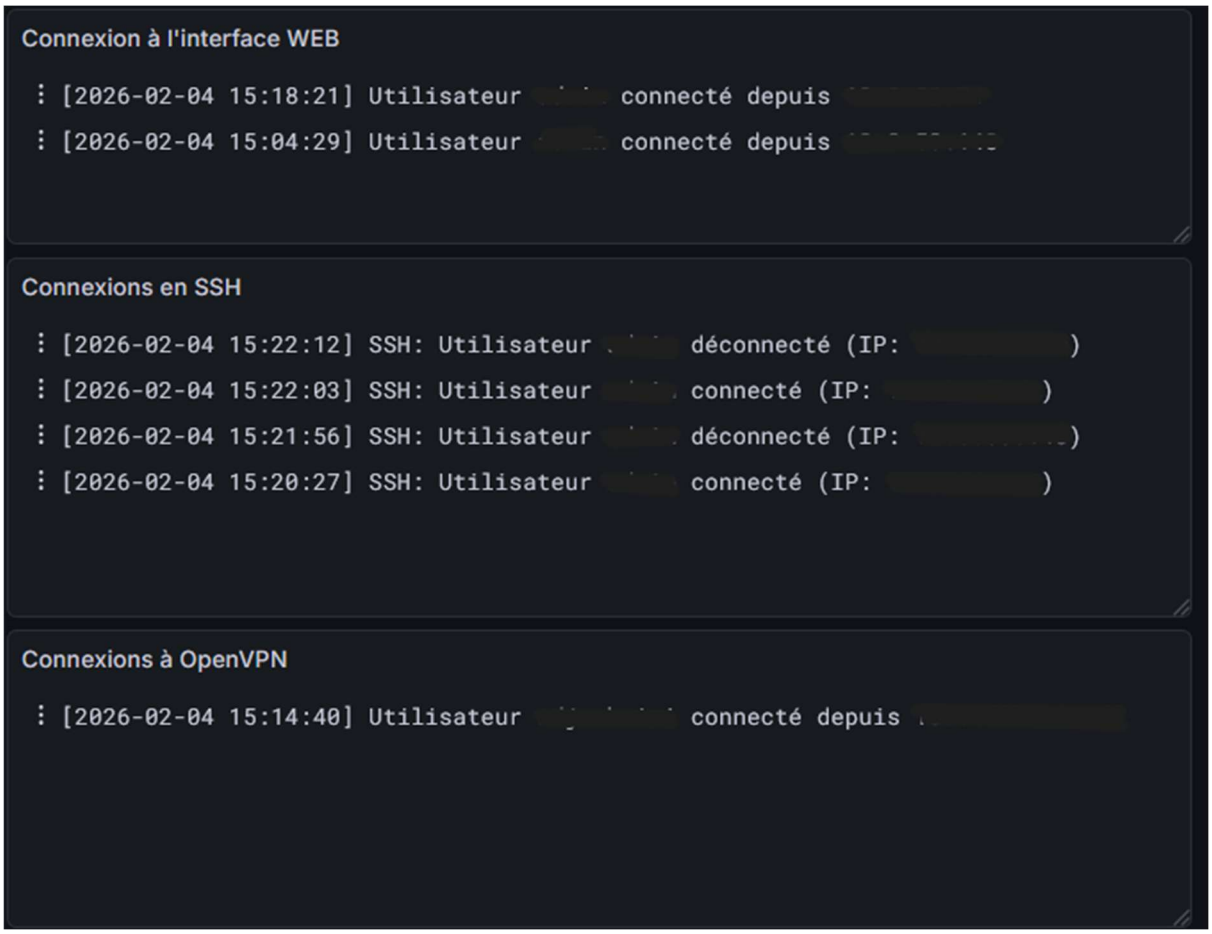
En tout cas pour les logs de base et les connexions VPN ça marche très bien les connexions et les flux bloqués remontent bien dans Grafana :



Je n'ai pas dans les connexions OpenVPN car personne ne s'est connecté dessus mais, l'historique de connexion remonte bien. J'ai aussi essayé de remettre des graphiques en camembert sur le vrai grafana mais c'est pareil j'ai trop de requêtes, trop d'IP bloquées et de ports pour être traités par grafana. Du coup au lieu de faire des graphiques j'ai fais remonté d'autres logs comme les événements d'authentifications :

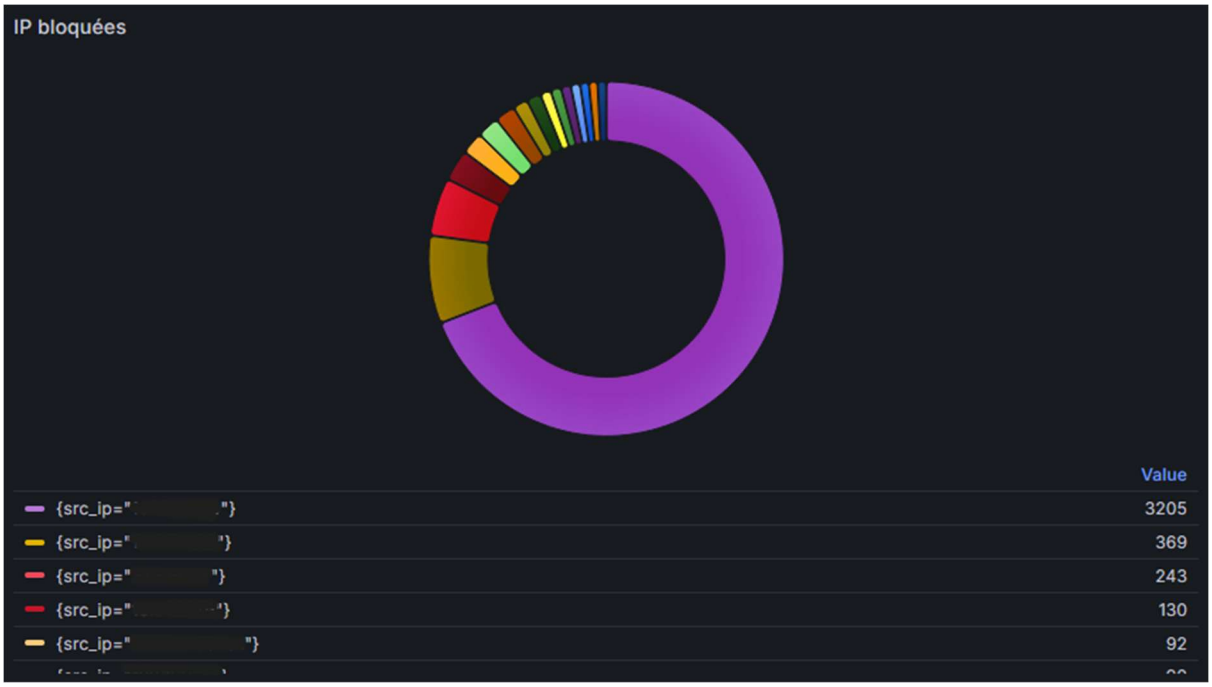


ce qui me donne quelque chose comme ça :



Le meilleur résultat que je peux avoir avec les graphiques c’est sur une range de temps de 30 minutes et faire apparaître le nombre d’IP bloquées sur 30 glissante et

pas sur un total c'est-à-dire que ça sera le total d'IP bloquées sur les dernières 30 minutes et pas en totalité.



Jeudi 5 février :
Ce matin j'ai essayé d'attribuer plus de coeurs processeurs et de RAM à la machine virtuelle pour qu'elle puisse traiter plus de requêtes, je me suis dis que ça allait régler le problème et ça l'a réglé. Je peux maintenant mettre une plage de temps plus grande sans que le graphique ou la grafana ne plante. Je finis donc avec un dashboard pfsense assez complet :

- un panel pour les BLOCK qui reprend les flux bloqués
- un panel pour les connexions à la Web UI de pfsense
- un panel pour les connexions SSH au pfsense
- un panel pour les connexions et les tentatives échouées aux différents tunnels VPN qui sont sur le pfsense
- et un panel graphique pour récupérer les IP les plus bloquées sur une certaine plage de temps.

Jeudi après-midi, j'ai participé à un atelier DevOps avec Maxime et un collègue à lui Noël, pendant les trois heures de l'après-midi j'ai appris plein de choses sur les clusters Kubernetes et leurs fonctionnement, les différences avec Docker etc et comment eux peuvent s'en servir pour leur infrastructure as code. Ensuite, ils étaient sur une problématique depuis quelques jours. En effet, Innlog est une entreprise d'édition de logiciel et vend donc une solution logicielle qui s'appelle Tesfri, un logiciel pour la gestion d'entrepôt frigorifique et ils ont une application lourde et une application web, la problématique elle se trouvait sur l'application web ou une API d'authentification renvoyait une valeure null en précisant un certain paramètre alors qu'elle était censée renvoyer un token. Le problème venait des fichiers de configuration de plusieurs API qui en fait tapait sur le endpoint de l'application en pré prod et non pas en prod.

Vendredi 6 février :
En arrivant ce matin, Maxime m'a proposé d'approfondir mes connaissances sur les clusters Kubernetes et comment leur infrastructure fonctionne étant donné qu'ils sont hébergés chez OVH.

Déjà dans un premier temps la principal différences avec Docker est que Docker est un **conteneur** donc une seule image à la fois et Kubernetes est un **orchestrateur** de conteneur c'est à dire que Kubernetes lui va s'occuper de gérer et faire communiquer des **dizaines voir des centaines de conteneurs à la fois.**

Il y a tout un jargon propre à Kubernetes qui est très important à comprendre pour être le plus familier possible avec cette technologie.

On a d'abord les composants cerveau du cluster :

- **Control plane :** sont l'ensemble des services qui dirigent le cluster. Dans l'offre MKS de chez OVH cloud, cette partie est infogérée par OVH directement.
- **API Server :** C'est l'API qui reçoit les fichiers YAML des conteneurs.
- **ETCD :** L'ETCD est un des points d'ancrage d'un cluster Kubernetes, c'est une base de données qui stocke toutes les configurations de tout ce qui tourne dans le cluster.
- **Scheduler :** C'est un algorithme qui permet de gérer l'espace de stockage de votre cluster, c'est lui va décider sur quel serveur (Node) placer les conteneurs.

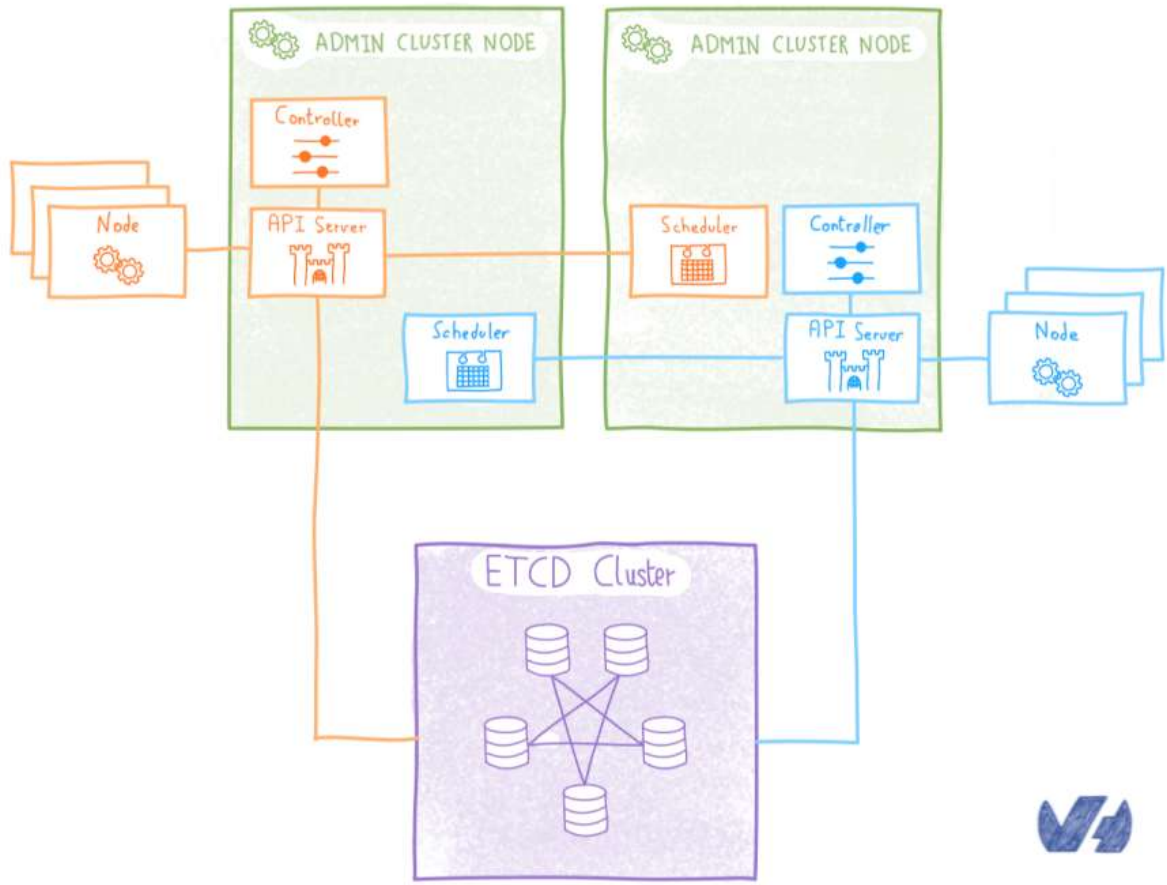
Ensuite on a le vocabulaire des objets qui se trouvent un cluster :

- **les Pods :** c'est la plus petite unité d'un cluster, il contient un ou plusieurs conteneurs et ces mêmes conteneurs partagent la même adresse IP et le même stockage local.
- **Deployement :** Le Deployement est un objet qui définit comment l'application tourne, la version des images Docker, les mises à jour automatiques etc.
- **ReplicaSet :** le ReplicaSet est une stratégie de déploiement dans Kubernetes qui va nommer les pods aléatoirement si le pod redémarre, l'ordre de démarrage est en parallèle donc tous les pods en même temps, les volumes des conteneurs sont souvent partagé et les IP sont changeantes.
- **StatefulSet :** le StatefulSet est une autre stratégie de déploiement qui est beaucoup plus stable avec une nomenclature des conteneur qui est définit, un ordre de démarrage séquentiel, des volumes individuel et des IP stables.
- **Ingress :** ingress est la couche de routage HTTP, les règles de routage étant définies dans un fichier YAML comme tous les éléments de configuration d'un cluster kubernetes, il utilise **l'ingress controller** qui agit souvent comme un reverse proxy ou un Load balancer à l'image de nginx, traefik ou HAproxy.

Les stratégies de stockage :

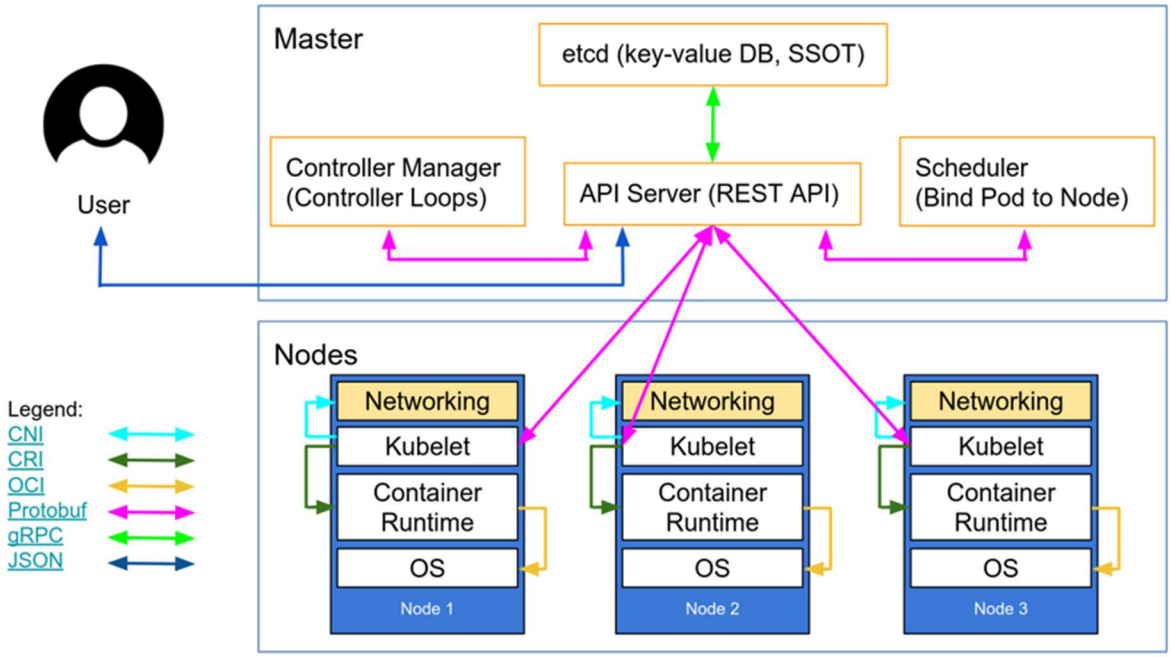
- **Les PV :** sont les ressources physiques du cluster.
- **Les PVC (Persistent Volume Controller) :** les PVC correspondent à des requêtes des conteneurs pour demander un espace de stockage pour le volume du conteneur.

Si l'on devait mettre tous ces outils dans un schéma ça donnerait quelque chose comme ça :



Ici on voit bien que les ETCD sont redondés dans l'infrastructure ce qui est très important étant donné qu'ils contiennent toutes les configurations du cluster.

Maintenant sur une infrastructure **MKS (Managed Kubernetes as a Service)** chez OVH c'est légèrement différent, ça se présenterait plutôt de cette manière.



Détail du schéma :

Encadrement Master :

- **API Server :** Dans ce cas, elle joue le rôle de cœur du cluster, toutes les requêtes et toutes les commandes arrivent sur cette API.
- **ETCD :** Dans ce cas, elle joue le même rôle que dans la définition précédente. **SSOT** veut dire **Single Source of Truth**, c'est pour spécifier que si une information n'est pas dans ETCD, elle n'existe pas pour le cluster.
- **Controller Manager :** Lui, joue le rôle de surveillance du cluster via ce qu'on appelle des boucles de contrôle dans l'exemple d'un cluster de 3 pods, s'il en détecte que deux, il demandera la création du troisième.
- **Scheduler :** il joue le même rôle que dans la définition précédente également, il va vérifier les ressources disponibles sur les trois Nodes et décider où envoyer les nouveaux pods.

Encadrement Node :

- **Kubelet :** Kubelet reçoit les ordres de l'API Server et s'assure que les conteneurs tournent bien sur sa machine (cycle de vie des conteneurs, fait des rapports qu'il envoie au reste du cluster et vérifie la santé des conteneurs.
- **Container Runtime :** C'est lui qui télécharge les images et lance les processus (souvent via containerd)
- **Networking :** c'est là que se passe l'attribution des IP aux pods et la communication entre eux
- **OS :** c'est le système d'exploitation (très souvent une version légère de Linux)

Détails de la légende et des différents protocoles utilisés dans un cluster Kubernetes :

- **CNI (Container Network Interface) :** Quand un pod est créé dans le cluster, le Kubelet appelle ce plugin pour avoir une adresse IP et le connecter au cluster.
- **CRI (Container Runtime Interface) :** c'est ce qui permet la communication entre le Kubelet et le moteur de conteneur, par exemple pour utiliser Docker ou containerd.
- **OCI (Open Container Initiative) :** C'est ce qu'on appelle un standard de format et il permet en fait qu'un conteneur créé sur ton pc fonctionnera partout ailleurs.
- **Protobuf :** C'est un format plus rapide et léger que le JSON pour la communication massive entre le Master et les Nodes.
- **gRPC :** C'est le format de communication entre ETCD et l'API Server ce qui permet une communication ultra rapide pour les écriture de nouvelle configuration par exemple.
- **JSON :** C'est le format standard, c'est que l'utilisateur envoie via ses fichiers YAML/JSON ou ce qui s'affiche dans le dashboard etc.

Comment ça marche concrètement en termes de flux ?

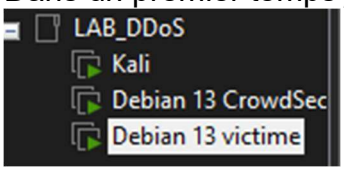
1. L'utilisateur envoie son **fichier YAML/JSON à l'API Server.**
2. **L'API Server** enregistre la demande dans **ETCD via le gRPC.**
3. Le **Controller Manager** voit le **nouveau déploiement** d'un nouveau conteneur.
4. **Le Scheduler** choisit un des Nodes ou il y a des places pour le(s) conteneur(s).
5. **L'API Server** envoie l'ordre au **Kubelet du Node** choisi par le Scheduler via **Protobuf.**
6. **Le Kubelet** demande au **Container Runtime** via le **CRI** de lancer le conteneur.
7. **Le Container Runtime** va utiliser les standards définis par **OCI** pour demander à **l'OS la création du processus.**
8. Et le **Networking** via **CNI** va "brancher" le conteneur récemment ajouté au reste du cluster.

L'après-midi, j'ai décidé de mettre en place un LAB pour ma familiariser avec CrowdSec et les attaques DDoS, j'ai donc mis en place des VM locales, une sous kali linux qui exécutera l'attaque, une sous Debian 13 qui jouera le rôle de protection en exécutant le CrowdSec et une troisième qui sera la victime et qui aura un simple serveur apache. J'ai encore une fois réalisé un schéma réseau pour ce lab que je ne vais pas mettre dans ce compte rendu car le lab n'est pas fini et donc tous les outils qui seront utilisés lors de ce lab ne sont pas présents sur le schéma.

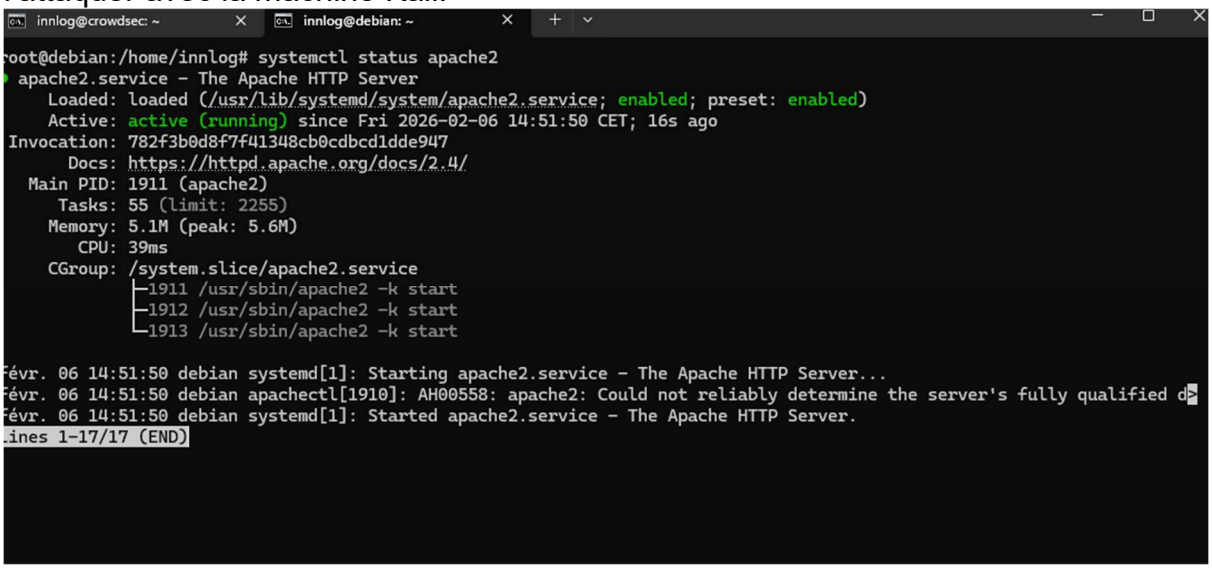
Tout d'abord ce lab était pour apprendre les différences qu'il y a entre un simple fail2ban qui est déjà efficace à son échelle et un CrowdSec qui est beaucoup plus

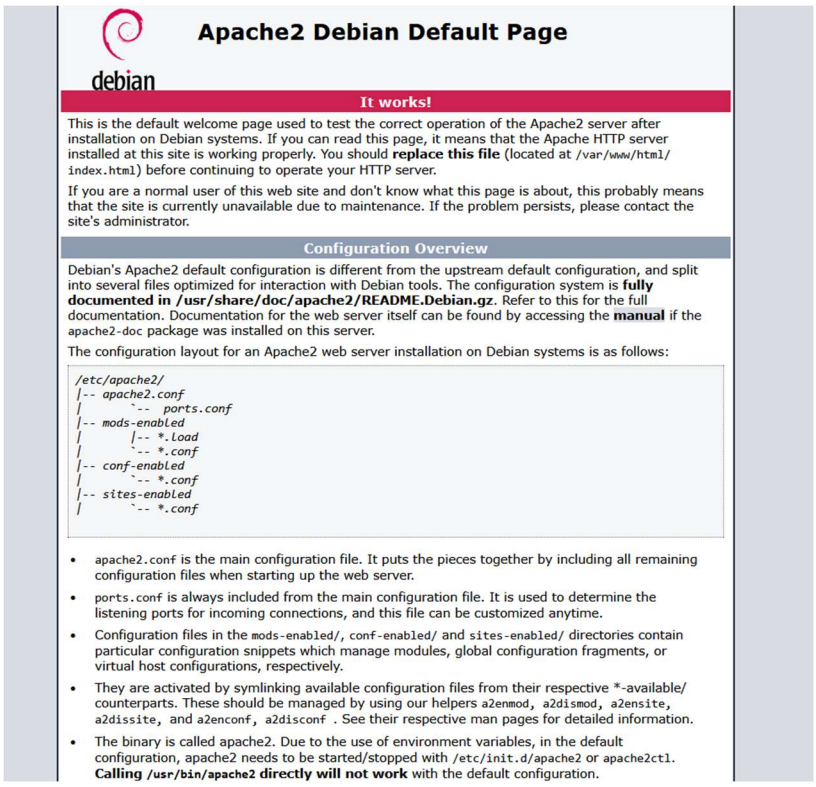
complet et surtout comment peut être lancer une attaque DoS et non pas DDoS dans ce cas car évidemment je ne dispose pas d'assez de moyen pour réaliser ce genre d’attaque grand échelle.

Dans un premier temps j’ai créé mes VM dans VMware en locale.

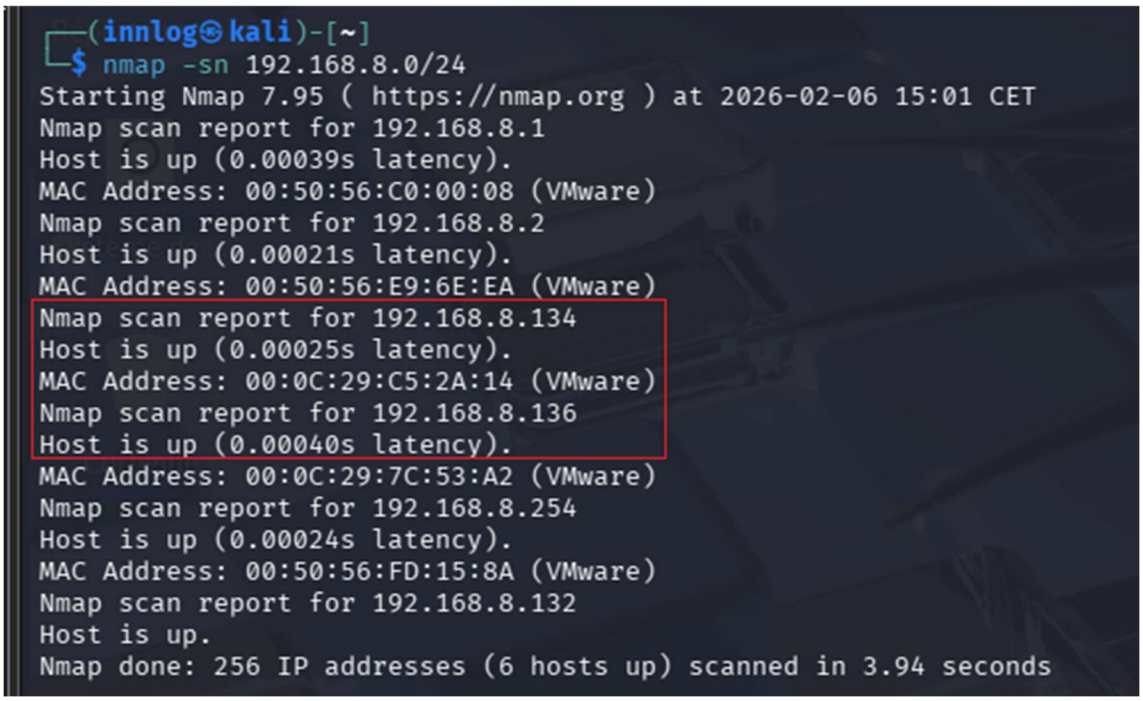


Pour l’instant, j’ai simplement mis en place le serveur apache et essayer de l’attaquer avec la machine Kali.

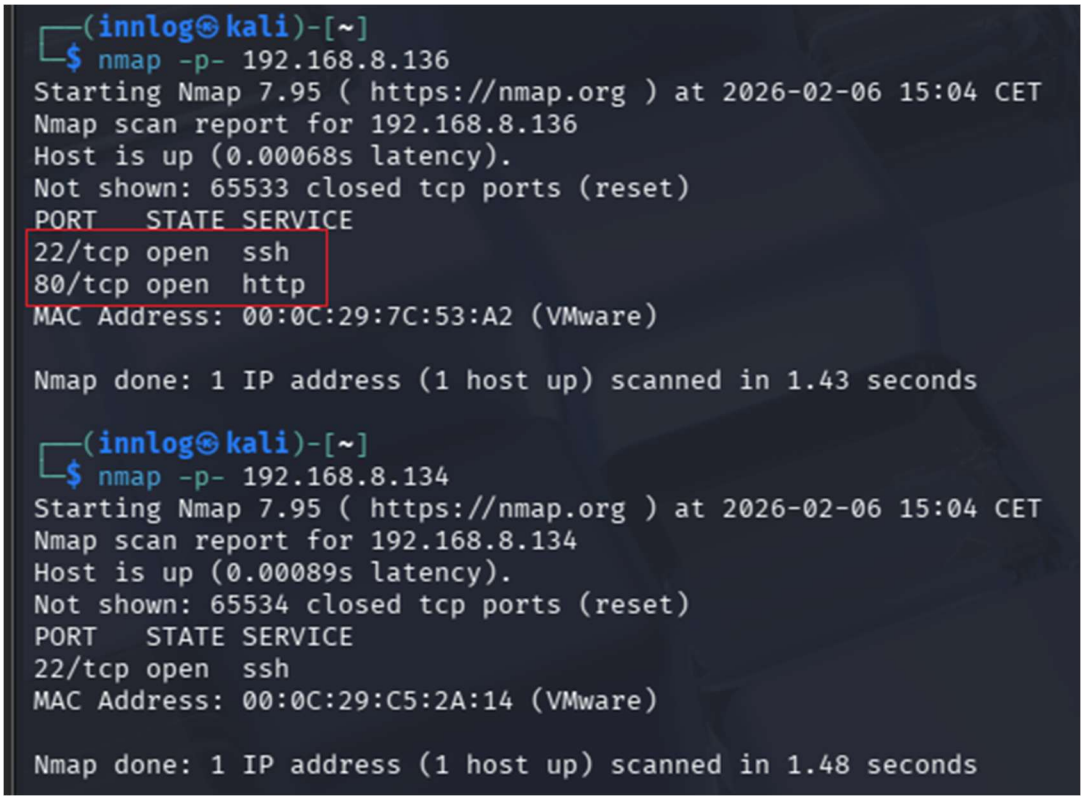




Pour commencer l'attaque, j'ai d'abord commencé par un simple **nmap** afin de détecter les machines présentes sur le réseau. **nmap -sn <adresse réseau> : découverte des hôtes actifs**



Ici on peut voir qu'il a détecté les deux machines, celle qui accueillera le CrowdSec (192.168.8.134) et la machine cible (192.168.8.136). Ce que je fais ensuite c'est scanner les ports ouverts sur les machines. **nmap -p- <ip> : découverte des ports ouverts sur la machine**



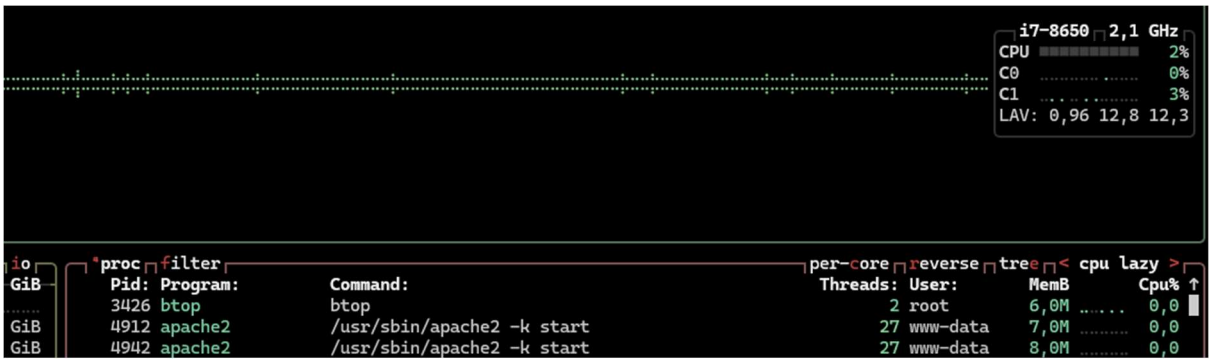
On peut voir qu'il a détecté que les ports 22 (SSH) et 80 (HTTP) sont ouverts sur la machine qui accueille le serveur apache.

Pour ensuite commencer le DoS j'ai utilisé un outil qui s'appelle **WRK** et qui n'est pas natif à Kali mais qui s'installe simplement via **apt.** Cet outil fonctionne avec plusieurs options, les plus importantes étant :

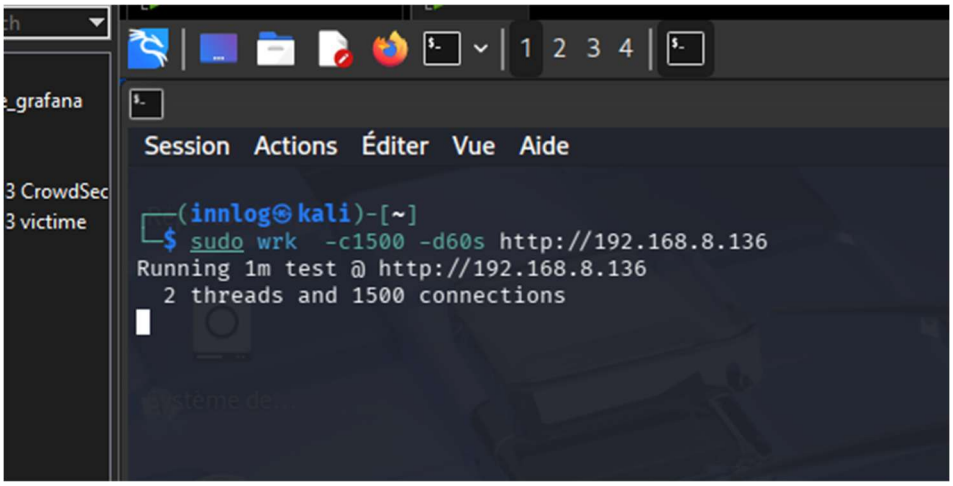
- **-c :** c'est le nombre de connexions HTTP qui resteront ouvertes le temps de l'attaque.
- **-d :** c'est le temps que durera l'attaque.
- **-t :** ce paramètre est extrêmement important car c'est le nombre de **threads** utilisés par le processeur pour réaliser l'attaque, plus le nombre de threads sera élevé plus l'attaque sera brutale (mais la machine doit suivre évidemment). La brutalité de l'attaque dépend également du paramètre **-c** car plus il y a de connexion HTTP ouvertes moins la machine cible à de chance de supporter l'attaque.

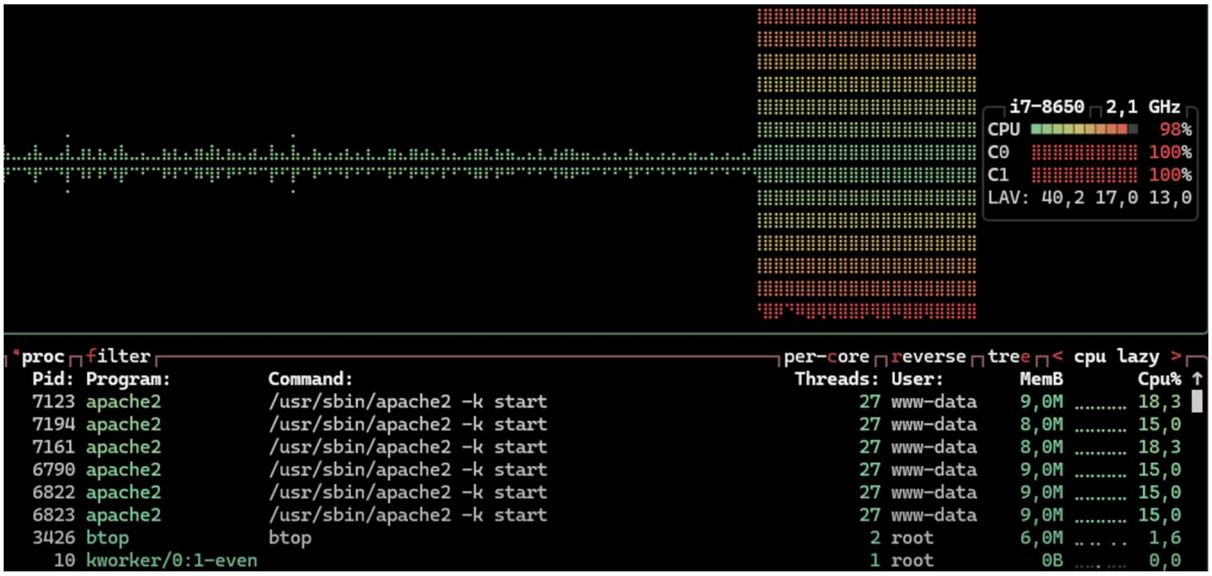
Sur la machine cible, j'ai installé **btop** un petit utilitaire que j'ai mis sur mon raspberry aussi pour surveiller la charge CPU, la RAM, la bande passante, les disques etc.

Alors voici l'état du CPU avant l'attaque :



et une fois l'attaque lancé :





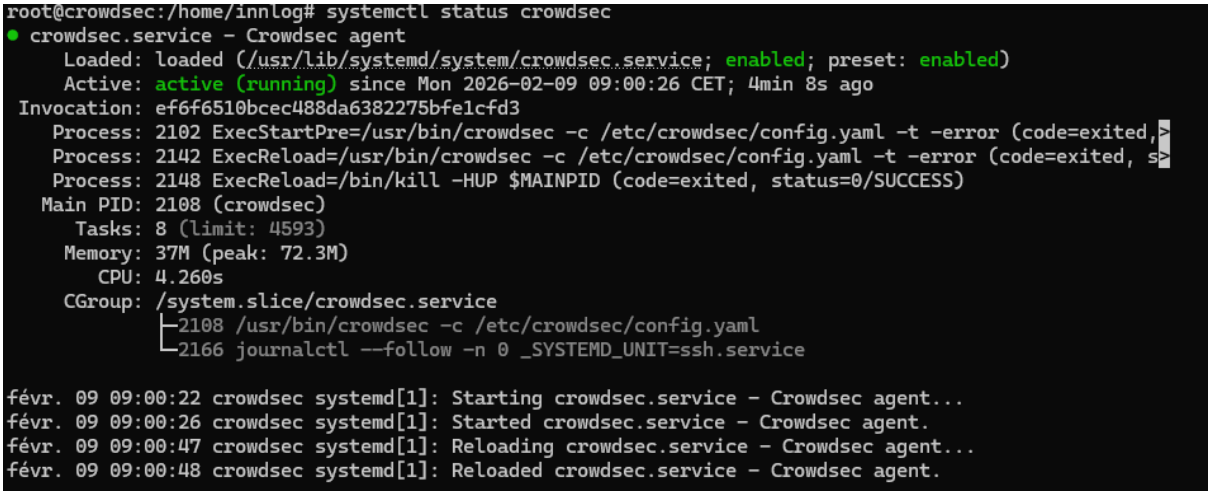
Ici on voit très bien que le CPU est en sueur et que les processus d'apache le sont aussi. L'attaque à donc "réussi", je met entre guillemet car il en faut plus que ça pour faire tomber un serveur web et l'accès à la page par défaut d'apache est encore disponible mais on voit bien quand même que la machine subit une attaque et qu'avec une plus grosse machine et un plus gros CPU sur la machine attaquante, le serveur ne ferait pas long feu, et encore pire si plusieurs machines s'y mettent par exemple avec des botnet, là ça deviendrait une attaque DDoS et non plus une DoS.

La semaine prochaine je n'aurai plus qu'à mettre en place le CrowdSec sur l'autre machine Debian et installer l'agent sur la machine cible de voir les blocages.

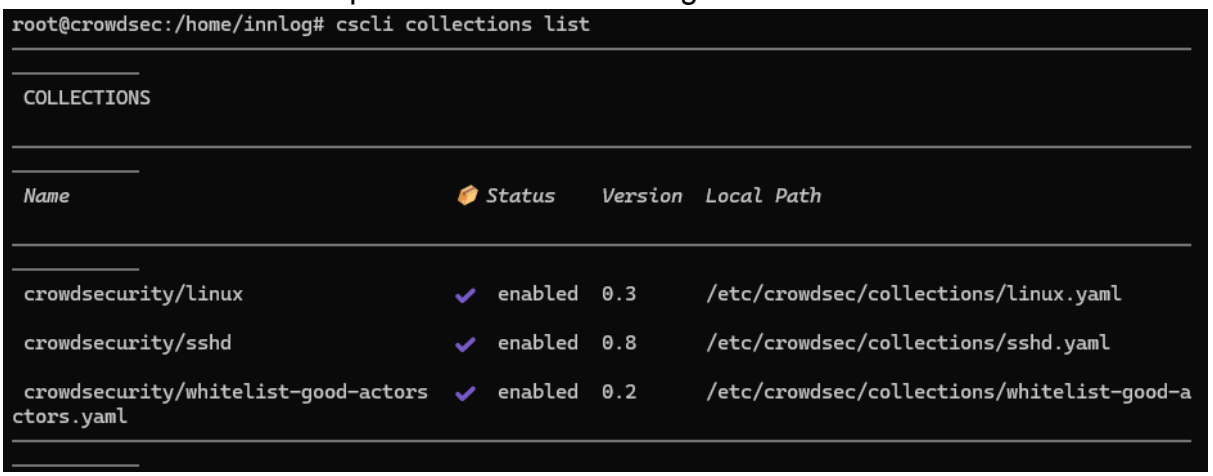
Compte rendu hebdomadaire sixième semaine

Lundi 9 février :

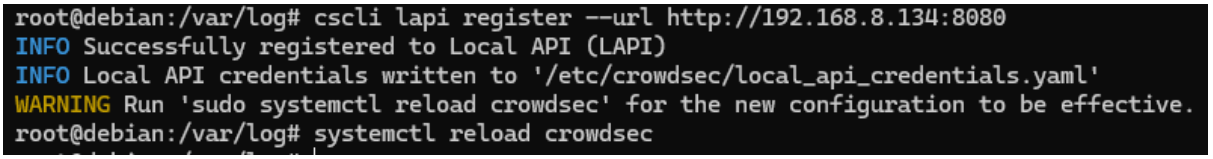
Pour aujourd'hui je suis revenu sur mon LAB DoS + CrowdSec et j'ai installé CrowdSec sur mes machines.



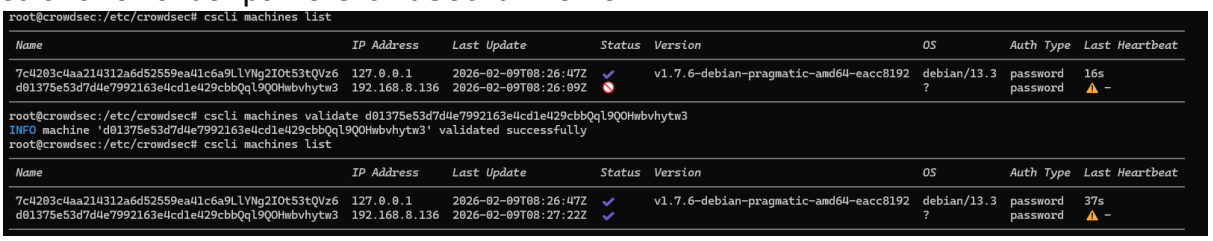
Étant donné que sur la machine crowdsec il y a le service SSH, CrowdSec va par défaut créer des fichiers pour la détection des logs. Des **collections.**



Ensuite il faut enregistrer la machine distante qui va être protégée par notre CrowdSec, dans notre cas la Debian 13.



et la faire valider par le CrowdSec lui même :



Ensuite sur les deux machines, un fichier a été créé **/etc/crowdsec/parsers/s02-enrich/whitelists.yaml** dans ce fichier qui est le fichier de la whitelists comme son nom l'indique il y a des IP enfin plutôt des CIDR qui ne sont par défaut pas bloqués comme 192.168.0.0/16, il faut donc commenter la ligne à ce niveau là.

```
  cidr:
    - "127.0.0.0/8"
#    - "192.168.0.0/16"
#    - "10.0.0.0/8"
#    - "172.16.0.0/12"
  # expression:
  #   - "'foo.com' in evt.Meta.source_ip.reverse"
```

Une fois que ça c'est fait sur les deux machines aussi il faudra installer un **bouncers** une sorte de module en fonction des services que l'on veut protéger sur notre machine donc là il faut installer le bouncers firewall qui va permettre d'en plus détecter les comportements malveillants, les bloqués.
On va l'installer d'abord sur la machine CrowdSec et ensuite générer une clé API que l'on mettra dans la machine victime grâce à cette commande **sudo cscli bouncers add ApacheServer.**

```
root@debian:/etc/crowdsec# cscli bouncers list
```

Name	IP Address	Valid	Last API pull	Type	Version	Auth Type
cs-firewall-bouncer-1770626533	127.0.0.1	✔	2026-02-09T08:43:10Z	crowdsec-firewall-bouncer	v0.0.34-debian-pragmatic-amd64-4144555453620958398aee64253dfd90bbc1f698	api-key

Une fois le bouncers installé, sur la machine CrowdSec il faut aussi l'installer et faire la commande ci-dessus pour générer une clé API afin que les deux machines communiquent via l'API de CrowdSec.

```
root@crowdsec:/etc/crowdsec# cscli bouncers list
```

Name	IP Address	Valid	Last API pull	Type	Version	Auth Type
ApacheServer	192.168.8.136	✔	2026-02-09T09:31:52Z	crowdsec-firewall-bouncer	v0.0.34-debian-pragmatic-amd64-4144555453620958398aee64253dfd90bbc1f698	api-key

Dans la logique des choses il faudra faire écouter l'API sur 0.0.0.0 dans **/etc/crowdsec/config.yaml** car par défaut elle écoute sur 127.0.0.1

```
  server:
    log_level: info
    listen_uri: 0.0.0.0:8080
    profiles_path: /etc/crowdsec/profiles.yaml
```

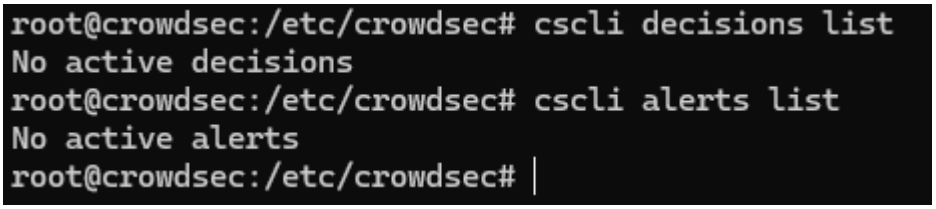
la clé API générer va aller dans ce fichier côté machine victime **/etc/crowdsec/bouncers/crowdsec-firewall-bouncer.yaml**

```
GNU nano 8.4
mode: iptables
update_frequency: 10s
log_mode: file
log_dir: /var/log/
log_level: info
log_compression: true
log_max_size: 100
log_max_backups: 3
log_max_age: 30
api_url: http://192.168.8.134:8080/
api_key: KFT/xSWPhjJvZpLF0M+BlMlgVuy/+sr61nyEsByqhPY
## TLS Authentication
# cert_path: /etc/crowdsec/tls/cert.pem
```

Pour vérifier que la machine pointe bien vers l'API on peut faire un curl :

Le message HTTP 200 OK nous montre bien que l'API écoute et reçoit.

Une fois que les deux machines communiquent on peut essayer une attaque et voir comment le CrowdSec réagi :

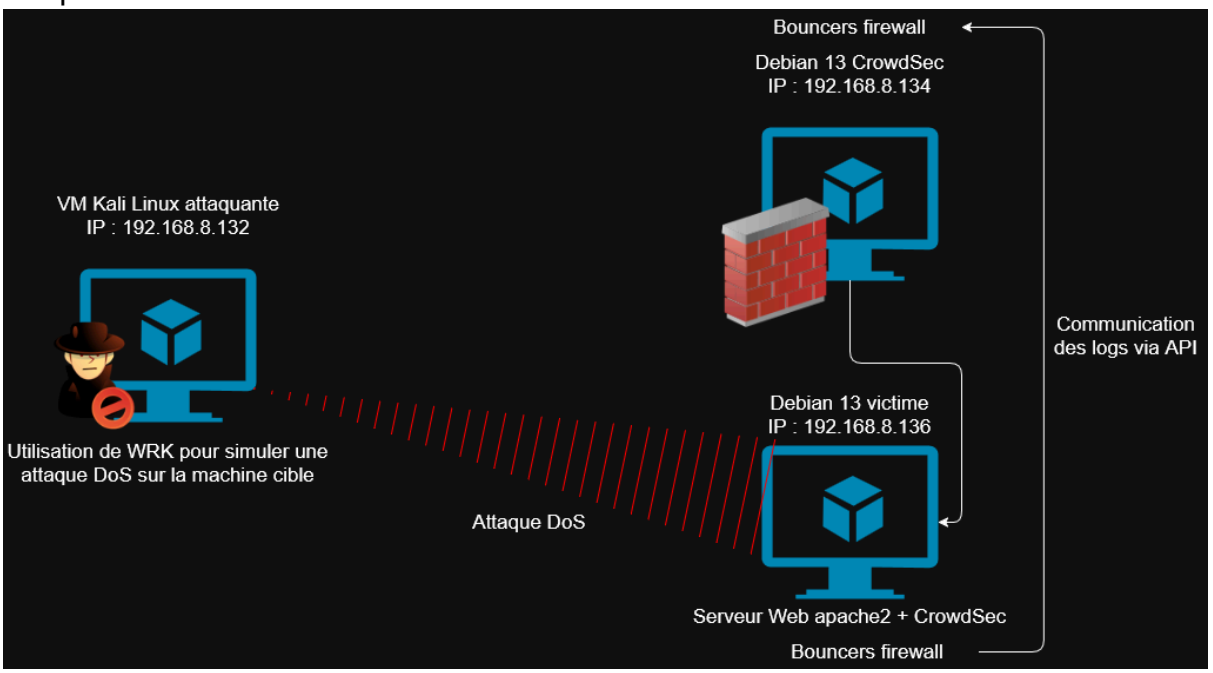


En lançant une attaque via WRK comme la semaine dernière, CrowdSec ne détecte rien et donc ne bloque pas, c'est en fait tout à fait normal. Les paquets qu'envoi WRK sont des HTTP 200 OK ce qui fait qu'il n'y a pas d'erreur juste beaucoup de requêtes à la fois mais ce n'est pas pris en compte par CrowdSec. Il faudrait donc envoyer des requêtes pour l'instant qui génère des erreurs comme celle-ci par exemple :
for i in {1..50}; do curl -I http://192.168.8.136/fake-page-$i; done

C'est une ligne de bash qui pour une boucle de 1 à 50 faire des curl sur le serveur apache2 de page qui n'existe pas ce qui génère donc des erreurs 404, voyons comment CrowdSec interprète ça :

Ici on voit que des alertes ont été reçues par le CrowdSec, il a donc décidé de bannir cette adresse IP qui est l'IP de ma Kali Linux.

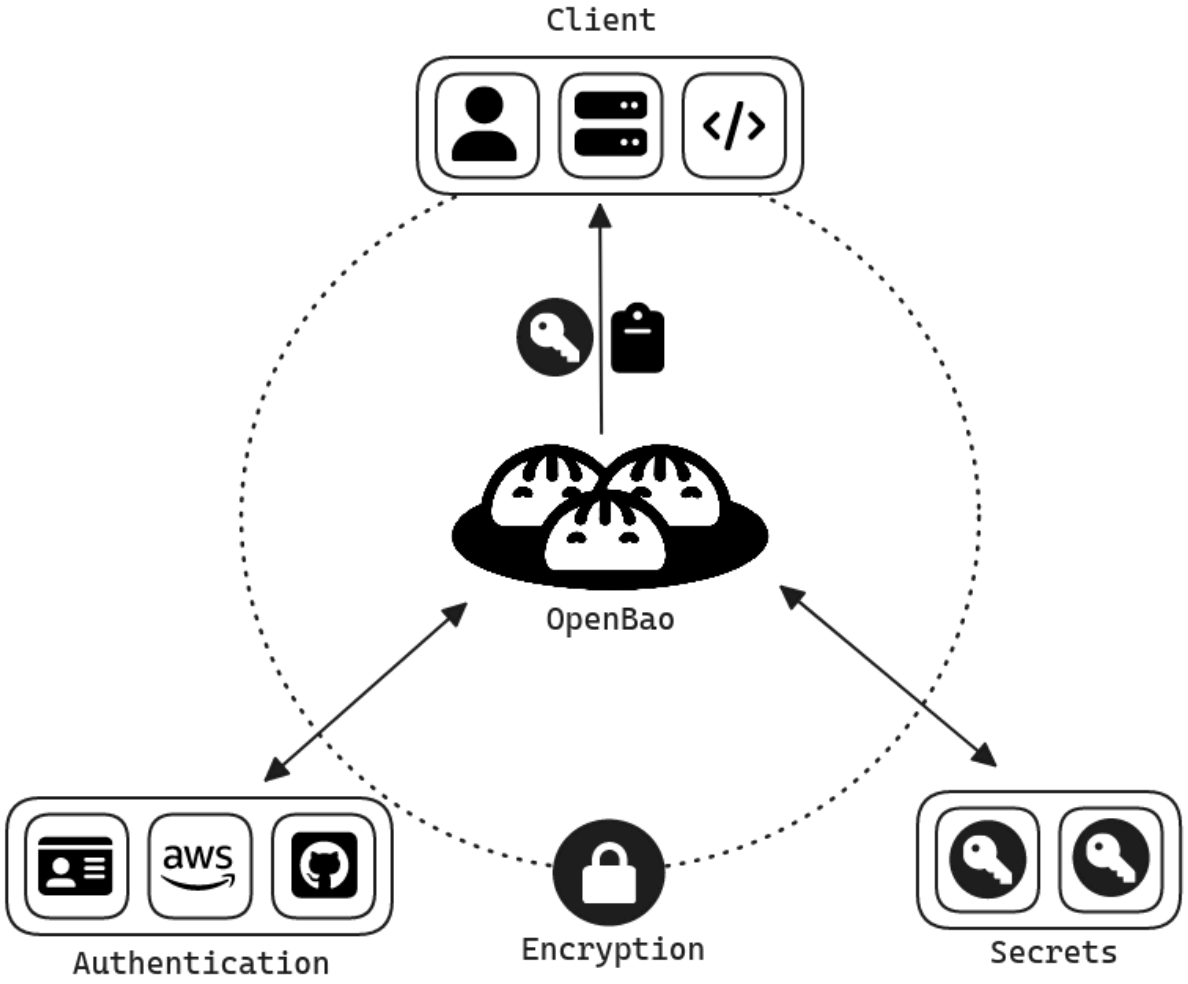
J’ai également profité de la matinée pour refaire mon schéma enfin du moins le compléter.



Mardi 10 février :
En arrivant mardi matin, j’ai fini mon **LAB CrowdSec + attaque DoS**, en effet bloquer des attaques via WRK doit surement être possible mais étant donnée que cette attaque n’envoie que des requêtes **HTTP 200 OK**, c’est très compliqué de pouvoir bloquer ce genre de requête car dans le cas d’une mise en production d’un outil pareil cela voudrait dire que si beaucoup de gens se connectent en même temps sur un site, le serveur va aussi recevoir plusieurs requêtes d’un coup ce qui est plus ou moins équivalent à une attaque WRK mais beaucoup moins violente quand même. **Donc dans ce cas ça voudrait dire que tous les clients qui ont fait ces requêtes seront bannis.**

Maxime m’a donc mis sur l’installation **d’un Vault OpenBao**, qui est un projet open source et un **fork du Vault de HashiCorp** étant donné que c’est devenu propriétaire. C’est un outil qui sert à **stocker des secrets, des mots de passe et des informations sensibles pour des applications.**

```
root@INL-LSO-OpenBao:/etc/openbao# systemctl status openbao
● openbao.service - "OpenBao - A tool for managing secrets"
     Loaded: loaded (/usr/lib/systemd/system/openbao.service; enabled; preset: enabled)
     Active: active (running) since Tue 2026-02-10 09:31:06 CET; 11min ago
 Invocation: df86e511664d4ad8adb30c8fc39b90b1
       Docs: https://github.com/openbao/openbao/tree/main/website/content/docs
   Main PID: 9377 (bao)
      Tasks: 7 (limit: 2302)
     Memory: 16.1M (swap max: 0B, peak: 16.6M)
        CPU: 944ms
     CGroup: /system.slice/openbao.service
             └─9377 /usr/bin/bao server -config=/etc/openbao/openbao.hcl

févr. 10 09:31:06 INL-LSO-OpenBao bao[9377]:               Log Level:
févr. 10 09:31:06 INL-LSO-OpenBao bao[9377]:           Recovery Mode: false
févr. 10 09:31:06 INL-LSO-OpenBao bao[9377]:                 Storage: file
févr. 10 09:31:06 INL-LSO-OpenBao bao[9377]:                 Version: OpenBao v2.5.0, built 2026-02-04T16:19:33Z
févr. 10 09:31:06 INL-LSO-OpenBao bao[9377]:             Version Sha: bcbb6036ec2b747bceb98c7706ce9b974faa1b23
févr. 10 09:31:06 INL-LSO-OpenBao bao[9377]: ==> OpenBao server started! Log data will stream in below:
févr. 10 09:31:06 INL-LSO-OpenBao bao[9377]: 2026-02-10T09:31:06.352+0100 [INFO]  proxy environment: http_proxy="" https_proxy="" no_proxy=""
févr. 10 09:31:06 INL-LSO-OpenBao bao[9377]: 2026-02-10T09:31:06.353+0100 [WARN]  no `api_addr` value specified in config or in BAO_API_ADDR;
févr. 10 09:31:06 INL-LSO-OpenBao bao[9377]: 2026-02-10T09:31:06.353+0100 [INFO]  core: Initializing version history cache for core
févr. 10 09:31:06 INL-LSO-OpenBao systemd[1]: Started openbao.service - "OpenBao - A tool for managing secrets".
```



Au début je ne voyais pas vraiment comment concrètement utiliser cet outil, j'ai donc eu l'idée de **l'utiliser avec mes scripts python** que j'ai utilisés au tout début de mon stage pour récupérer les mots de passe afin de les migrer vers Vaultwarden. **Les trois premiers scripts à utiliser font des curl** et ont donc **besoin de se connecter à l'API de TPM avec le login et le mot de passe** et ces données **étaient en clair dans le script ce qui niveau sécurité des données n'est pas au top.** Ce que je vais faire c'est donc **modifier ces scripts afin qu'ils interrogent l'API d'OpenBao pour qu'ils puissent se connecter** et effectuer les curl.

Il faut d'abord **initialiser le vault**

```
root@INL-LSO-OpenBao:~# bao operator init
Unseal Key 1: [redacted]
Unseal Key 2: [redacted]
Unseal Key 3: [redacted]
Unseal Key 4: [redacted]
Unseal Key 5: [redacted]

Initial Root Token: [redacted]

Vault initialized with 5 key shares and a key threshold of 3. Please securely
distribute the key shares printed above. When the Vault is re-sealed,
restarted, or stopped, you must supply at least 3 of these keys to unseal it
before it can start servicing requests.

Vault does not store the generated root key. Without at least 3 keys to
reconstruct the root key, Vault will remain permanently sealed!

It is possible to generate new unseal keys, provided you have a quorum
of existing unseal keys shares. See "bao operator rotate-keys" for more
information.
```

Cette commande va nous donner 5 clés et un token ROOT qui sont extrêmement importants et qu'il ne faut pas perdre car sinon on perd tous les secrets.

Ensuite il faut **unseal le vault** donc le déverrouiller à l'aide des clés, il faudra rentrer les trois premières clés afin de le déverrouiller jusqu'à ce que le résumé dise **Sealed : false**

```
root@INL-LSO-OpenBao:~# bao operator unseal
Unseal Key (will be hidden):
Key                Value
---                -----
Seal Type          shamir
Initialized        true
Sealed             true
Total Shares       5
Threshold          3
Unseal Progress    1/3
Unseal Nonce       be90812e-d9a2-7bde-1577-11e541fb8c1d
Version            2.5.0
Build Date         2026-02-04T16:19:33Z
Storage Type       file
HA Enabled         false
root@INL-LSO-OpenBao:~# bao operator unseal
Unseal Key (will be hidden):
Key                Value
---                -----
Seal Type          shamir
Initialized        true
Sealed             true
Total Shares       5
Threshold          3
Unseal Progress    2/3
Unseal Nonce       be90812e-d9a2-7bde-1577-11e541fb8c1d
Version            2.5.0
Build Date         2026-02-04T16:19:33Z
Storage Type       file
HA Enabled         false
root@INL-LSO-OpenBao:~# bao operator unseal
Unseal Key (will be hidden):
Key             Value
---             -----
Seal Type       shamir
Initialized     true
Sealed          false
Total Shares    5
Threshold       3
Version         2.5.0
Build Date      2026-02-04T16:19:33Z
Storage Type    file
Cluster Name    vault-cluster-b598cc79
Cluster ID      ac9550c1-a07d-1e14-977f-a2113b9c2879
HA Enabled      false
```

Une fois le coffre déverrouiller, il faut se connecter avec le token ROOT qui nous a précédent été fourni et autoriser le stockage des secrets via la commande **bao kv**

```
root@INL-LSO-OpenBao:~# bao login
Success! You are now authenticated. The token information displayed below is
already stored in the token helper. You do NOT need to run "bao login" again.
Future OpenBao requests will automatically use this token.

Key                  Value
---                  -----
token
token_accessor
token_duration       ∞
token_renewable      false
token_policies       ["root"]
identity_policies    []
policies             ["root"]
```

```
root@INL-LSO-OpenBao:~# bao secrets enable -path=secret kv-v2
Success! Enabled the kv-v2 secrets engine at: secret/
```

À partir de ce moment là nous pouvons stocker des secrets:

```
root@INL-LSO-OpenBao:/etc/openbao# bao kv put secret/tpm login="      " password="      "
= Secret Path =
secret/data/tpm

======= Metadata =======
Key                Value
---                -----
created_time       2026-02-10T10:07:06.5981876492
custom_metadata    <nil>
deletion_time      n/a
destroyed          false
version            1
```

et vérifier si l'entrée à été écrite

```
root@INL-LSO-OpenBao:/etc/openbao# bao kv get secret/tpm
= Secret Path =
secret/data/tpm

======= Metadata =======
Key                Value
---                -----
created_time       2026-02-10T10:07:06.5981876492
custom_metadata    <nil>
deletion_time      n/a
destroyed          false
version            1

====== Data ======
Key         Value
---         -----
login
password
```

Pour effectuer la modification sur le script python il faut d'abord installer la bibliothèque **hvac** via pip.

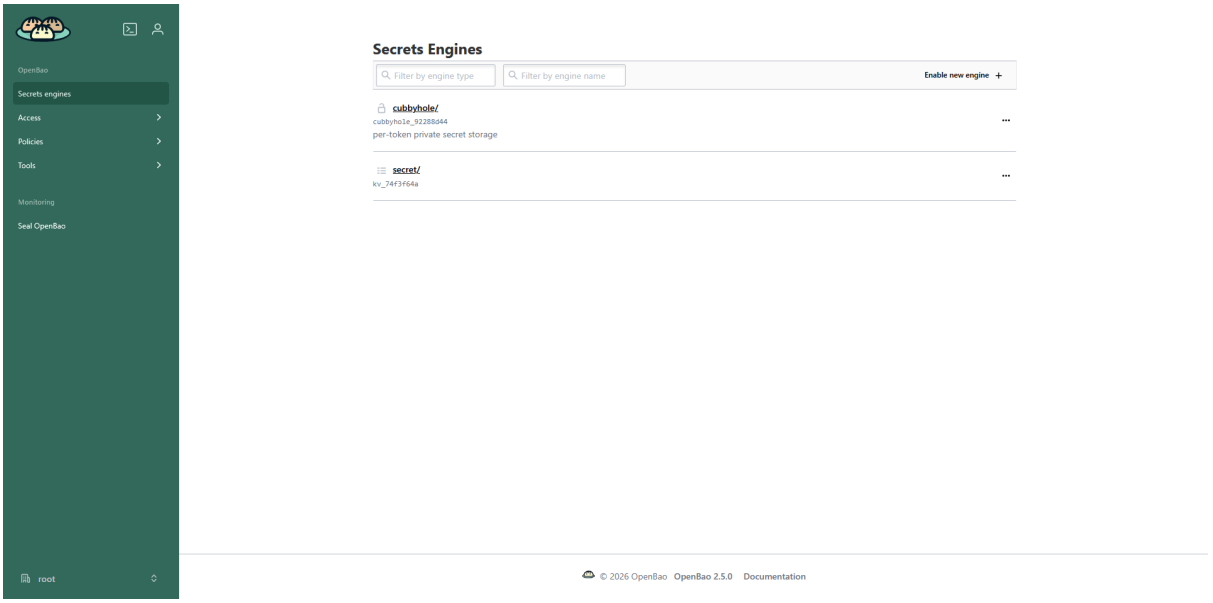
Petit point sur le fonctionnement de cette bibliothèque :
HVAC pour **H**ashiCorp **V**ault **A**pi **C**lient sert dans un script python à la gestion de l'authentification par token, username/password ou encore par annuaire LDAP mais également la manipulation des secrets **lire, écrire ou mettre à jour.** Elle peut aussi servir pour l'administration du serveur en lui-même par exemple en automatisant le déverrouillage du coffre ou encore la gestion des ACL par script.

Il faut ensuite importer la bibliothèque **hvac** dans le script python faire un bloc de configuration de la connexion et une fonction qui fait le call API et qui va récupérer le login et le mot de passe dans le path **secret/tpm**

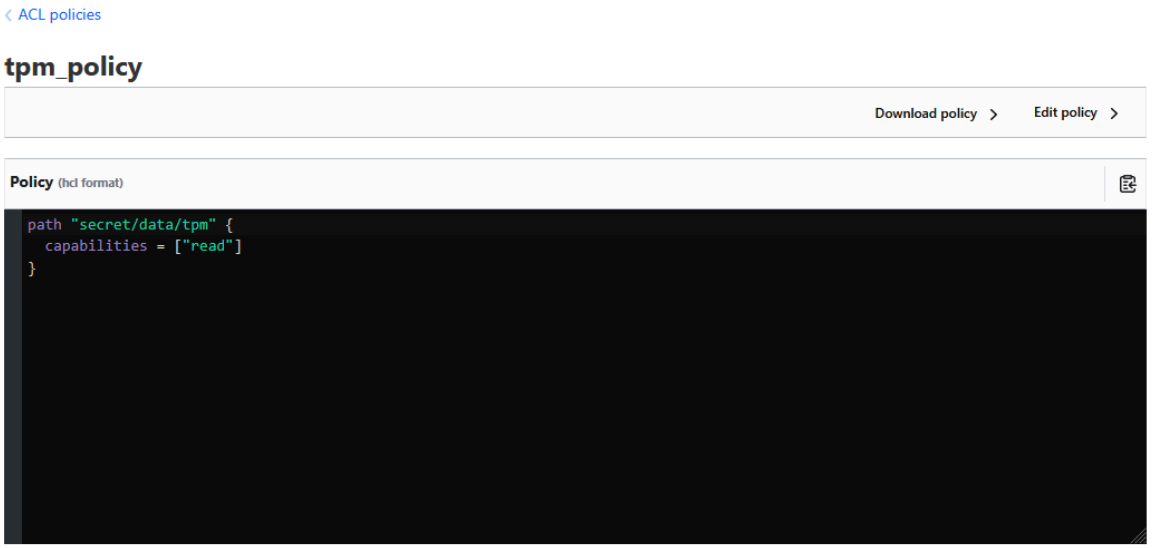
```python
_curl_no_mdp.py > get_tpm_credentials
import asyncio
import aiohttp
import json
import logging
import hvac

# --- CONFIGURATION ---
BASE_URL = "https://[redacted]/index.php/api/v5/passwords/page/"
MAX_WORKERS = 5
MAX_RETRIES = 3

# --- CONFIGURATION OPENBAO ---
BAO_URL = "http://[redacted]:8200"
BAO_TOKEN = "[redacted]"
# ----------------------

logging.basicConfig(
    level=logging.INFO,
    format='%(asctime)s - %(levelname)s - %(message)s',
    handlers=[logging.FileHandler("1_curl_no_mdp.log"), logging.StreamHandler()]
)

def get_tpm_credentials():
    logging.info(f"Connexion à OpenBao sur {BAO_URL}...")
    try:
        client = hvac.Client(url=BAO_URL, token=BAO_TOKEN)
        read_response = client.secrets.kv.v2.read_secret_version(path='tpm', mount_point='secret')

        credentials = read_response['data']['data']
        return credentials['login'], credentials['password']
    except Exception as e:
        logging.error(f"Impossible de récupérer les secrets sur OpenBao : {str(e)}")
        raise
```

Le seul problème pour l'instant est que j'expose le ROOT token dans le script pour la connexion à l'API il faudra donc trouver un moyen d'anonymiser ça également.

Il y a également une interface web pour OpenBao afin de gérer ses secrets autrement qu'en ligne de commande.



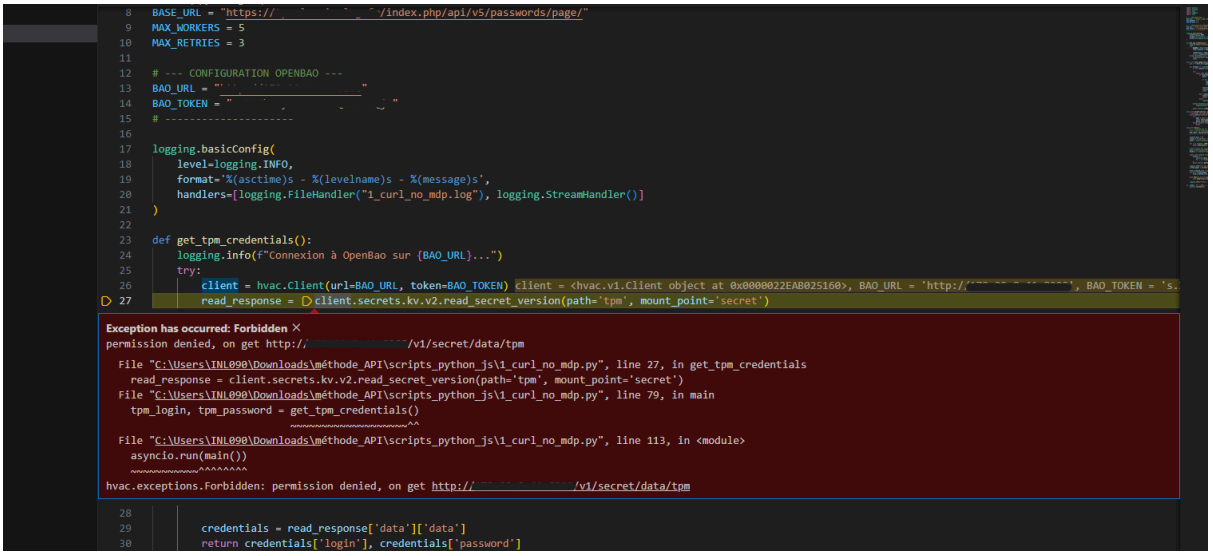
Pour régler le problème du ROOT token, j'ai créé une ACL dans OpenBao qui ne procure que les droits de lecture sur ce secret en particulier grâce à son **path secret/data/tpm** et non plus des droits root.



et ensuite il suffit de taper cette commande pour permettre de générer un token qui applique cette ACL et qui n'est utilisable qu'une seule fois et qui valable pendant 5 minutes

```
bao token create -policy=tpm_policy -use-limit=1 -ttl=5m
```

C'est à dire que si l'on relance le script une deuxième fois avec le même token on a une erreur Forbidden.



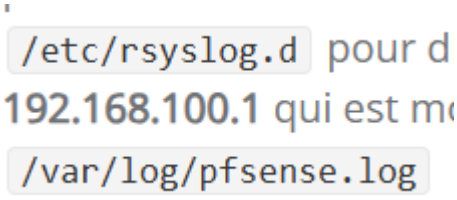
Pour éviter que la tâche soit trop répétitive j'ai fait un script bash qui précise le serveur ou doit pointer le call API et ensuite qui exécute la commande en ne prenant que la ligne ou se trouve le token grâce à **grep** et **awk** vérifie la contenance de la chaîne de caractère et l'affiche.

Car comme me l'a dit un de mes profs un jour : **"Si tu dois le faire deux fois, fait un script."**

```
#!/bin/bash

#Adresse du serveur + port
export BAO_ADDR='http://<ip de la machine OpenBao>:8200'

#Création du token l'utilisation de awk pour ne prendre que le token
TOKEN=$(bao token create -policy=tpm_policy -use-limit=1 -ttl=5m -format=table | grep "^token\s" | awk '{print $2}')

#Affichage du token
if [ -z "$TOKEN" ]; then
    echo "Erreur : Impossible de générer le token."
else
    echo ""
    echo "=== TOKEN à placer dans le script python ==="
    echo "$TOKEN"
    echo ""
    echo "ATTENTION, ce token n'est valable que 5 minutes et utilisable qu'une seule fois à partir de sa création"
    echo ""
fi
```

Mercredi 11 février :
Alors aujourd'hui Maxime était très occupé, j'en ai donc profité pour avancer sur mon portfolio car j'étais en retard de deux semaines sur mon article de stage. J'en ai également profité pour améliorer des choses qui ne me plaisaient pas auparavant comme le style des blocs de code que je mettais dans mes articles.

ils ressemblent maintenant à ça, avant il y avait un gros espacement entre le code et les bords haut/bas du bloc maintenant il n'y en a plus, la barre de scroll paraît beaucoup plus moderne.



J'ai également essayé de mieux faire ressortir les chemins et les services quand j'en parle dans mes articles.

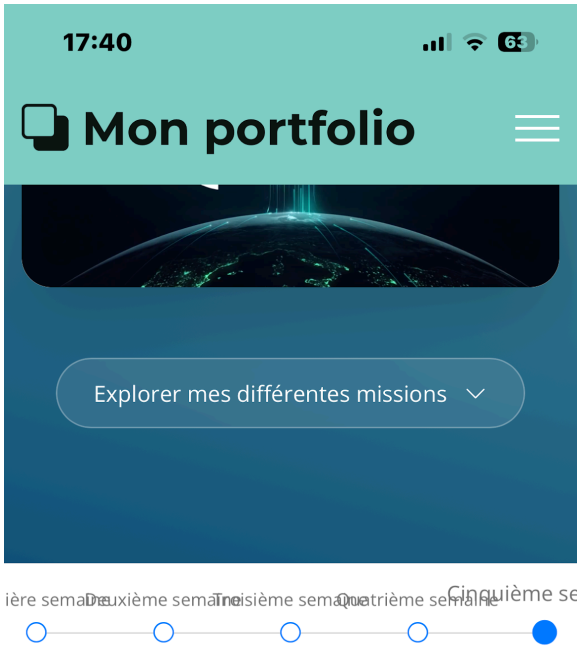
Il y avait aussi un problème qui n'était pas propre, dans l'URL l'extension de fichier était présente donc ça faisait quelque chose du genre **https://enzo-martin.fr/stage-innlog.php**

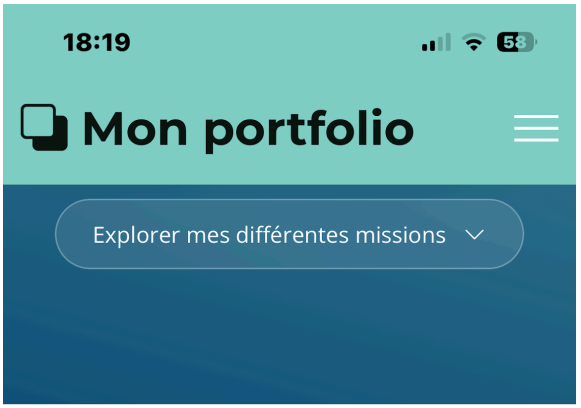
donc maintenant avec un simple fichier .htaccess, l'extension a disparu et c'est beaucoup plus propre.

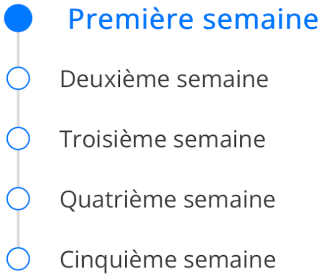
Ensuite j'ai réglé un gros problème de responsive qu'il y a avait sur la page de mon article de stage sur téléphone. En effet le premier problème était que quand on glissait le doigt dans n'importe quelle direction la page bougeait et ne restait pas droite ce qui est très désagréable lors de la navigation.

Le deuxième problème était que les écritures, les photos et les bouts de code touchait les bords de l'écran comme ceci :

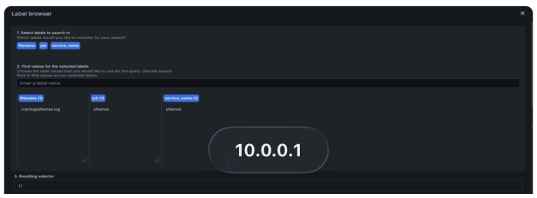
Mon troisième problème et peut être le plus gros était que la barre de la timeline n'était pas du tout responsive sur téléphone et pas lisible.







Pour la suite de la stack Grafana, j'ai fait remonter **label pfsense** dans Grafana ce qui m'a permis **d'afficher les premier logs du pfsense** dans un format pas très lisible mais qu'il faudra filtrer et parser avec les bon **LogQL**

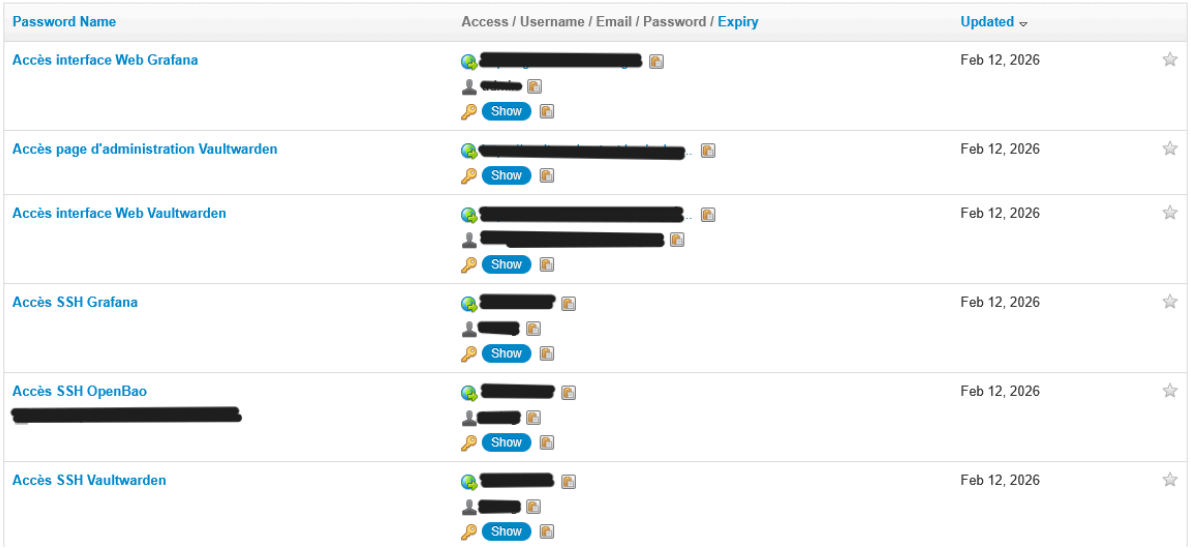


Migration d'une base de données de mots de passe

Lors de mon arrivée chez **Innlog**, j'ai dû m'approprier rapidement leurs outils internes. L'entreprise utilisait historiquement **TPM** comme

vers **Vaultwarden**, une alternative

Jeudi 12 février :
Jeudi étant donné que c'est l'avant dernier jour de mon stage j'ai mis les mots de passe et les accès dans le TPM de Team IT pour qu'ils aient accès à l'avenir aux machines et aux services que j'ai déployé durant toute la durée de mon stage.



De plus en revenant sur le Vaultwarden car j'ai invité un autre collègue du service dans le coffre je me suis rendu compte que la permissions **Afficher les éléments, les mots de passe cachés** n'était pas bonne, en effet, cette permissions est je pense mal traduite ou implicitement mal expliqué mais elle veut en fait dire que la personne ne peut pas voir les mots de passe il fallait donc remettre les permissions **Afficher les éléments** seulement, j'ai donc rebalancer le script JavaScript dans la console pour modifier les éléments et la il ne marchait pas. J'ai donc relu mon code JavaScript et il fallait changer quelques petites choses.

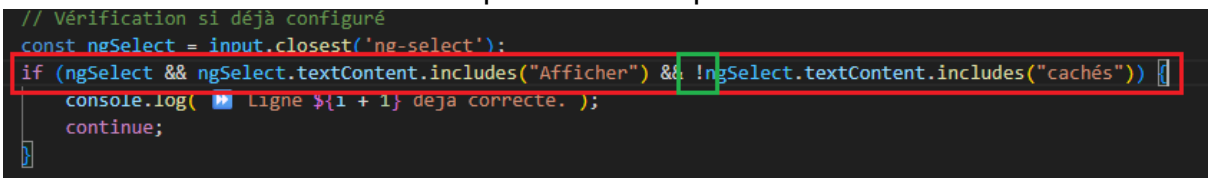
Avant cette ligne vérifiait si l'option sélectionnée contenait **"Afficher"** et **"cachés"** pour prendre l'option **"Afficher les éléments, les mots de passe cachés".**

```
const ngSelect = input.closest('ng-select');
if (ngSelect && ngSelect.textContent.includes("Afficher") && ngSelect.textContent.includes("cachés")) {
    console.log(`⏭ Ligne ${i + 1} déjà correcte.`);
    continue;
```

En plus de cette ligne, il y avait un **try** plus tard dans le code qui prenait l'option qui commençait pas **"Afficher les éléments, les"** ce qui encore une fois prenait la bonne option.

```
try {
    console.log(`🔄 Ligne ${i + 1}/${inputs.length}`);
    input.scrollIntoView({ block: 'center' });

    input.focus();
    await attendre(200);

    input.value = "Afficher les éléments, les";
```

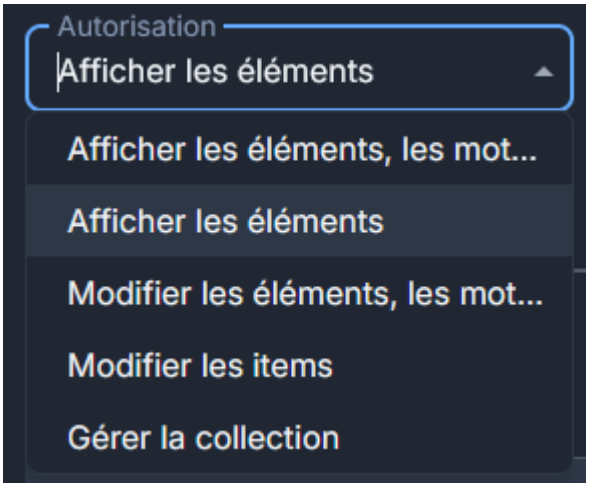
Pour modifier tout ça et repasser les permissions en **“Afficher les éléments”** il fallait mettre un **!** devant le premier **if** pour dire au programme si quelque chose contient **“Afficher”** et ne contient pas **“cachés”** prend la.



Mais il fallait également modifier le **try** et enlever le **“, les”** pour qu’il s’arrête à **“Afficher les éléments”**

Vendredi 13 février :
Ce matin je me suis un peu plus penché sur les différentes permissions qui étaient applicables sur les groupes que l’on crée dans Vaultwarden pour maîtriser l’outil encore un peu plus. Et en faisant des tests je me suis rendu compte des différences subtiles qu’il y avait entre ces différentes permissions.

Dans Vaultwarden quand on crée un groupe dans une organisation il y a **cinq permissions** différentes que l’on peut appliquer au groupe :



Afficher les éléments, les mots de passe cachés : l'utilisateur peut voir les éléments d'une entrée mais pas le mot de passe. Et ne peut pas créer de nouvelles entrées dans les collections de l'organisation.

Afficher les éléments : l'utilisateur peut voir les éléments d'une entrée et le mot de passe. Et ne peut pas créer de nouvelles entrées dans les collections de l'organisation.

Modifier les éléments, les mots de passe cachés : l'utilisateur peut modifier une entrée mais pas son mot de passe. Et peut créer de nouvelles entrées dans les collections de l'organisation.

Modifier les items : l'utilisateur peut modifier une entrée et son mot de passe. Et il peut créer de nouvelles entrées dans les collections de l'organisation.

Gérer la collections : l'utilisateur a tous les droits sur les collections, créer, supprimer etc.

Si on mixe ces permissions avec les rôles que l’on peut attribuer aux membres, on se retrouve avec un système similaire à **AGDLP.**