



# KEY MANAGER

Installation d'un gestionnaire de clés Hashicorp Vault  
dans le contexte financeur moB





## Version du document

Auteur(s)		Vérificateur(s)		Approbateur(s)	
Nom	Date/Visa	Nom	Date/Visa	Nom	Date/Visa
SADAoui W.	27/07/2022			GIFFARD A.	09/11/2022

## Diffusion

Pour validation	Pour information

## Historique

Version	Date	Auteur	Description
1.0	25/07/2022	Walid SADAoui	Initialisation du document
1.1	27/07/2022	Walid SADAoui	Configuration MS Azure
1.2	07/09/2022	Walid SADAoui	Certificats et configuration
1.3	14/10/2022	Walid SADAoui	Vault Multifinanceur
1.4	09/11/2022	Arnaud GIFFARD	Publication Opensource
1.5	25/01/2023	Walid SADAoui	Installation offline de packages linux

# Sommaire

<b>1. INTRODUCTION</b>	<b>4</b>
<b>2. FONCTIONNEMENT DU KEY MANAGER</b>	<b>4</b>
<b>3. DESCRIPTION DU LIVRABLE</b>	<b>5</b>
3.1. FICHIERS FOURNIS	5
3.2. LISTE DES SERVICES	5
3.2.1. <i>Service vault principal</i>	5
3.2.2. <i>Service vault-init</i>	6
3.2.3. <i>Service vault-cron</i>	6
3.3. SCHEMA D'ARCHITECTURE	7
<b>4. VUE D'ENSEMBLE DES ACTIONS REQUISES</b>	<b>7</b>
4.1. ELEMENTS A RECUPERER AUPRES DE MOB	7
4.2. ACTIONS A REALISER PAR LE FINANCEUR	7
<b>5. CREATION DES CERTIFICATS</b>	<b>8</b>
<b>6. CONFIGURATION DU SERVEUR VAULT</b>	<b>8</b>
<b>7. DEMARRAGE DU SERVEUR VAULT</b>	<b>9</b>
<b>8. ACTIONS DE SECURISATION POST-INSTALLATION</b>	<b>10</b>
8.1. RECUPERATION DES CLES DE DESCELLEMENT	10
8.2. APPLICATION DE CONTRAINTES SUR LES ROLES DE CERTIFICATS	10
<b>9. MONITORING</b>	<b>11</b>
<b>10. CONFIGURATION POSTE UTILISATEUR</b>	<b>12</b>
10.1. ENREGISTRER LES CERTIFICATS DANS LE MAGASIN DE CERTIFICATS	12
10.1.1. <i>Enregistrer le certificat Manager dans les certificats personnels</i>	12
10.1.2. <i>Enregistrer le certificat de l'autorité de certification dans les autorités de certification racines de confiance</i>	14
<b>11. DEPLOIEMENT SOUS WINDOWS</b>	<b>18</b>
11.1. UTILISATION DE DOCKER WSL	18
11.2. SECURISATION D'UN ENVIRONNEMENT	18
<b>12. CONFIGURATION MANUELLE DU VAULT (SI SOUHAITE)</b>	<b>19</b>
12.1. CONFIGURATION COMPLETE, SANS LES SERVICES FOURNIS	19
12.1.1. <i>En utilisant la CLI du Vault</i>	19
12.1.2. <i>En utilisant l'interface graphique du vault</i>	20
12.2. CONFIGURATION COMPLEMENTAIRE, A PARTIR DES SERVICES FOURNIS	26
12.2.1. <i>Données générées</i>	26
12.2.2. <i>Rotation de la clé de chiffrement</i>	28
12.2.3. <i>Envoi de la clé publique à moB</i>	29
<b>13. ENVOYER UNE CLE PUBLIQUE A MOB (HORS VAULT)</b>	<b>30</b>
13.1. OBTENTION DU TOKEN D'AUTHENTIFICATION	30
13.2. ENVOI DE LA CLE PUBLIQUE A MOB	31
<b>14. TROUBLESHOOTING</b>	<b>33</b>
14.1. PROBLEMATIQUES RESEAU	33
14.1.1. <i>Accès au Vault derrière un proxy</i>	33
14.1.2. <i>Installation hors-ligne des packages linux dans le conteneur Vault</i>	33

14.2.	NETTOYAGE DE COMPOSANTS DOCKER .....	34
14.3.	ACCES INTERNET DANS LES CONTENEURS DOCKER .....	34
14.4.	MAUVAIS CHOIX DE CERTIFICAT .....	34
14.5.	FIREFOX NE PROMPTE PAS POUR SELECTIONNER LE CERTIFICAT CLIENT .....	34
14.6.	APPELS API VAULT .....	35
14.7.	TESTER LE VAULT EN LOCAL .....	35
14.8.	UTILISATION D'UN PROXY TRAEFIK .....	36
<b>15.</b>	<b>REFERENCES .....</b>	<b>36</b>

# 1. Introduction

La plateforme moB permet aux citoyens de faire des souscriptions à des aides proposées par des financeurs. Afin que les financeurs puissent traiter ces souscriptions, les citoyens peuvent joindre des justificatifs pour prouver leur éligibilité aux différentes aides proposées.

Pour respecter la réglementation RGPD, seuls les utilisateurs financeurs autorisés doivent être capables de consulter ces justificatifs, toute personne ou tout système extérieur ne doit pas avoir la possibilité d'accéder au contenu de ces documents pouvant être sensibles.

Pour répondre à cette problématique, la solution choisie a été de chiffrer tous les justificatifs fournis par les citoyens et de ne donner la possibilité de déchiffrer ces documents qu'aux personnes autorisées.

Pour implémenter ce besoin et accélérer la mise en œuvre dans le SI du financeur lors de l'expérimentation, nous avons choisi de proposer un Key Manager basé sur la solution Open source [Hashicorp Vault](#).

## 2. Fonctionnement du Key Manager

Le Key Manager est une solution qui est fournie aux financeurs ne possédant pas leur propre solution pour stocker les clés de chiffrement, qui seront utilisées pour chiffrer les justificatifs transmis par un citoyen lors de la souscription à une aide d'un financeur.

La mise en place du Key Manager et en particulier la transmission d'une première clé publique à MOB est un prérequis pour autoriser un citoyen à souscrire à une aide proposée par un financeur.

C'est une solution générique fournie par moB qui se veut agnostique, elle est basée sur les principes suivants : Open Source, réutilisabilité, portabilité, agnosticité.

Nous recommandons en ce sens une installation sous Linux.

Une installation Windows reste fonctionnelle mais il revient alors au financeur d'ajuster l'installation pour répondre aux enjeux de sécurité. Des [indications sont fournies](#) dans ce manuel en ce sens.

Le financeur est responsable de la mise en place des différents processus de sécurisation des accès au Key Manager, conformément à leurs politiques de sécurité internes.

Le Key Manager fonctionne suivant les processus principaux ci-dessous :

- Au lancement, un script d'initialisation s'exécute pour déverrouiller (ou desceller) le Vault pour générer un couple de clés publique/privée rsa-2048 et envoyer la clé publique à moB via appel API (flux sortant Internet). A noter, on considère qu'un couple de clés expire au bout de 6 mois à partir de sa date de création.
- Une tâche planifiée récurrente (cron) est aussi déclenchée une fois par semaine et va vérifier la date d'expiration de la dernière clé stockée et générer une nouvelle version de la clé si la date d'expiration est dans moins de 2 semaines puis envoyer la nouvelle version de la clé publique à moB.

La scalabilité de la solution est bien possible avec cette solution mais n'est pas traitée par ce manuel ni le livrable, dans le cadre de l'expérimentation. Ce sujet est à prendre en main par le Financeur si besoin, dans l'objectif d'une intégration pérenne au SI.

## 3. Description du livrable

Le livrable est une archive zip nommé **mcm-vault-vX.Y.Z.zip**.

### 3.1. Fichiers fournis

- ❓ Un fichier **Dockerfile** créée à partir de l'image docker **vault:1.11.3** et enrichi des différentes configurations spécifiques au contexte MOB
- ❓ Un fichier **vault-docker-compose.yml** qui permet de lancer le Vault à l'aide de 3 services :
  - o Le service **vault** principal
  - o Un service **vault-init** qui permet d'initialiser le Vault et de créer un couple de clés
  - o Un service **vault-cron** qui lance un cronjob pour renouveler le couple de clés publique/privée périodiquement et envoie la nouvelle version de la clé publique à MOB
- ❓ Un script d'initialisation du Vault : **init-vault.sh**
- ❓ Un script de renouvellement de clé : **renew-key.sh**
- ❓ Un fichier de configuration du Vault : **config.hcl**
- ❓ Un fichier policy pour les autorisations à accorder aux utilisateurs qui auront les accès admin au Vault : **admin-policy.hcl**
- ❓ Un fichier policy pour les autorisations à accorder aux utilisateurs connectés au Vault en tant que gestionnaire à l'aide d'un certificat client : **manager-policy.hcl**
- ❓ Un crontab qui permet de lancer le script de renouvellement de clé tous les samedis à 3h du matin : **vault-crontab**
- ❓ Un script permettant de générer un certificat avec une autorité de certification, à utiliser en cas de besoin : **createCertificates.sh**
- ❓ Un fichier contenant la liste des variables d'environnement à renseigner, à renommer en **.env** : **.env.sample**

### 3.2. Liste des services

La solution de Vault proposée est composée de plusieurs services permettant d'automatiser le renouvellement et la transmission des clés de chiffrement à moB. Ces services sont détaillés dans le fichier **vault-docker-compose.yml**.

#### 3.2.1. Service vault principal

Le service **vault** est le service qui démarre le Vault à l'aide de la commande **vault server**. Le serveur Vault se lance en utilisant la configuration se trouvant dans le fichier **/vault/config/config.hcl**.

Le Vault écoute par défaut sur le port 8200 en HTTPS. Le storage 'file' est utilisé dans le fichier **config.hcl** comme configuration par défaut. Toutes les données du Vault sont stockées dans le conteneur du Vault, dans le dossier **/vault/file/data**. Un volume docker persistant est associé au dossier **/vault/file** du Vault. Vous pouvez :

- modifier le type de stockage dans le fichier **config.hcl** pour utiliser le type de stockage souhaité.
- Modifier le port souhaité d'écoute dans le

### 3.2.2. Service vault-init

Le service **vault-init** lance le script **init-vault.sh** qui va réaliser plusieurs actions permettant de configurer le Vault et d'envoyer la première clé MOB :

- ❑ Fonction **init** : au premier lancement, il faut initialiser le Vault, c'est-à-dire préparer le Vault à recevoir les données sur son storage. Une clé root cryptée est générée et stockée dans le fichier **/vault/file/keys** avec la liste de clés de déscllement permettant de décrypter la clé root. 5 « **Unseal Keys** » sont générées, et il en faut 3 pour desceller le Vault.
- ❑ Fonction **unseal** : permet de déverrouiller/désceller le vault à l'aide des clés stockées dans le fichier **/vault/file/keys**
- ❑ Fonction **root\_log\_in** : connexion au Vault CLI à l'aide du token root généré par le Vault
- ❑ Fonction **enable\_cert** : autorisation de l'authentification par certificat
- ❑ Fonction **create\_policy** : création d'une policy admin (**admin-policy.hcl**) et d'une policy manager (**manager-policy.hcl**) permettant d'attribuer les droits d'admin ou de manager
- ❑ Fonction **create\_cert\_role\_admin** : création d'un rôle admin d'authentification par certificat avec la policy admin
- ❑ Fonction **create\_cert\_role\_manager** : création d'un rôle manager d'authentification par certificat avec la policy manager
- ❑ Fonction **create\_token** : création d'un token d'authentification qui permet de ne pas utiliser le token root précédent qui a tous les droits. La policy admin y est associée pour restreindre les droits d'accès associés à ce token
- ❑ Fonction **cert\_log\_in** : connexion au Vault à l'aide du certificat admin créé dans la fonction **create\_cert\_role\_admin**
- ❑ Fonction **setup\_cors** : permet de configurer le CORS pour le vault. Dans notre cas autoriser toutes les origines. Vous pouvez adapter les règles à votre propre politique.
- ❑ Fonction **enable\_transit** : autorisation de l'engine transit qui permet de générer des clés de chiffrement
- ❑ Fonction **enable\_secrets** : autorisation de l'engine secrets qui permet de stocker des secrets de type key/value.
- ❑ Fonction **create\_key\_pair** : permet de générer un couple de clés de type rsa-2048 exportable et de l'envoyer à MOB

En cas d'erreur dans le script **init-vault.sh**, le service **vault-init** se relance automatiquement avec un délai d'attente de 15 minutes.

### 3.2.3. Service vault-cron

Un service **vault-cron** est lancé avec le service vault principal et contient une tâche cron qui va se lancer une fois par semaine dans le but de renouveler la clé de chiffrement si elle expire dans moins de 2 semaines (168 jours après la date de création de la clé). Il est configuré pour se lancer tous les samedis à 3h du matin.

Le script **renew-key.sh** est le script lancé par la tâche cron :

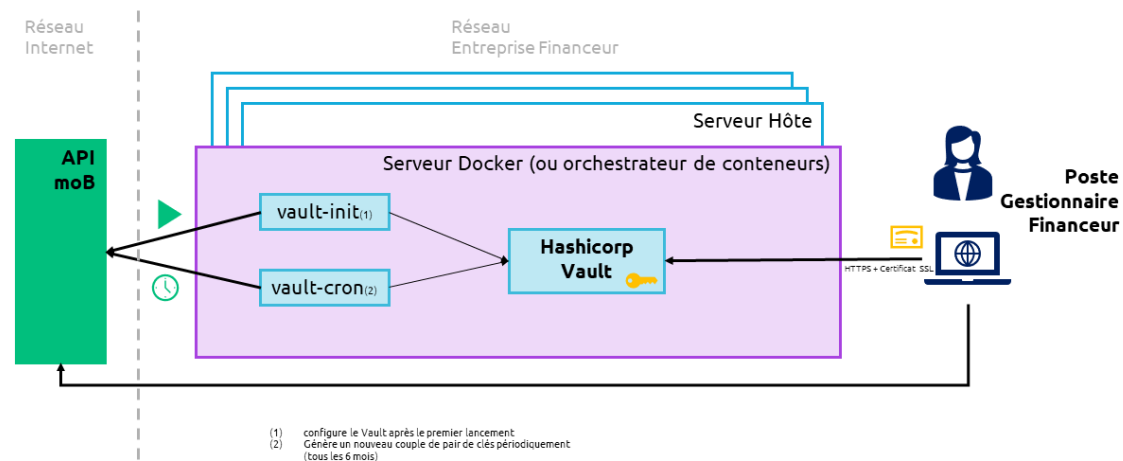
- ❑ Il s'authentifie au Vault à l'aide d'un certificat avec le rôle admin
- ❑ Il vérifie qu'une clé a déjà été créée dans le vault (depuis le script **init-vault.sh**), sinon il s'arrête
- ❑ Il vérifie si la clé est expirée. Si la clé est encore valide, il s'arrête.
- ❑ Si la clé est expirée, il crée une nouvelle version de la clé et l'envoie à MOB en remplaçant l'ancienne clé expirée par celle qui vient d'être créée

### 3.3. Schéma d'Architecture

Ci-dessous le schéma d'architecture de mise en place. Il faut parfois adapter la configuration réseau en fonction du SI Financier afin que les flux puissent transiter.



#### ENABLER « Key Manager » – Schéma d'Architecture



Presentation Title | Author | Date

Company Confidential | © Capgemini 2021. All rights reserved | 27

## 4. Vue d'ensemble des actions requises

### 4.1. Eléments à récupérer auprès de moB

- Un client créé dans l'environnement moB associé avec son **client\_id**, et son **client\_secret**.
- La liste du ou des financeurs créés dans le SI moB et leurs identifiants **funderId**
- L'archive mcm-vault-v0.4.0.zip contenant le Vault moB

### 4.2. Actions à réaliser par le Financier

- Installation de **Docker** (20.10+) et **Docker Compose** sur un serveur de son choix, accessible dans le réseau des collaborateurs gestionnaires
- Décompresser l'archive du Vault moB reçue
- Seules les personnes autorisées par le financier doivent pouvoir déchiffrer les justificatifs fournis par un citoyen lors de la souscription à une aide du financier. Pour cela il faut :
  - o Autoriser l'authentification par certificat au sein du Vault → opération incluse dans le Vault livré
  - o Générer un certificat serveur pour le nom de domaine du Vault signé par une autorité de certification. Le certificat de l'autorité de certification doit être stocké dans le magasin de certificats de la machine de l'utilisateur, dans les autorités de certification racine de confiance → un script de création de certificats est fourni pour aider (voir [Création des certificats](#))
  - o Générer des certificats clients pour les gestionnaires du Vault signé par une autorité de certification. Le certificat client du gestionnaire doit être stocké dans le magasin de



certificats de la machine de l'utilisateur, dans les certificats personnels de l'utilisateur  
→ un script de création de certificats est fourni pour aider (voir [Création des certificats](#))

## 5. Création des certificats

Pour faire fonctionner le Vault avec le TLS activé, il faut être en possession d'un certificat serveur associé au nom de domaine du Vault (**VAULT\_CERT**), de sa clé privée associée (**VAULT\_KEY**) ainsi que du certificat de l'autorité de certification qui a signé le certificat serveur (**VAULT\_ROOT\_CA**). Vous

Il faudra ajouter l'autorité de certification (**VAULT\_ROOT\_CA**) dans les autorités de certification racines de confiance dans votre manager de certificats.

Ensuite pour pouvoir utiliser l'authentification par certificat, il faut aussi être en possession de certificats clients signés par votre autorité de certification (**CLIENT\_CA**) qui seront utilisés pour s'authentifier aux rôles admin ou manager. Dans la configuration actuelle, pour les rôles admin et manager on utilise la même autorité de certification pour vérifier les certificats clients.

Attention, si vous gardez la configuration actuelle avec la même autorité de certification utilisée pour les rôles admin et manager, il faudra mettre en place des contraintes sur certains champs des certificats pour les rôles pour permettre d'identifier si le client doit avoir les droits d'admin ou de manager car sans contraintes, le certificat client sera valide pour les rôles admin et manager. Vous pouvez voir [ici](#) les différentes contraintes qu'il est possible d'appliquer pour les certificats client.

Le script **createCertificates.sh** permet de créer une autorité de certification, un certificat serveur pour le nom de domaine du Vault signé par l'autorité de certification, ainsi que des certificats client admin et manager également signés par l'autorité de certification.

Pour lancer le script de création de certificat pour le nom de domaine « **vault.example.com** » :

```
.. ./createCertificates.sh vault.example.com
```

Pour l'initialisation du Vault il faut fournir un certificat client (**ADMIN\_CERT**) pour le rôle admin avec sa clé privée associée (**ADMIN\_CERT\_KEY**), car on s'authentifie en tant qu'admin pour réaliser les actions dans les scripts d'initialisation et de renouvellement de clé.

Pour les utilisateurs qui utiliseront le Vault en tant que manager, il faudra également ajouter le certificat client au format pkcs12 pour le rôle manager dans les certificats personnels de l'utilisateur.

Lorsque vous accéderez à l'interface graphique du Vault, ou que vous voudrez télécharger un justificatif, vous serez invité à sélectionner le certificat client lors de la première ouverture.

## 6. Configuration du serveur Vault

Plusieurs variables d'environnement doivent être renseignées dans un fichier **.env** pour permettre le bon fonctionnement du Key Manager.

*Valeurs fournies par moB*

**CLIENT\_ID** : Client ID du client Keycloak créé pour le financeur

**CLIENT\_SECRET** : Client Secret du client Keycloak créé pour le financeur

**FUNDER\_IDS** : Liste des ID des financeurs, séparés par une virgule, auxquels il faut envoyer une clé de chiffrement dans MOB

**IDP\_URL** : URL permettant d'obtenir un token d'identification MOB à l'aide du **CLIENT\_ID** et **CLIENT\_SECRET** à renseigner dans les appels à l'API MOB. **Attention à ne pas mettre de slash à la**

fin de l'url. (exemple : il faut mettre **https://idp.preprod.moncomptemobilite.fr** et pas **https://idp.preprod.moncomptemobilite.fr/**)

**API\_URL** : URL de l'API MOB pour envoyer la clé publique à moB. **Attention à ne pas mettre de slash à la fin de l'url.** (ex : il faut mettre **https://api.preprod.moncomptemobilite.fr** et pas **https://api.preprod.moncomptemobilite.fr/**)

#### *Valeurs internes*

**AVAILABLE\_KEYS** : Nombre de versions de clés à conserver dans le Vault. Par défaut à 2 si non renseigné (ex : 2)

**FUNDER\_TOKEN** : Un token aléatoire à générer permettant de se connecter au Key Manager en tant qu'administrateur et interagir avec, en particulier pouvoir accéder à l'UI du Vault. **Il est recommandé de le supprimer une fois l'initialisation terminée puis d'utiliser des tokens temporaires et d'en recréer en cas de besoin.**

**VAULT\_ADDR** : URL du Vault sans slash (ex : **https://vault.example.com:port** et pas **https://vault.example.com:port/**)

**VAULT\_API\_ADDR** : URL de l'API du Vault (même URL que VAULT\_ADDR)

**VAULT\_CERT** : Chemin vers l'emplacement du certificat serveur sur la machine où est lancé le Vault, à utiliser pour le TLS dans le Vault (ex : **./certs/simulation-vault.preview.moncomptemobilite.fr.crt**)

**VAULT\_KEY** : Chemin vers l'emplacement de la clé privée du certificat serveur sur la machine où est lancé le Vault (ex : **./certs/simulation-vault.preview.moncomptemobilite.fr.key**)

**VAULT\_ROOT\_CA** : Chemin vers l'emplacement du certificat de l'autorité de certification sur la machine où est lancé le Vault, utilisé pour vérifier le certificat du serveur SSL du Vault. (ex : **./certs/rootCA.pem**)

**ADMIN\_CERT** : Chemin vers l'emplacement du certificat client sur la machine où est lancé le Vault, utilisé pour s'authentifier en tant qu'administrateur (ex : **./certs/admin-client-cert.pem**)

**ADMIN\_CERT\_KEY** : Chemin vers la clé privée du certificat client administrateur sur la machine où est lancé le Vault (ex : **./certs/admin-client-key.pem**)

**CLIENT\_CA** : Chemin vers l'emplacement du certificat de l'autorité de certification sur la machine où est lancé le Vault, utilisé pour vérifier les certificats clients utilisés pour l'authentification par certificat. (ex : **./certs/client-ca.pem**). Peut être le même que **VAULT\_ROOT\_CA** mais pas nécessairement.

## 7. Démarrage du serveur Vault

Vérifier que les variables d'environnement mentionnées précédemment sont bien enregistrées dans un fichier **.env** placé dans le répertoire courant contenant le fichier **vault-docker-compose.yml**, puis lancer les commandes suivantes :

**docker volume create vault-data**

**docker-compose -f vault-docker-compose.yml up -d**

## 8.Actions de sécurisation post-installation

### 8.1. Récupération des clés de descellement

Lorsque que le script d'initialisation a été lancé, vous pouvez afficher les clés de descellement avec la commande suivante :

**`docker exec -it vault-init cat /vault/file/keys`**

```
Unseal Key 1: kMqfydmyruSgjjxjbt5ysltmUd+xr35m74vgalvFV8cdx
Unseal Key 2: IGsYBpmaEhBGZt1hMOOCtaYjiXbl7+wPR3y5DNJLMMnJ
Unseal Key 3: 6pNVzh3h6Opf+hrvGwe6lsfhHAjH39CC8KDHDA91wJhB
Unseal Key 4: W83TzRWziEtcaA4ngMTjKEBAhyPkd+byL10v+gAvuIZT
Unseal Key 5: H5UebaBpwMCG58g/xOWBac+mX6oaujBv7PvHPEKrfjy3

Initial Root Token: hvs.VnKfdg0rs5gH9YvPreH901Ds

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 "vault operator rekey" for more information.
```

**Il faut récupérer ces clés et les stocker en lieu sûr avec la commande ci-dessus.**

Elles sont enregistrées dans le fichier /vault/file/keys stockés sur le volume vault-data. Quand le vault redémarre, il va aller chercher les clés de descellement dans ce fichier afin de le desceller et qu'il soit opérationnel.

Le processus vault-init se charge pour vous d'aller récupérer les clés dans ce fichier du vault. Cependant, si vous lancer le service vault seul, les clés de descellement vous seront demandées sur l'UI, c'est ainsi que ces clés vous seront utiles.

**Si un descellement n'est pas possible, vous serez obligés de réinitialiser le Vault et les données stockées dans le Vault seront perdues, ce qui rendra impossible le déchiffrement des pièces justificatives pour les dossiers chiffrés avec une clé stocké dessus.**

### 8.2. Application de contraintes sur les rôles de certificats

Après le déroulement du script d'initialisation, le Vault est fonctionnel et peut être utilisé pour que les gestionnaires puissent déchiffrer les justificatifs en utilisant le certificat installé sur son poste.

Lors de l'initialisation, 2 rôles de certificats sont créés :

- le **rôle manager**, pour les gestionnaires,
- et le **rôle admin** pour les administrateurs du Vault.

Les 2 rôles sont créés avec la même autorité de certification. Par défaut, la configuration permet de se connecter en tant qu'administrateur à l'aide du certificat manager.

Pour empêcher cela, il faut donner des contraintes sur chaque rôle de certificat, afin d'autoriser uniquement l'administrateur de se connecter en tant qu'administrateur.

Pour ajouter une contrainte, se rendre sur l'interface graphique du Vault, se connecter à l'aide de la variable d'environnement **\$FUNDER\_TOKEN**, se rendre dans Access, menu « cert/ » et éditer les contraintes pour les rôles admin et manager. Par exemple en ajoutant une règle sur le Common Name qui doit matcher le Common Name renseigné lors de la création du certificat client avec le script **createCertificates.sh**

The screenshot displays the Vault web interface. At the top, the navigation bar includes 'Secrets', 'Access' (highlighted), 'Policies', and 'Tools'. The left sidebar shows 'ACCESS' with sub-items 'Auth Methods' and 'Leases'. The main panel is titled 'certificates' and shows the 'admin' role. The 'Edit certificate' button is highlighted with a red box. Below the 'Certificate' field, the 'Constraints' section is highlighted with a red box, containing the following fields:

- Allowed common names
- Allowed DNS SANs
- Allowed Email SANs
- Allowed names
- Allowed organizational units
- Allowed URI SANs

## 9. Monitoring

Pour connaître l'état de fonctionnement du Vault, vous pouvez regarder les logs des containers du Vault :

***docker logs vault***

***docker logs vault-init***

***docker logs vault-cron***

Le Financier peut ensuite intégrer la sortie standard de ces conteneurs dans sa solution de supervision transverse.

# 10. Configuration poste utilisateur

Il est nécessaire d'ajouter les certificats clients utilisateur sur les postes des gestionnaires amenés à traiter les demandes et à déchiffrer les justificatifs attachés.

## 10.1. Enregistrer les certificats dans le magasin de certificats

### 10.1.1. Enregistrer le certificat Manager dans les certificats personnels

En se basant sur les certificats créés avec le script **createCertificates.sh** :

- Double-clic sur **manager-client-cert.pfx**
- Choisir « **Ordinateur Local** » ou « **Utilisateur actuel** » et cliquer sur « **Suivant** »



←  Assistant Importation du certificat

#### Bienvenue dans l'Assistant Importation du certificat

Cet Assistant vous aide à copier des certificats, des listes de certificats de confiance et des listes de révocation des certificats d'un disque vers un magasin de certificats.

Un certificat, émis par une autorité de certification, confirme votre identité et contient des informations permettant de protéger des données ou d'établir des connexions réseau sécurisées. Le magasin de certificats est la zone système où les certificats sont conservés.

#### Emplacement de stockage

- ☒ Utilisateur actuel  
☐ Ordinateur local

Cliquez sur Suivant pour continuer.

Suivant

Annuler

- Cliquer encore sur « **Suivant** » pour valider le chemin vers l'emplacement du certificat à importer

← Assistant Importation du certificat

**Fichier à importer**  
Spécifiez le fichier à importer.

Nom du fichier :  
se\mcm-vault-v0.4.0\vault.walidcorp.com\manager-client-cert.pfx Parcourir...

Remarque : plusieurs certificats peuvent être stockés dans un même fichier aux formats suivants :

- Échange d'informations personnelles - PKCS #12 (.PFX,.P12)
- Standard de syntaxe de message cryptographique - Certificats PKCS #7 (.P7B)
- Magasin de certificats sérialisés Microsoft (.SST)

Suivant Annuler

- Entrer la passphrase du certificat, renseignée pendant le déroulement du script **createCertificates.sh** et cliquer sur « **Suivant** »
- Cocher « **Placer tous les certificats dans le magasin suivant** » et cliquer sur « **Parcourir** »
- Sélectionner « **Personnel** » et cliquer sur « **Ok** » puis cliquer sur « **Suivant** »
- Cliquer sur « **Terminer** »

← Assistant Importation du certificat

**Protection de clé privée**  
Pour maintenir la sécurité, la clé privée a été protégée avec un mot de passe.

Tapez le mot de passe pour la clé privée.

Mot de passe :  
pass  
☒ Afficher le mot de passe

Options d'importation :

- ☐ Activer la protection renforcée de clé privée. Une confirmation vous est demandée à chaque utilisation de la clé privée par une application, si vous activez cette option.
- ☐ Marquer cette clé comme exportable. Cela vous permettra de sauvegarder et de transporter vos clés ultérieurement.
- ☐ Protéger la clé privée à l'aide de la sécurité par virtualisation (non exportable)
- ☒ Induire toutes les propriétés étendues.

Suivant Annuler

← Assistant Importation du certificat

**Magasin de certificats**  
Les magasins de certificats sont des zones système où les certificats sont conservés.

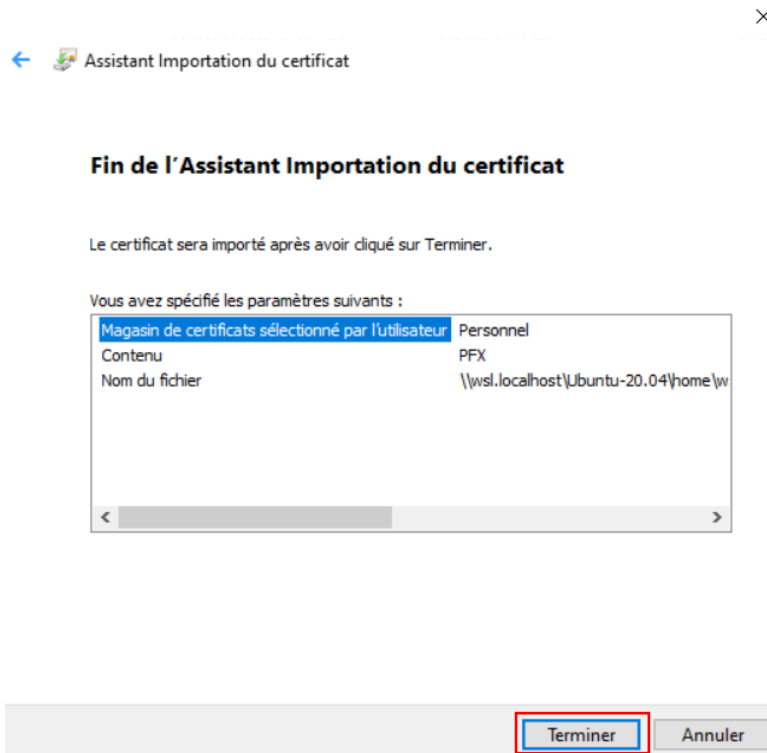
Windows peut sélectionner automatiquement un magasin de certificats, ou vous pouvez spécifier un emplacement pour le certificat.

☐ Sélectionner automatiquement le magasin de certificats en fonction du type de certificat

☒ Placer tous les certificats dans le magasin suivant

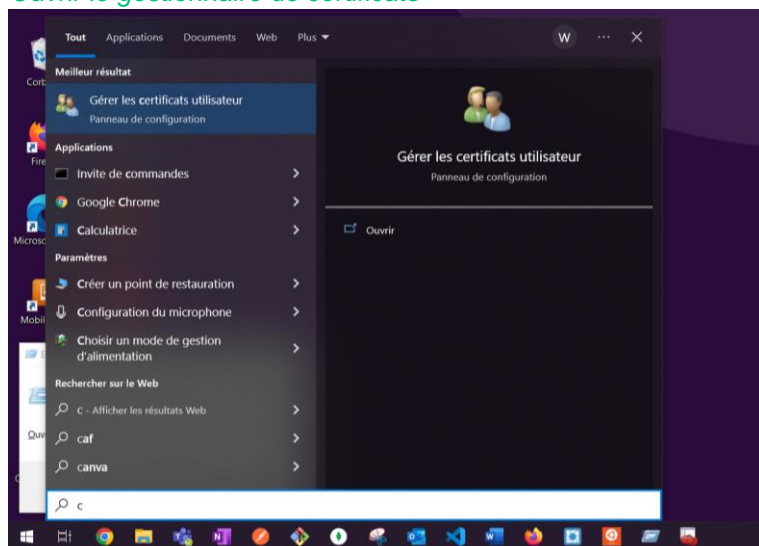
Magasin de certificats :  
Personnel Parcourir...

Suivant Annuler



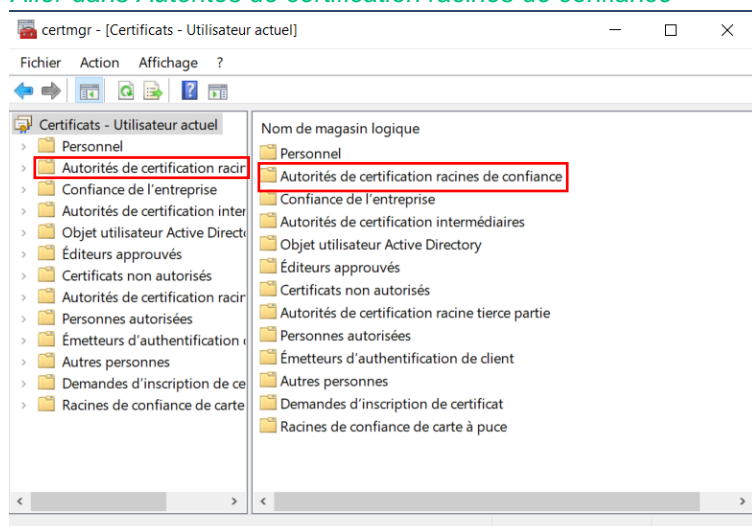
### 10.1.2. Enregistrer le certificat de l'autorité de certification dans les autorités de certification racines de confiance

Ouvrir le gestionnaire de certificats

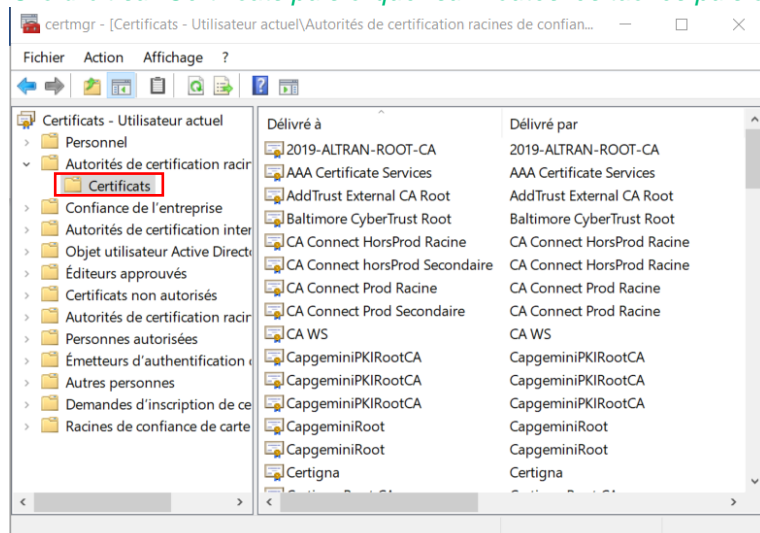




### Aller dans Autorités de certification racines de confiance

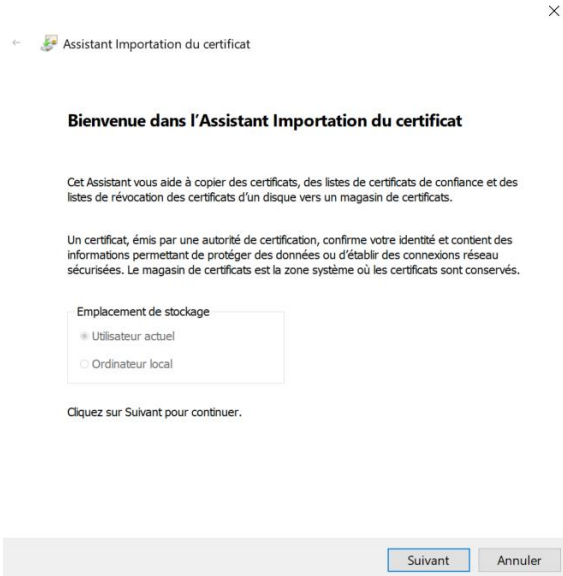


### Clic droit sur Certificats puis cliquer sur Toutes les tâches puis cliquer sur Importer

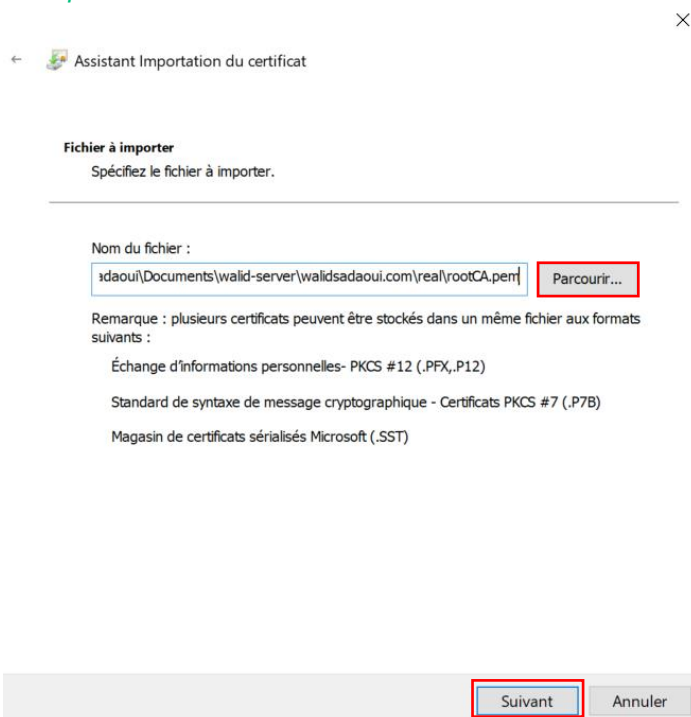




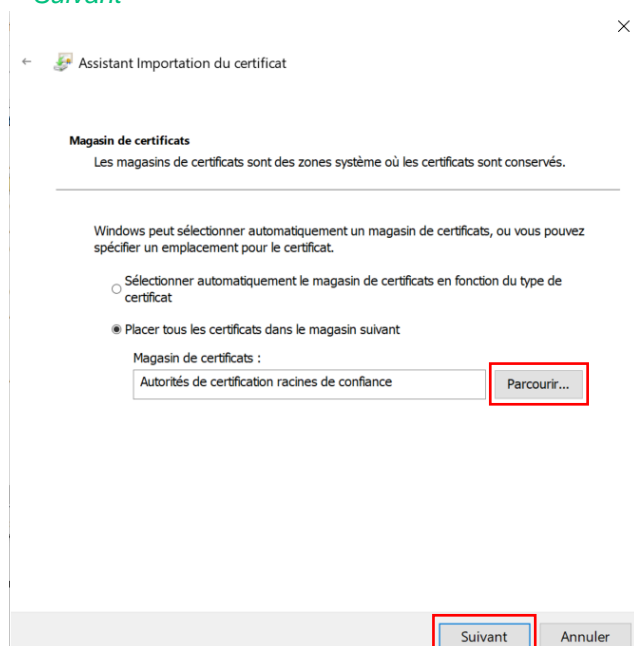
Cliquer sur Suivant



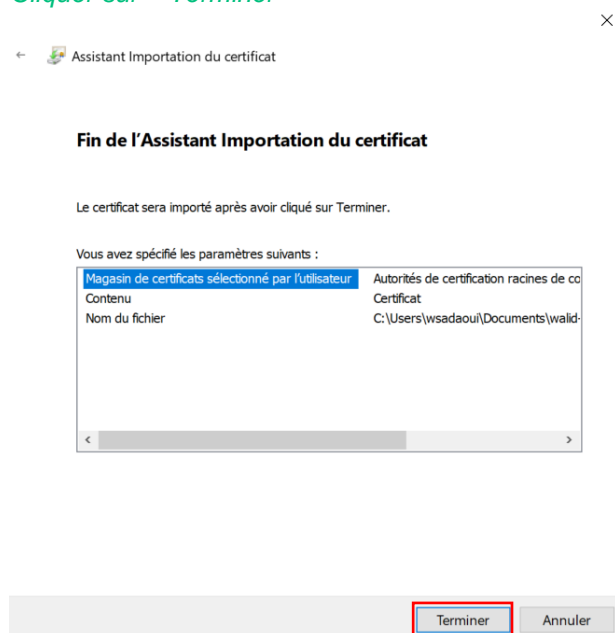
Cliquer sur Parcourir et sélectionner le fichier **rootCA.pem** créé dans le script **createCertificates.sh** et cliquer sur Suivant



*Si ce n'est pas déjà fait, sélectionner « Placer tous les certificats dans le magasin suivant », Parcourir et choisir « Autorités de certification racines de confiance » et cliquer sur « Ok ». Puis cliquer sur « Suivant »*



*Cliquer sur « Terminer »*



**A ce stade, tout est fonctionnel.**

Les sections suivantes sont présentes à titre d'information ou pour aller plus loin, notamment exécuter des actions en dehors des scripts fournis, en particulier dans le cas où le financeur s'appuie sur une autre solution de gestionnaire de clés.

# 11. Déploiement sous Windows

Comme évoqué en introduction, le déploiement sous Windows de la solution fournie est possible et fonctionnel en utilisant [Docker WSL](#).

Cependant, elle n'est pas préconisée pour des raisons de sécurité dans le cadre d'un environnement pérenne de production, et nécessite d'être ajustée en fonction de ces objectifs.

## 11.1. Utilisation de Docker WSL

Elle nécessite des actions particulières à exécuter afin qu'elle le soit.

1. Installation de WSL
2. Installation de Docker et Docker-Compose
3. Configuration réseau
4. Démarrage et ouverture du port du service Vault

*Mapper le port 8200 de l'hôte WSL au port 8200 de l'hôte windows*

***netsh interface portproxy add v4tov4 listenport=8200 listenaddress=0.0.0.0 connectport=8200 connectaddress=172.19.79.74***

*Ouvrir le port 8200 si le Firewall le bloque*

***netsh advfirewall firewall add rule name= "Open Port 8200" dir=in action=allow protocol=TCP localport=8200***

L'IP de WSL change à chaque redémarrage, donc il faut relancer ces commandes à chaque redémarrage de WSL avec l'IP de WSL récupérée à l'aide de la commande suivante lancée à l'intérieur d'un terminal WSL : ***ip addr | grep eth0***

## 11.2. Sécurisation d'un environnement

Des travaux supplémentaires sont à mettre en œuvre par le Financier afin d'atteindre un niveau de sécurité satisfaisant dans un environnement de production pérenne.

Nous listons ci-dessous les préconisations à suivre dans ce but par le Financier :

- Si déploiement dans un environnement conteneurisé Windows, ne pas utiliser WSL qui est moins sécurisé et passer par une [licence Docker](#) Desktop/Business payante

OU

- Passer par [l'installateur Windows Vault](#) pour installer le service vault à partir d'un binaire .exe
  - Cela devrait permettre de se passer des commandes netsh utilisées pour ouvrir le port d'accès au service dans le reste du SI
  - Utiliser un outil tel que Git Bash portable pour initier le Vault et exécuter le script shell fourni *init-vault.sh*
  - Mettre en place une tâche planifiée hebdomadaire permettant d'exécuter le script de régénération des clés à partir du script shell fourni *renew-key.sh*

## 12. Configuration manuelle du Vault (si souhaité)

### 12.1. Configuration complète, sans les services fournis

Cette section s'applique uniquement si pour une raison particulière vous ne souhaitez pas utiliser les services vault-init (scripts d'initialisation) et vault-cron (renouvellement de clés pour fournir à moB des versions de clés à jour), vous pouvez réaliser les actions manuellement.

Dans ce cas, attention, si des informations erronées sont envoyées à moB ou si aucun renouvellement de clé n'est envoyé chaque 6 mois à moB, les citoyens ne pourront plus joindre des justificatifs et souscrire à vos aides tant qu'une clé valide ne sera pas envoyée à moB.

#### 12.1.1. En utilisant la CLI du Vault

Le Vault CLI est utilisable lorsqu'on se trouve à l'intérieur du conteneur docker du Vault. Vous pouvez également installer le Vault CLI sur votre machine.

*Connaître l'état du Vault*  
**`vault status`**

*Initialiser le Vault*  
**`vault operator init`**

*Desceller le Vault*  
**`vault operator unseal`**

*Se connecter avec le token root*  
**`vault login $ROOT_TOKEN`**

*Créer la policy admin avec le fichier de configuration admin-policy.hcl*  
**`vault policy write admin /vault/config/admin-policy.hcl`**

*Créer la policy manager avec le fichier de configuration manager-policy.hcl*  
**`vault policy write admin /vault/config/admin-policy.hcl`**

*Créer le token \$FUNDER\_TOKEN*  
**`vault token create -id $FUNDER_TOKEN -policy="admin"`**

*Se connecter au Vault à l'aide du FUNDER\_TOKEN*  
**`vault login $FUNDER_TOKEN`**

*Autoriser l'authentification par certificat*  
**`vault auth enable cert`**

*Créer un rôle admin pour l'authentification par certificat*  
**`vault write auth/cert/certs/admin display_name=admin policies=admin certificate=@/etc/ssl/certs/client-ca.pem`**

*Créer un rôle manager pour l'authentification par certificat*  
**`vault write auth/cert/certs/manager display_name=manager policies=manager certificate=@/etc/ssl/certs/client-ca.pem`**

*Activer le transit secret engine pour pouvoir générer des clés*  
**`vault secrets enable transit`**

Activer le secret engine kv-v2 pour pouvoir stocker des secrets de type key/value  
**`vault secrets enable -path=kv kv-v2`**

Créer une clé rsa-2048 et la rendre exportable pour pouvoir accéder à la clé privée :  
**`vault write -f transit/keys/${CLIENT_ID} type=rsa-2048 exportable=true`**

Créer une nouvelle version de la clé  
**`vault write -f transit/keys/${CLIENT_ID}/rotate`**

Toutes ces commandes sont également disponibles depuis l'API du Vault, par exemple pour récupérer les différentes versions de la clé publique créée :

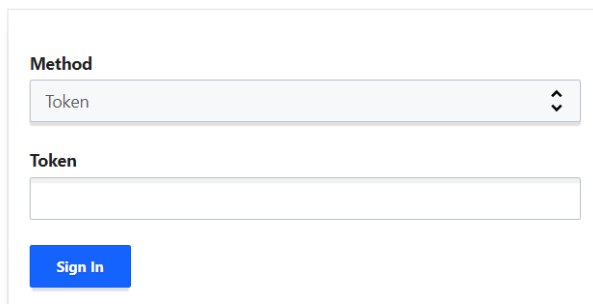
**`GET ${VAULT_ADDR}/v1/transit/keys/${CLIENT_ID}`**

Documentation de l'API du vault : <https://www.vaultproject.io/api-docs>

### 12.1.2. En utilisant l'interface graphique du vault

Se connecter au Vault à l'aide du token root la première fois ou avec le **`$FUNDER_TOKEN`**

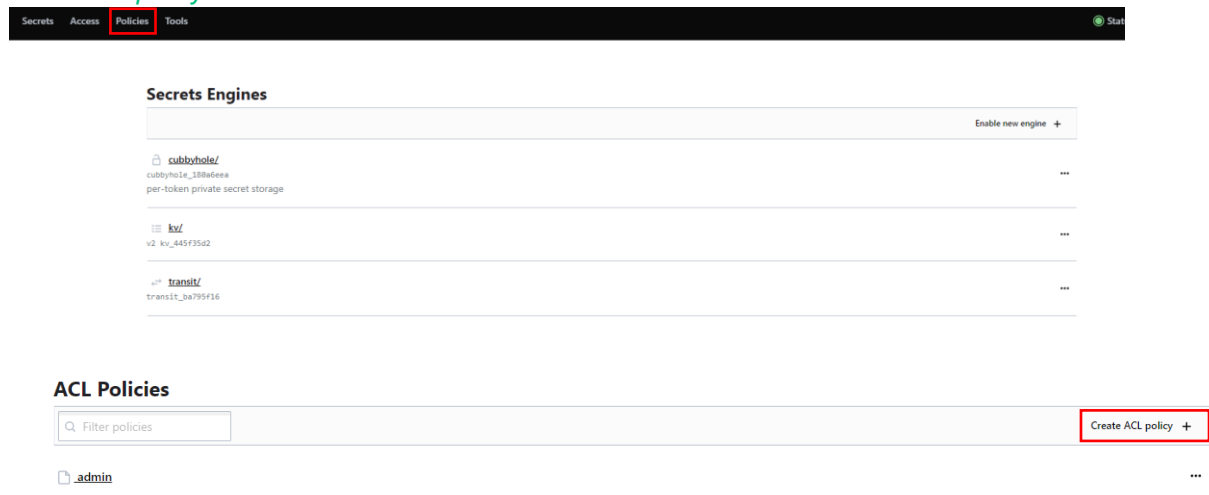
#### Sign in to Vault



The form contains a 'Method' dropdown menu with 'Token' selected, a 'Token' text input field, and a blue 'Sign In' button.

Contact your administrator for login credentials

#### Créer une policy



The screenshot shows the Vault web interface with the 'Policies' tab selected in the top navigation bar. Below the navigation bar, there are two main sections: 'Secrets Engines' and 'ACL Policies'. The 'Secrets Engines' section lists three engines: 'cubbyhole/' (per-token private secret storage), 'kv/' (v2 kv\_645f35d2), and 'transit/' (transit\_9a795f16). The 'ACL Policies' section has a search bar labeled 'Filter policies' and a red-bordered button labeled 'Create ACL policy +'. Below this, there is a list of policies, with the first one being '\_admin'.

## Create ACL policy

Name

admin

Policy

Upload file

```
63 path "transit/*" {
64   capabilities = ["create", "read", "update", "list"]
65 }
66
67 # Manage secrets engines
68 path "sys/mounts/*"
69 {
70   capabilities = ["create", "read", "update", "delete", "list", "sudo"]
71 }
72
73 # List existing secrets engines.
74 path "sys/mounts"
75 {
76   capabilities = ["create", "read", "update", "delete", "list", "sudo"]
77 }
```

You can use Alt+Tab (Option+Tab on MacOS) in the code editor to skip to the next field

Create policy


Cancel

## Autoriser l'authentification par certificat

Secrets Access Policies Tools

### Secrets Engines

Enable new engine +

 cubbyhole/  
cubbyhole\_108a6eea  
per-token private secret storage

Secrets Access Policies Tools

ACCESS


Auth Methods


### Authentication Methods


Enable new method +


### Enable an Authentication Method


Generic

  
AppRole  
☐


  
JWT  
☐


  
OIDC  
☐


  
TLS  
Certificates  
☒


  
Username &  
Password  
☐


Cloud

  
AllCloud  
☐


  
AWS  
☐


  
Azure  
☐


  
Google  
Cloud  
☐


  
GitHub  
☐

Infra

  
Kubernetes  
☐

  
LDAP  
☐

  
Okta  
☐

  
RADIUS  
☐

Next

## Enable TLS Certificates Authentication Method

Path

cert

Method Options

Enable Method

Back

Créer un rôle de certificat : donner un nom, un certificat d'une autorité de certification, des contraintes sur les certificats clients et la policy associée à ce rôle

ACCESS

Auth Methods

Leases

methods

cert

Certificates Configuration

Create certificate +

certificates

### Create certificate

Name ⓘ

admin

Allowed metadata extensions ⓘ

Add

Certificate ⓘ

Enter as text

```
-----BEGIN CERTIFICATE----- MIIDuTCCAqGgAwIBAgIUApjSGlx9IFjYi5IAWiiULNzLQwDQYJKoZIhvcNAQEL
BQAwbDELMAkGA1UEBhMCRIxDbzANBgNVBAMkZyYyW5jZTEOMAwGA1UEBwwFUGFy
aXMxEDAQBgNVBAoMB1dTIENvcnAxEDAQBgNVBAsMB1dTIHvbWUxGDAWBgNVBAMM
D1dTIENvcnAgUm9vdCBDQTAeFw0yMjEwMDMxNDIwMjRlMjRlMjRlMjRlMjRlMjRl
MGwxGzAJBgNVBAYTAkZSMQ8wDQYDVQQIDAZGcmFuY2UxGDAWBgNVBAMkZyYyW5jZTEOMAwGA1UEBwwFUGFy
```

Enter the value as text

Display name ⓘ

admin

^ Hide Constraints

Allowed common names ⓘ

Add

Allowed DNS SANs ⓘ

Add

Allowed Email SANs ⓘ

Add

Allowed names ⓘ

Add

Allowed organizational units ⓘ

Add

Allowed URI SANs ⓘ

Add



☐ **Generated Token's Explicit Maximum TTL**  
Vault will use the default lease duration.

☐ **Generated Token's Maximum TTL**  
Vault will use the default lease duration.

☐ **Do Not Attach 'default' Policy To Generated Tokens** ⓘ

**Maximum Uses of Generated Tokens** ⓘ

☐ **Generated Token's Period**  
Vault will use the default lease duration.

**Generated Token's Policies** ⓘ

admin

☐ **Generated Token's Initial TTL**  
Vault will use the default lease duration.

**Generated Token's Type** ⓘ

Save

Cancel

*Autoriser le secret engine transit*

## Secrets Engines

Enable new engine +

**cubbyhole/**  
cubbyhole\_188a6eea  
per-token private secret storage

...

## Enable a Secrets Engine

**Generic**

KV  
☐

PKI Certificates  
☐

SSH  
☐

Transit  
☒

TOTP  
☐

**Cloud**

Active Directory  
☐

AliCloud  
☐

AWS  
☐

Azure  
☐

Google Cloud  
☐

Google Cloud KMS  
☐

**Infra**

Consul  
☐

Databases  
☐

Nomad  
☐

RabbitMQ  
☐

Next

## ↔ Enable Transit Secrets Engine

### Path

transit

▼ Method Options

Enable Engine

Back

Accéder au transit secret engine

## Secrets Engines



**cubbyhole/**

cubbyhole\_774c39c7

per-token private secret storage



**transit/**

transit\_daa14512

Créer une nouvelle clé rsa-2048 exportable

< secrets < transit

↔ **transit**

Keys Configuration

Q Filter keys

Create encryption key +

< transit

## Create encryption key

### Name

simulation-maas-backend

☐ Auto-rotation period

Key will never be automatically rotated

### Type

rsa-2048

☒ Exportable

Create encryption key

Cancel

## Accéder au détail de la clé créée

[secrets](#) [transit](#)

↔ **transit**

Keys Configuration

Filter keys

🔍 simulation-maas-backend

## Accéder à la liste des versions de la clé

[transit](#) [simulation-maas-backend](#)

**Encryption key** simulation-maas-backend

Key Actions

Details

Versions

**Encrypt**  
Looks up wrapping properties for the given token

**Decrypt**  
Decrypts the provided ciphertext using this key

**Datakey**  
Generates a new key and value encrypted with this key

**Rewrap**  
Rewraps the ciphertext using the latest version of the named key

**Sign**  
Get the cryptographic signature of the given data

**HMAC**  
Generate a data digest using a hash algorithm

**Verify**  
Validate the provided signature for the given data

**Export Key**  
Get the named key

## Créer une nouvelle version de la clé

[transit](#) [simulation-maas-backend](#)

**Encryption key** simulation-maas-backend

Key Actions

Details

Versions

Rotate encryption key ^

Edit encryption key >

🕒 <b>Version 7</b>	4 minutes ago	✓ Current minimum decryption v	...
🕒 <b>Version 8</b>	less than a minute ago		...

⚠️ **Rotate this key?**  
After rotation, all key actions will default to using the newest version of the key.

Rotate

Cancel

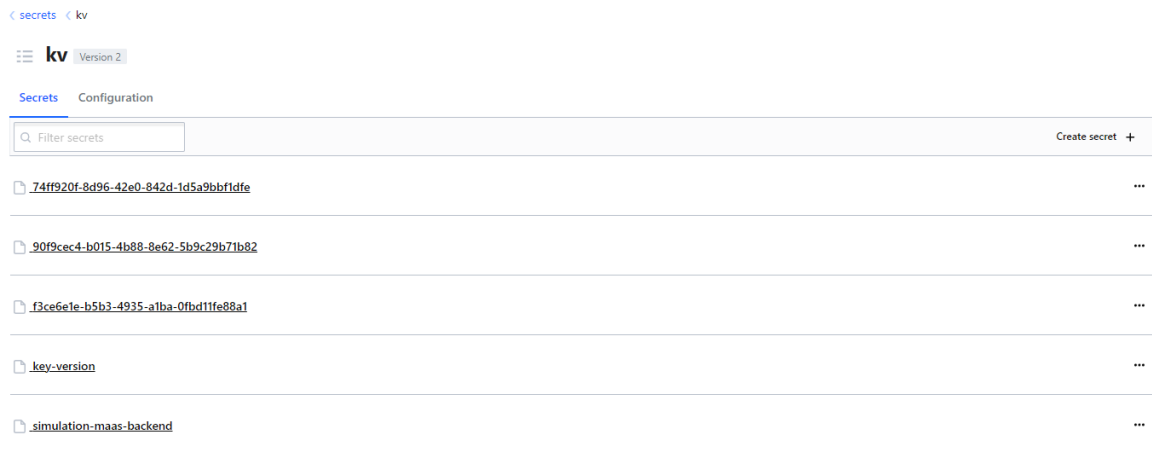
## 12.2. Configuration complémentaire, à partir des services fournis

Cette section s'applique uniquement si pour une raison particulière, en complément de l'utilisation des services vault-init (scripts d'initialisation) et vault-cron (renouvellement de clés pour fournir à moB des versions de clés à jour), le Financier souhaite réaliser des actions manuelles sur les données générées par les scripts d'initialisation et de renouvellement de clé. Dans ce cas, il faut veiller à ne pas réaliser des actions qui peut faire échouer les scripts plus tard.

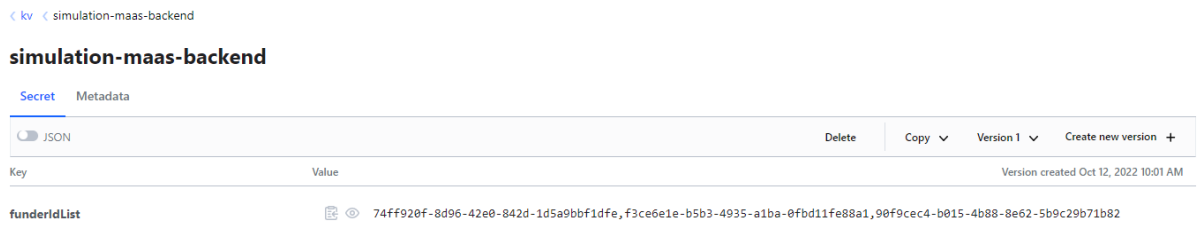
Attention, suite à ces actions, des informations erronées sont envoyées à moB ou si aucun renouvellement de clé n'est envoyé dans les 6 mois par le service vault-cron, les citoyens ne pourront plus joindre des justificatifs et souscrire à aux aides du Financier tant qu'une clé valide ne sera pas envoyée à moB.

### 12.2.1. Données générées

Lors des scripts d'initialisation et de renouvellement de clé plusieurs données sont stockées dans la secret engine kv.



- Path  **$\${CLIENT\_ID}$**  : dans ce path est stocké la liste des identifiants de financeur à qui on envoie une clé de chiffrement dans une clé **funderIdList**. La valeur de **funderIdList** est mise à jour lorsque vous modifiez la variable d'environnement  **$\${FUNDER\_IDS}$**  et que vous relancez le script d'initialisation ou que le script de renouvellement de clé est lancé par le cronjob.



- Path **key-version** : dans ce path est enregistré la version de la clé utilisée ainsi que l'id de cette clé lors du stockage de la clé dans MOB. A chaque rotation de la clé, une nouvelle version est créée avec la nouvelle version et son Id associé.



- Path  **$\${FUNDER\_ID}$**  : Pour chaque identifiant de financeur de **funderIdList** une entrée est créée pour stocker le numéro de version et l'id de la clé utilisée pour ce financeur. Cela permet aux scripts de relancer un envoi de la clé à MOB si ce financeur ne possède pas la dernière version de la clé.

< kv < f3ce6e1e-b5b3-4935-a1ba-0fbd11fe88a1

## f3ce6e1e-b5b3-4935-a1ba-0fbd11fe88a1

Secret Metadata

JSON

Delete

Copy

Version 1

Create new version +

Version Data

```
{
  "keyPairId": "c297fa53-e4c1-4680-985f-bcc4679668a1",
  "version": 1
}
```

## 12.2.2. Rotation de la clé de chiffrement

< transit < simulation-maas-backend

### Encryption key simulation-maas-backend

Key Actions Details Versions

				Rotate encryption key ^	Edit encryption key >
Version 7	4 minutes ago	Current minimum decryption v	...	<div><div>Rotate this key?</div><div>After rotation, all key actions will default to using the newest version of the key.</div><div>Rotate Cancel</div></div>	
Version 8	less than a minute ago		...		

Lorsque que vous effectuez une rotation de la clé de chiffrement manuellement, vous devez également mettre à jour le path kv/key-version en spécifiant la nouvelle version de la clé et un identifiant associé pour permettre aux scripts de toujours fonctionner

< secrets < kv

kv Version 2

Secrets Configuration

Filter secrets

Create secret +

74ff920f-8d96-42e0-842d-1d5a9bbf1dfe

90f9cec4-b015-4b88-8e62-5b9c29b71b82

f3ce6e1e-b5b3-4935-a1ba-0fbd11fe88a1

key-version

simulation-maas-backend

< kv < key-version

### key-version

Secret Metadata

JSON

Delete

Copy

Version 1

Create new version +

Version Data

```
{
  "keyPairId": "c297fa53-e4c1-4680-985f-bcc4679668a1",
  "version": 1
}
```

< kv < key-version

### Create new version

JSON

#### Version Data

```
1 {  
2   "keyPairId": "c23daeze-e4c1-4680-985f-bcc423de9668a1",  
3   "version": 2  
4 }
```

Save

Cancel

## 12.2.3. Envoi de la clé publique à moB

Si vous envoyez manuellement une clé publique à moB (Voir partie [Envoyer une clé publique à moB](#)), Vous devez référencer le bon numéro de version avec son keyPairId associé pour ne pas créer d'erreur dans les scripts. Par exemple si la dernière version du path **kv/key-version** est :

< kv < key-version

### key-version

Secret Metadata

JSON

Delete

Copy

Version 1

Create new version +

#### Version Data

```
{  
  "keyPairId": "c297fa53-e4c1-4680-985f-bcc4679668a1",  
  "version": 1  
}
```

Alors dans le body de l'endpoint **PUT /v1/funders/\${FUNDER\_ID}/encryption\_key** il faudra bien spécifier :

```
{  
  "id": "c297fa53-e4c1-4680-985f-bcc4679668a1",  
  "version": 1,  
  "publicKey": "-----BEGIN PUBLIC KEY-----\nMIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEApk  
UKTww771tjeFsYFCZq\inn76SSpOzolmtf9VntGIPbP5j1dEr6jAuTthQPoIDaEed6P44yyL3/1GqWJMGbRbfnn8  
qqvnu8dH8xB+c9+er0tNezafK9eK37RqzsTj7FNW2Dpk70nUYncTiXxjf+ofLq\lnsokEllp2zHPEZce2o6jAloFO  
V90MRhJ4XcCik2w3IjxdJSifBYX2/rDgEVN0T85\lnOOd9ChaYpKCPKKfnpvhjEw+KdmzUFP1u8aao2BNKyI2  
C+MHuRb1wSlu2ZAYfHgoG\lnX6FQc/nXeb1cAY8W5aUXOP7ITU1EtlucD8WuxXMfIS446vyfCmJWt+OFyveq  
gJ4n\lnowIDAQAB\n-----END PUBLIC KEY-----\n",  
  "expirationDate": "2022-12-17T14:22:01Z",  
  "lastUpdateDate": "2022-06-17T14:22:01Z",  
  "privateKeyAccess": {  
    "loginURL": "https://vault.example.com/v1/auth/cert/login",  
    "getKeyURL": "https://vault.example.com/v1/transit/export/encryption-key/keyname/1" }  
}
```

Vous devrez également pour chaque identifiant de financeur pour lesquels vous avez envoyé une clé manuellement à MOB, mettre à jour le path **kv/\${FUNDER\_ID}** avec les nouvelles informations à jour pour ce financeur (**version** et **keyPairId**) :

< secrets < kv

kv Version 2

Secrets Configuration

Filter secrets Create secret +

74ff920f-8d96-42e0-842d-1d5a9bbf1dfe ...

90f9cec4-b015-4b88-8e62-5b9c29b71b82 ...

f3ce6e1e-b5b3-4935-a1ba-0fbd11fe88a1 ...

key-version ...

simulation-maas-backend ...

< kv < 74ff920f-8d96-42e0-842d-1d5a9bbf1dfe

74ff920f-8d96-42e0-842d-1d5a9bbf1dfe

Secret Metadata

JSON Delete Copy Version 1 Create new version +

Version Data

```
{
  "keyPairId": "c297fa53-e4c1-4680-985f-bcc4679668a1",
  "version": 1
}
```

< kv < 74ff920f-8d96-42e0-842d-1d5a9bbf1dfe

Create new version

JSON

Version Data

```
1 {
2   "keyPairId": "a697fa53-e4c1-4680-985f-edc3456668a1",
3   "version": 2
4 }
```

Save Cancel

## 13. Envoyer une clé publique à moB (hors Vault)

Cette étape est déjà réalisée automatiquement par les services vault-init et vault-cron fournis. Les détails ci-dessous ne sont présents qu'à des fins d'informations.

### 13.1. Obtention du token d'authentification

URI et méthode

POST \${IDP\_URL}/auth/realms/mcm/protocol/openid-connect/token

### Client credentials

Il s'agit de s'authentifier en tant que client via un compte de service en fournissant dans le corps de la requête IDP :

- + **grant\_type** : client\_credentials
- + **client\_id** : <identifiant de l'application client>
- + **client\_secret** : <secret de l'application client>

### Réponse

Récupérer le champ **access\_token** pour l'utiliser en tant que jeton d'accès dans l'endpoint d'envoi de la clé publique à MOB ci-dessous.

## 13.2. Envoi de la clé publique à moB

### URI et méthode

PUT \${API\_URL}/v1/funders/\${FUNDER\_ID}/encryption\_key

### En-têtes

- + **Authorization Bearer** : <jeton d'accès>

### Corps

Propriété	Type	Description
id	string	Identifiant de la clé
version	number	Version de la clé
publicKey	string	Clé publique du financeur
expirationDate	Date	Date d'expiration de la clé
lastUpdateDate	Date	Date de dernière mise à jour de la clé
privateKeyAccess	Object(JSON)	Objet contenant les URLs permettant de récupérer la clé privée du financeur <ul style="list-style-type: none"><li>? loginURL (string) : URL de connexion au Vault</li><li>? getKeyURL (string) : URL d'accès à la clé privée associée à la clé publique fournie dans le champ publicKey</li></ul>

### Exemple :

```
{
  "id": "c297fa53-e4c1-4680-985f-bcc4679668a1",
  "version": 1,
  "publicKey": "-----BEGIN PUBLIC KEY-----
MIICljANBgkqhkiG9w0BAQEFAAOCAg8AMIICCgKCAgEAYtHQWS40nUZpO9emt6XWnRiaoJ
fUpTN8NftTLBrVnI876FMPM5YIptpBe6LyY/kvpmUPZLaRIJ3tOkdqj1eTR\n1Vlyc03nWAh4Sbd/e
WJU5gqw89Jqaqyi72Xon3IdlSTgjO/X5bIMAohGH2WVDsW\nDDW7KAMMar9ExemiN9VgUoy
YpwffxJSZVkb5egK5noHnPbyVPXvzPQPbG6xKKD2\nXR8y+YPNfpWSbUVS7kXZq9DvGZdjR
ISze8U7734ddWHEiUSuSNg/i7TZdvN7P88\nUoVhY4/DYpjNEcupniRUXQOUyKKdUuCcYDa1M
+8FbFWZazSk2MYvSNEXkULj+rKV\n2xfUMnvH5yH4OAZAWG0Mp6JXFYXHsoEF7YfOhxJKo5w
xMGv1rPvaRcPNeMfqacJX\n7zy1XX40Q18kwu/onKXS2BQxB5UuxYXUo5TA3YExUZIZPcoiwai
```



```
prBNcRoQFSss\n0SKI/G5bQX3IX0OUN8vaUuJlnJf3g/vrY2VHm2hgAY9+JfdphdXw87Pn5SvQK
qFg\nYffmMX3mxuf/nO5h9yADrgBrRfDdxjxjGifVNCymCStNVJNHhXJN/dCzz4IPkok\nU6UzMJk
zJleQR6X8vyrw40P4EPEU2+fzJhYRMncU4srw1fISclixd89tgda2PR0D\noOYOoDTWddvLzprlDy
KqilUCAwEAAQ==\n-----END PUBLIC KEY-----",
"expirationDate": "2022-12-17T14:22:01Z",
"lastUpdateDate": "2022-06-17T14:22:01Z",
"privateKeyAccess": {
  "loginURL": "https://vault.example.com/v1/auth/cert/login",
  "getKeyURL": "https://vault.example.com/v1/transit/export/encryption-key/keyname/1"
}
}
```

### Règles de gestion

- ❓ Les champs **id**, **version**, **publicKey**, **expirationDate** et **lastUpdateDate** sont obligatoires.
- ❓ L'objet **privateKeyAccess** est obligatoire sauf pour les financeurs qui utilisent leur propre SIRH (et n'utilisent donc pas l'interface financeur MOB)

### Réponse

La réponse succès (code HTTP 204) ne renvoie aucun contenu.

# 14. Troubleshooting

## 14.1.Problématiques réseau

### 14.1.1. Accès au Vault derrière un proxy

#### *Dans WSL*

Pour avoir accès au réseau à l'intérieur d'une instance WSL, il faut renseigner les paramètres proxy dans les fichiers suivants : `/etc/wgetrc`, `~/.docker/config.json`, `/etc/systemd/system/docker.service.d/http-proxy.conf`, `/etc/environment`

Les variables d'environnement à renseigner sont : `http_proxy`, `https_proxy`, `http_PROXY`, `HTTPS_PROXY`, `no_proxy`, `NO_PROXY`. Les variables de proxy http sont à renseigner au format : `http://${username}:${password}@${proxy_url}:${port}`

Encoder les caractères spéciaux dans le username et le password, exemple : @ → %40

#### *Dans un conteneur de Docker*

Pour avoir accès au réseau à l'intérieur d'un conteneur Docker, il faut renseigner au lancement du conteneur :

- `HTTP_PROXY_AUTH=basic*:username:password` (ne pas encoder les caractères spéciaux ici)
- `http_proxy`, `https_proxy`, `http_PROXY`, `HTTPS_PROXY`, `no_proxy`, `NO_PROXY`. Les variables de proxy http sont à renseigner au format : `http://${proxy_url}:${port}`

### 14.1.2. Installation hors-ligne des packages linux dans le conteneur Vault

#### *Lancer un conteneur alpine avec accès internet pour télécharger les packages*

```
docker run -it -v /apk-packages:/usr/local/bin/apk-packages alpine:3.14 sh
```

```
apk update
```

```
apk fetch --recursive -o /usr/local/bin/apk-packages curl jq apk-cron coreutils util-linux
```

Copier le dossier apk-packages dans le dossier contenant le livrable du Vault

#### *Modifier le Dockerfile*

```
COPY ./apk-packages/ /usr/local/bin/apk-packages/  
RUN touch repo.list && apk add --allow-untrusted --no-network --repositories-  
file=repo.list /usr/local/bin/apk-packages/*.apk && rm -rf /var/cache/apk/*
```

#### *Lancer le vault*

```
docker-compose -f vault-docker-compose.yml up --build
```

## 14.2. Nettoyage de composants Docker

Il peut être utile de nettoyer les éléments Docker inutilisés (networks, volumes, conteneurs, images), pour éviter des conflits de configuration et libérer de l'espace disque, voici quelques commandes utiles :

*Supprimer les conteneurs stoppés, les networks, images et volumes non utilisés*  
**docker system prune**

*Supprimer les volumes inutilisés*  
**docker volume prune**

*Supprimer les images non utilisées*  
**docker image prune -a**

*Supprimer les conteneurs du Vault et supprimer le volume associé vault-data avec le flag -v*  
**docker-compose -f vault-docker-compose.yml down -v**

Attention : cette commande détruit l'historique des clés enregistrées dans le Vault. Ainsi, les justificatifs des dossiers enregistrés avec ces clés ne seront plus lisibles. Pour une telle manipulation, il est souhaitable de clore les dossiers existants (validation ou rejet).

## 14.3. Accès internet dans les conteneurs Docker

Si vous n'arrivez pas à accéder au réseau internet à l'intérieur des conteneurs Docker, vous pouvez essayer de passer par le réseau du host en remplaçant dans le fichier **vault-docker-compose.yml** :

```
networks:  
  - dev_web-nw
```

Par :

```
network_mode: host
```

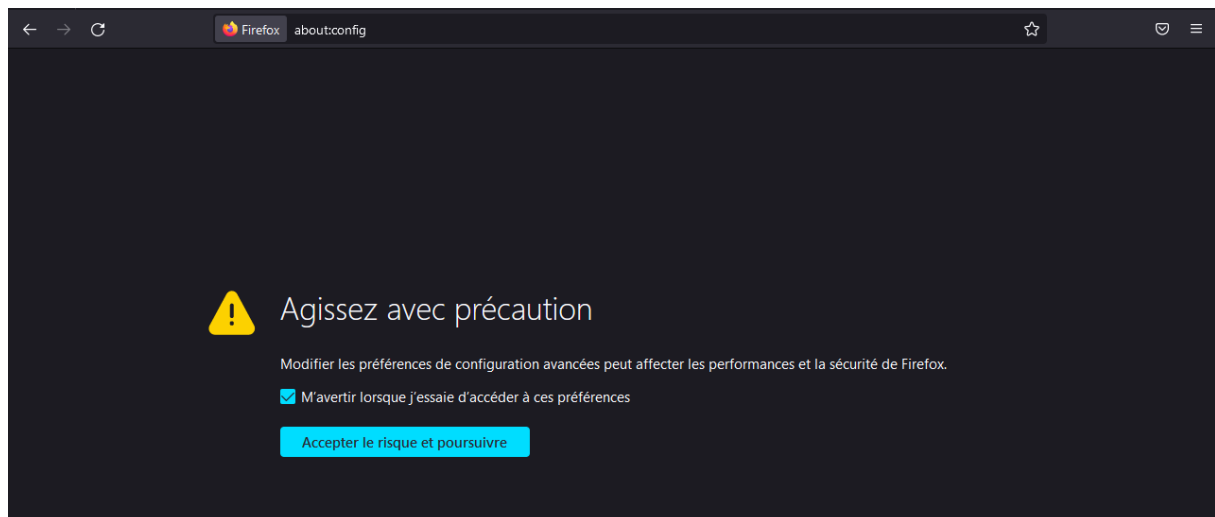
## 14.4. Mauvais choix de certificat

Si vous avez été prompté pour sélectionner un certificat mais que vous avez annulé ou sélectionné un mauvais certificat, vous devez relancer votre navigateur pour être prompté de nouveau et sélectionner le bon certificat client.

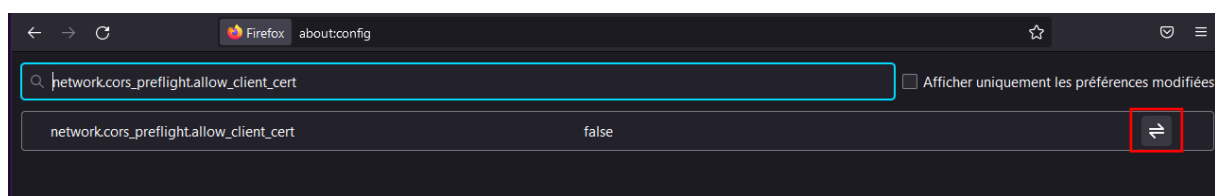
## 14.5. Firefox ne prompte pas pour sélectionner le certificat client

Si sur sa machine, le gestionnaire utilise Firefox pour accéder à MOB pour télécharger les justificatifs et traiter les demandes, vérifier que le flag "**network.cors\_preflight.allow\_client\_cert**" est à **true** :

Sur Firefox, écrire « **about:config** » et cliquer sur « **Accepter le risque et poursuivre** »



Dans la barre de recherche, écrire : « **network.cors\_preflight.allow\_client\_cert** » et double-cliquez sur la règle ou cliquez sur l'icone à droite pour faire passer la valeur de la règle à « **true** »



## 14.6.Appels API Vault

Si vous voulez faire des API au Vault, en utilisant cURL par exemple, vous devez spécifier l'autorité de certification du certificat serveur du Vault dans la requête avec **--cacert**, par exemple :

```
curl -s -w "\n%{http_code}" --cacert /etc/ssl/certs/vault-ca.pem --header "X-Vault-Token: $AUTH_TOKEN" $VAULT_ADDR/v1/kv/funder
```

## 14.7.Tester le Vault en local

Pour tester le Vault en local, il faut attribuer un nom de domaine à votre IP locale dans votre fichier hosts :

- ❓ Sur Linux : **/etc/hosts**
- ❓ Sur Windows : **C:\Windows\System32\drivers\etc\hosts**
- ❓ Si vous utilisez WSL :
  - o Attribuer un nom de domaine à l'IP de WSL dans le fichier **C:\Windows\System32\drivers\etc\hosts**
  - o Attribuer un nom de domaine à l'IP de WSL dans le fichier **/etc/hosts**

Ensuite il faut générer un certificat serveur pour le nom de domaine choisi, le stocker dans les certificats racines de confiance, et utiliser ce nom de domaine avec le port 8200 spécifié pour accéder au Vault. Il faut renseigner les variables **VAULT\_ADDR** et **VAULT\_API\_ADDR** avec le nom de domaine et le port. (ex : **https://simulation-vault.preview.moncomptemobilite.fr:8200**)

## 14.8.Utilisation d'un proxy Traefik

Traefik arrête la connexion TLS par défaut pour communiquer en http en interne. Comme le Vault gère lui-même le TLS, il faut dire à Traefik de ne pas terminer la connexion TLS. Pour cela il faut autoriser le TLS passthrough pour le routeur TCP et ajouter une couche ServersTransport avec l'URL du vault et le certificat serveur à utiliser.

Une explication des tâches à réaliser se trouve dans cet article de blog réalisé par Traefik : [Traefik Proxy 2.x and TLS 101](#)

## 15. Références

*Vault HTTP API*

<https://www.vaultproject.io/api-docs>

*Vault CLI*

<https://www.vaultproject.io/docs/commands>

*Vault storage*

<https://www.vaultproject.io/docs/v1.10.x/configuration/storage>

*Vault TCP Listener*

<https://www.vaultproject.io/docs/configuration/listener/tcp>

*Vault Docker image*

[https://hub.docker.com/\\_/vault](https://hub.docker.com/_/vault)

*Recommandations pour l'utilisation du Vault en production*

<https://learn.hashicorp.com/tutorials/vault/production-hardening>

*Tutoriel Traefik pour gérer le TLS*

<https://traefik.io/blog/traefik-2-tls-101-23b4fbee81f1/>

*Réaliser des Backup et Restauration du Vault*

<https://learn.hashicorp.com/collections/vault/standard-procedures>